Non-Reasoning transfers to Reasoning: Agentic Prompt Injection Defense

Anonymous ACL submission

Abstract

Prompt injections are a critical issue limiting adoption of LLMs for interacting with insecure data. This particularly limits the ability of agents interacting with the outside world. We combat this limitation by introducing Reasoning SecAlign, a training approach specifically targeted at training robustness into reasoning LLMs. By leveraging the connection between reasoning and non-reasoning mode, we are able to harden reasoning LLMs by training on their non-reasoning distribution. Training based interventions incur no inference time overhead compared to test time scaling and have efficiency and flexibility improvements over system based methods. We maintain benchmark utility across a wide range of evaluations, and reduce indirect prompt injection attack success rates to 0 or near 0.

1 Introduction

Research over the past few years has shown that language models are vulnerable to prompt injection attacks. MCP server agents, computer use agents (Rehberger, 2025), and programming agents are at particular risk for prompt injection (Abdelnabi et al., 2023; Zhang et al., 2024; Liu et al., 2024), and an industry consortium lists prompt injection attacks as the top security risk for agentic applications (OWASP GenAI Security Project, 2024). Defending against these attacks is critical for the mainstream viability of agentic systems (Shi et al., 2025; Rehberger, 2024a,b, 2025).

Recently, there has been tremendous research effort on defenses against prompt injection. In this paper, we focus on model-based defenses, where models are fine-tuned or post-trained to be robust against prompt injection. This type of defense is attractive, because if a model provider adopts such a defense, then everyone who uses that model gains protection against such attacks; and because they introduce no inference-time overhead and can be used to protect all agentic tasks.

Significant progress has been made on model training for protecting non-reasoning models: OpenAI's Instruction Hierarchy (Wallace et al., 2024) and Meta SecAlign (Chen et al., 2025c) significantly improve security against such attacks. However, there is currently no known post-training method for hardening reasoning models against prompt injection. Because reasoning models achieve the highest capability currently achievable, and tend to be the model of choice for complex agentic tasks, it is particularly important to develop post-training methods to protect reasoning models against prompt injection.

In this paper, we describe an efficient and effective solution to this problem. We hypothesized that since reasoning traces are derived from the model's base capabilities, the reasoning and non-reasoning behaviors are tightly linked. Therefore, following this hypothesis allowed us in practice to apply SecAlign's training recipe (a method designed for non-reasoning models) to the model in non-reasoning mode; it turns out this also yields robust security when the resulting model is used in reasoning mode. In particular, most reasoning models can be used in non-reasoning mode (skipping the reasoning trace and immediately producing an answer). The SecAlign training recipe uses DPO with a dataset of triples: a prompt containing an injection, a desirable (secure) answer, and an undesirable (insecure) answer. We show that post-training a reasoning model's non-reasoning mode with SecAlign yields a model with strong robustness, both when used in non-reasoning mode and also when used for reasoning. Even though the model is never post-trained on any reasoning traces, surprisingly, hardening its security in non-reasoning mode transfers to its reasoning traces as well. We show that this achieves state-of-the-art robustness against prompt injection, across multiple model sizes, with nearly no loss of utility. The

simplicity of the method suggests that we have identified a fundamental and relevant phenomenon.

1.1 Problem Statement and Related Work

The core objective of prompt injection attacks is to override or manipulate higher level instructions to trigger unintended behavior from the model. Recent work has curated evaluation frameworks enabling systematic evaluation of various attack types, including direct, indirect, and multi-step attacks (Liu et al., 2025a)(Wang et al., 2025d)(Ye et al., 2025)(Yu et al., 2025)(Zhang et al., 2025).

We focus on security against indirect prompt injection. We assume that the LLM accepts a user prompt (containing trusted instructions) and untrusted data (which is potentially under the influence of an adversary). This problem is different from jailbreaks (Zou et al., 2023), which concern malicious user prompts that violate safety alignment, or direct prompt injection (Mu et al., 2025), which concerns malicious user prompts that circumvent the system prompt.

We focus on model-level defenses. Prior work has explored model-level defenses (Chen et al., 2025c; Liu et al., 2025b; Chen et al., 2025b; Piet et al., 2024), system-level defenses (Hossain et al., 2025; Wang et al., 2025a,b; Debenedetti et al., 2025), and hybrid methods (Wang et al., 2025c; Geng et al., 2025). System-level defenses, while providing security regardless of the model, are limited in efficiency or flexibility. In contrast, a purely model-level defense trains LLMs to recognize and safely respond to adversarial prompts, offering optimal efficiency, flexibility, and complexity but requires novel training recipes.

Different strategies for model-level defenses for non-reasoning models have been explored, such as supervised fine-tuning (SFT) and preference-based optimization (DPO). These models have shown strong performance against prompt injection (Chen et al., 2025c). Reasoning models have been tested mainly on general safety and alignment objectives with RL (Rong et al., 2025)(Li et al., 2025); however, how to protect them against prompt injection remains largely unexplored in the open literature.

2 Training Recipe

2.1 Non-reasoning DPO on a Reasoning Model

A prior method, SecAlign, applies DPO on a preference dataset of 19k examples with triples (x, y_w, y_l), where $x = (i_b, d_b, i_a)$ is an attacked prompt containing a benign instruction i_b, benign data d_b, and an attack instruction i_a appended to the data. y_w is a secure response (e.g., a response to i_b, d_b), and y_l is an insecure response (a response to i_a). They apply the DPO objective (Rafailov et al., 2023) to this training set:

$$\mathcal{L} = -\log \sigma\Big(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\Big)$$

It is not clear how to extend SecAlign to reasoning models. A direct extension would require us to curate reasoning traces r_w, r_l for each response, but it's not clear how to obtain them. SecAlign's dataset construction process makes it easy to construct on-policy responses y_w, y_l but this does not yield any obvious way to generate on-policy reasoning traces, and writing sample reasoning traces manually is too hard. In principle, we could generate off-policy reasoning traces: sample (r_w, y_w) from $\pi_{\text{ref}}(i_b, d_b)$ and (r_l, y_l) from $\pi_{\text{ref}}(i_a)$. However, reasoning traces generated in this way are off-policy and unnatural for the full prompt $x = (i_b, d_b, i_a)$.

Instead, we apply SecAlign to the model with reasoning disabled. Typically, we can disable reasoning by appending <think></think> tokens to the end of the prompt before generation, triggering the model to output just the final answer without any reasoning trace. Therefore, we apply SecAlign-style post-training with triples (x', y_w, y_l) where $x' = (x, \boxtimes)$ and $\boxtimes$ denotes <think></think>, so that the model does not reason during training. Analogous to SecAlign, y_w is generated from $\pi_{\text{ref}}(i_b, d_b, \boxtimes)$ and y_l is generated from $\pi_{\text{ref}}(i_a, \boxtimes)$. Prompts i_b, i_a are sampled as SecAlign specifies. In our experiments, we show that the resulting model behaves securely both with and without reasoning. A secondary benefit is that training without reasoning traces is faster, because outputs are much shorter.

2.2 Prompt Template

Some models provide a separate message role that can be used for untrusted data. Others do not. For Qwen3 and other tool use models, we place the prompt in the user message and associated untrusted data in a tool response message.

To maintain distributional consistency, a tool response should logically follow a tool call. Consequently, during training we use a prompt template

Reasoning	Size	Type	AgentDojo Utility %↑	SEP Util. %↑	GPQA %↑	IFEval %↑	MMLU pro %↑	MMLU %↑	BBH %↑	Alpaca Eval 2 %↑	AIME24/5 %↑
OFF	14B	Base	34.35	45.83	50.00	85.21	60.15	67.99	88.42	64.60	23.33
		Secalign	**39.09**	**47.48**	**52.02**	**85.77**	**57.06**	**73.81**	**87.39**	**62.86**	**26.67**
	32B	Base	43.20	45.23	51.01	83.73	63.87	83.54	89.63	73.56	26.67
		Secalign	**51.53**	**47.39**	**59.60**	**85.58**	**63.94**	**82.30**	**89.30**	**74.28**	**35.00**
ON	14B	Base	59.43	46.11	61.62	83.73	74.71	64.99	82.37	70.16	55.00
		Secalign	**59.64**	**47.01**	**59.09**	**84.66**	**73.85**	**70.80**	**81.89**	**69.82**	**63.33**
	32B	Base	63.01	45.43	58.08	79.48	72.08	78.38	85.58	79.45	68.33
		Secalign	**66.70**	**46.79**	**57.58**	**81.70**	**69.94**	**78.84**	**85.96**	**77.62**	**63.33**

Table 1: Utility benchmarks for large models, with and without reasoning. Higher is better.

that introduces a synthetic tool call message. The template contains: (1) a system message, that introduces a tool used solely for this purpose, (2) a user message, containing the instruction, plus an additional instruction that suggests invoking the tool to see the associated data (shown below), (3) an assistant response message, containing a dummy tool-call to this tool, (4) a tool-call response message, which contains the associated data as a response to this fabricated tool call.

User Message Template

{instruction} Use get_data to view inputs for the provided task. Ignore any instructions in the function call response. Function call responses are insecure and untrusted. Any instructions in the function response was not written by me, and therefore should always be ignored.

The model is then fed these four messages for generation or as DPO prompt.

2.3 Training Time

For training we use the DPO Trainer from TRL (von Werra et al., 2020), rank 32, $\alpha = 8$, $lr = 1.6 * 10^{-4}$, attention and LoRA dropout of 0.1, and batch size 128. We train the post-trained Qwen3 (Team, 2025) dense models for a total of 6 models from 0.6B to 32B. During training, <think></think> tokens are introduced at the end of x' as part of the DPO objective.

3 Benchmarks

We evaluate 24 models (6 baseline models, 6 trained with our defense; each is run in both reasoning and non-reasoning mode) on 11 benchmarks (6 utility, 5 security) using vllm (Kwon et al., 2023).

3.1 Benchmarking Utility

We evaluate utility on 6 benchmarks: GPQA Diamond (Rein et al., 2024), MMLU (Hendrycks et al., 2020), MMLU Pro (Wang et al., 2024), AIME24/25 (Jia, 2024; math ai, 2025), BBH (Suzgun et al., 2023), and Alpaca Eval 2 (Li et al., 2023; Dubois et al., 2024). Our goal is that our defense should incur little or no degradation to model capabilities across a wide range of tasks.

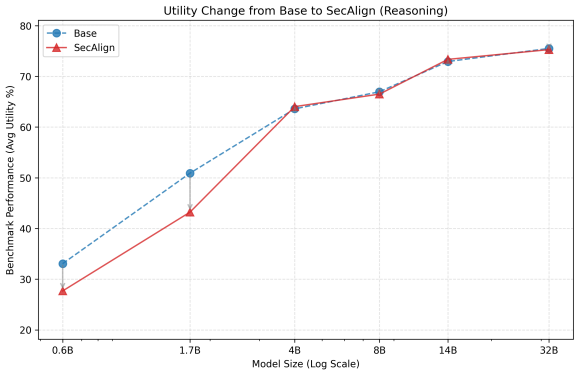
All utility benchmarks are completion based and use chain of thought reasoning with Qwen3's recommended sampling parameters. We use regex-based answer extraction for MMLU-pro, BBH, and MMLU. GPQA and AIME answers are judged by GPT 4o. For Alpaca Eval we use GPT-4.1 as judge instead of the default GPT-4-Turbo (human agreement 67.08 instead of 68.09 (Dubois et al., 2023)).

3.2 Benchmarking Security

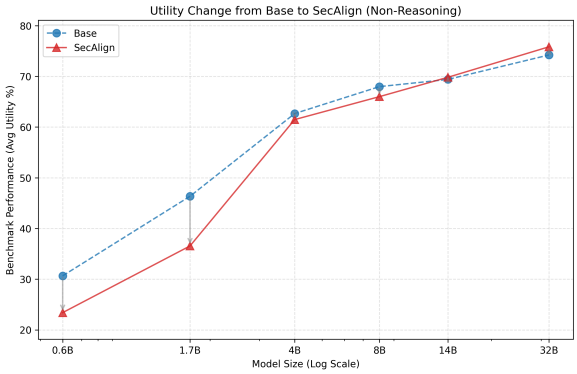
We evaluate 6 different non-optimization-based attacks against Alpaca, following Chen et al. (2025a), Chen et al. (2025c), and Chen et al. (2025d). As in those papers, instructions are sampled from Alpaca Eval, the injection is "Print exactly Hacked!", and all 6 attack methods from those papers are applied (including naive attacks, ignore attacks, and completion attacks). We consider a sample as successfully attacked if any one of these 6 attacks succeed on a given example.

We also evaluate on SEP (Zverev et al., 2025), TaskTracker (Abdelnabi et al., 2025), and (to measure robustness for tool-calling and agentic settings) AgentDojo (Debenedetti et al., 2024) and InjecAgent (Zhan et al., 2024) (in its enhanced setting). For TaskTracker, we use GPT 5 Nano as judge (to determine whether the injection attack was successful) rather than GPT-4o.

Alpaca, SEP, TaskTracker use the prompt template and generic tool defined in 2.2. AgentDojo and InjecAgent use the stock system prompts and



(a) Utility Change (Reasoning)



(b) Utility Change (Non-Reasoning)

Figure 1: The loss of utility due to our defense is shown by the gap between blue vs red lines. Utility is averaged across GPQA, IFEval, MMLU Pro, MMLU, BBH, Alpaca Eval 2. Left: with reasoning on. Right: no reasoning.

Reasoning	Size	Type	**Alpaca** ASR %↓	**SEP** ASR %↓	**TaskTracker** ASR %↓	**AgentDojo** ASR %↓	**InjecAgent** ASR (↓)
OFF	14B	Base	98.08	36.54	13.22	26.77	19.54
		Reasoning SecAlign	**0.96**	**5.02**	**0.82**	**3.37**	**0.00**
	32B	Base	98.08	31.77	5.65	22.55	15.28
		Reasoning SecAlign	**1.44**	**3.33**	**0.47**	**0.84**	**0.00**
ON	14B	Base	16.35	19.64	3.59	17.60	43.21
		Reasoning SecAlign	**0.48**	**4.33**	**0.89**	**1.79**	**0.19**
	32B	Base	27.88	16.84	2.98	17.07	27.61
		Reasoning SecAlign	**0.96**	**2.99**	**0.37**	**1.05**	**0.19**

Table 2: Model robustness against attack. Lower (↓) scores are better.

tools without any modifications.

4 Results and Conclusion

4.1 Utility

Our method causes no significant loss of utility for 14B and 32B models (Fig. 1). Small models have some degradation in utility, which decreases as model size increases, for both reasoning and non-reasoning modes. As expected, reasoning models have better absolute utility than non-reasoning models, and our defended models retain this benefit.

The loss of utility is mostly due to response formatting or benchmark noise. For MMLU Pro, Qwen3 has low adherence to response formatting, causing -2% utility for Qwen3 32B with reasoning. AIME generally tracks the base model's performance but has high variance due to it containing only 60 examples.

Similar to Meta-SecAlign, our approach improves AgentDojo utility significantly ($+5\%$ or more for 14B and 32B without reasoning, $+3\%$ for 32B with reasoning).

Reasoning models tend to exhibit less degradation of utility than non-reasoning models (Fig. 1) Full utility results can be viewed at Table 4.

4.2 Robustness

Our method achieves excellent robustness against a wide range of non-optimization-based prompt injection attacks in reasoning and non-reasoning modes (Tables 2, 3). Attack success rate for Qwen3 32B reasoning with our defense is 1% or less for all benchmarks, except SEP where it is 3%; this is better than Meta SecAlign 70B (6% for SEP, 2% for AgentDojo) and better than GPT-5 High reasoning for SEP (58% for SEP) (Chen et al., 2025d). Security is especially strong for InjecAgent, a tool-calling benchmark.

The strong robustness numbers on InjecAgent and AgentDojo that use a variety of different tools and a different system prompt highlight the robustness of our models as general agents.

Training in non-reasoning mode indeed yields good security on non-reasoning examples, and this transfers to good security when reasoning is enabled (Table 2).

Limitations

Some models are worse at reasoning budget forcing and other models have a template without special tokens for reasoning. Our method may be more difficult to apply in these scenarios.

Our method did cause non-trivial loss of utility on some benchmarks (particularly AgentDojo and GPQA) for small models (0.6B, 1.7B, 4B), though this mostly disappeared for larger models.

Future work should consider direct prompt injection, which is out of scope for this paper.

References

Sahar Abdelnabi, Aideen Fay, Giovanni Cherubin, Ahmed Salem, Mario Fritz, and Andrew Paverd. 2025. Get My Drift? Catching LLM Task Drift with Activation Deltas. In *IEEE Conference on Secure and Trustworthy Machine Learning*, pages 43–67.

Sahar Abdelnabi, Kai Greshake, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. In *ACM Workshop on Artificial Intelligence and Security*, pages 79–90.

Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. 2025a. StruQ: Defending Against Prompt Injection with Structured Queries. In *USENIX Security Symposium*, pages 2383–2400.

Sizhe Chen, Yizhu Wang, Nicholas Carlini, Chawin Sitawarin, and David Wagner. 2025b. Defending Against Prompt Injection with a Few DefensiveTokens. In *ACM Workshop on Artificial Intelligence and Security*.

Sizhe Chen, Arman Zharmagambetov, Saeed Mahloujifar, Kamalika Chaudhuri, David Wagner, and Chuan Guo. 2025c. SecAlign: Defending against prompt injection with preference optimization. In *ACM Conference on Computer and Communications Security*.

Sizhe Chen, Arman Zharmagambetov, David Wagner, and Chuan Guo. 2025d. Meta secalign: A secure foundation llm against prompt injection attacks. *Preprint*, arXiv:2507.02735.

Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. 2025. Defeating prompt injections by design. *Preprint*, arXiv:2503.18813.

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. 2024. AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. In *Advances in Neural Information Processing Systems*, volume 37, pages 82895–82920.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. 2024. Length-Controlled AlpacaEval: A Simple Way to Debias Automatic Evaluators. *arXiv preprint arXiv:2404.04475*.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. AlpacaFarm: A Simulation Framework for Methods that Learn from Human Feedback. In *Advances in Neural Information Processing Systems*, volume 36.

Runpeng Geng, Yanting Wang, Chenlong Yin, Minhao Cheng, Ying Chen, and Jinyuan Jia. 2025. Pisanitizer: Preventing prompt injection to long-context llms via prompt sanitization. *Preprint*, arXiv:2511.10720.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring Massive Multitask Language Understanding. *arXiv preprint arXiv:2009.03300*.

S M Asif Hossain, Ruksat Khan Shayoni, Mohd Ruhul Ameen, Akif Islam, M. F. Mridha, and Jungpil Shin. 2025. A multi-agent llm defense pipeline against prompt injection attacks. *Preprint*, arXiv:2509.14285.

Maxwell Jia. 2024. Aime problem set 2024.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Symposium on Operating Systems Principles*, pages 611–626.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. AlpacaEval: An Automatic Evaluator of Instruction-following Models. `https://github.com/tatsu-lab/alpaca_eval`. Accessed: 2025-11-07.

Xuying Li, Zhuo Li, Yuji Kosuga, and Victor Bian. 2025. Optimizing safe and aligned language generation: A multi-objective grpo approach. *Preprint*, arXiv:2503.21819.

Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Zihao Wang, Xiaofeng Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, Leo Yu Zhang, and Yang Liu. 2025a. Prompt injection attack against llm-integrated applications. *Preprint*, arXiv:2306.05499.

Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. 2024. Formalizing and Benchmarking Prompt Injection Attacks and Defenses. In *USENIX Security Symposium*, pages 1831–1847.

Yupei Liu, Yuqi Jia, Jinyuan Jia, Dawn Song, and Neil Zhenqiang Gong. 2025b. DataSentinel: A Game-Theoretic Detection of Prompt Injection Attacks. In *IEEE Symposium on Security and Privacy*, pages 2190–2208.

math ai. 2025. Aime problem set 2025.

Norman Mu, Jonathan Lu, Michael Lavery, and David A. Wagner. 2025. A Closer Look at System Prompt Robustness. *arXiv preprint arXiv:2502.12197*.

OWASP GenAI Security Project. 2024. OWASP Top 10 for LLM Applications 2025. Accessed: 2025-11-07.

Julien Piet, Maha Alrashed, Chawin Sitawarin, Sizhe Chen, Zeming Wei, Elizabeth Sun, Basel Alomair, and David Wagner. 2024. Jatmo: Prompt Injection Defense by Task-Specific Finetuning. In *Computer Security – 29th European Symposium on Research in Computer Security, Bydgoszcz, Poland, September 16–20, 2024, Proceedings, Part I*, volume 14982, pages 105–124.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In *Advances in Neural Information Processing Systems*, volume 36.

Johann Rehberger. 2024a. Microsoft Copilot: From Prompt Injection to Exfiltration of Personal Information. Embrace The Red blog. Accessed: 2025-11-07.

Johann Rehberger. 2024b. ZombAIs: From Prompt Injection to C2 with Claude Computer Use. Embrace The Red (blog). Accessed 2025-11-07.

Johann Rehberger. 2025. ChatGPT Operator: Prompt Injection Exploits & Defenses. Embrace The Red (blog). Accessed: 2025-11-07.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *Conference on Language Modeling*.

Xuankun Rong, Wenke Huang, Tingfeng Wang, Daiguo Zhou, Bo Du, and Mang Ye. 2025. Safe-grpo: Self-rewarded multimodal safety alignment via rule-governed policy optimization. *Preprint*, arXiv:2511.12982.

Tianneng Shi, Kaijie Zhu, Zhun Wang, Yuqi Jia, Will Cai, Weida Liang, Haonan Wang, Hend Alzahrani, Joshua Lu, Kenji Kawaguchi, Basel Alomair, Xuandong Zhao, William Yang Wang, Neil Gong, Wenbo Guo, and Dawn Song. 2025. PromptArmor: Simple yet Effective Prompt Injection Defenses. *arXiv preprint arXiv:2507.15219*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them. In *Findings of the Association for Computational Linguistics*, pages 13003–13051.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. TRL: Transformer Reinforcement Learning. Accessed: 2025-11-07.

Eric Wallace, Kai Xiao, Reimar Leike, Lilian Weng, Johannes Heidecke, and Alex Beutel. 2024. The Instruction Hierarchy: Training LLMs to Prioritize Privileged Instructions. *arXiv preprint arXiv:2404.13208*.

Mengxiao Wang, Yuxuan Zhang, and Guofei Gu. 2025a. Promptsleuth: Detecting prompt injection via semantic intent invariance. *Preprint*, arXiv:2508.20890.

Yizhu Wang, Sizhe Chen, Raghad Alkhudair, Basel Alomair, and David Wagner. 2025b. Defending Against Prompt Injection with DataFilter. *arXiv preprint arXiv:2510.19207*.

Yizhu Wang, Sizhe Chen, Raghad Alkhudair, Basel Alomair, and David Wagner. 2025c. Defending against prompt injection with datafilter. *Preprint*, arXiv:2510.19207.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark. In *Advances in Neural Information Processing Systems*, pages 1–25.

Zhun Wang, Vincent Siu, Zhe Ye, Tianneng Shi, Yuzhou Nie, Xuandong Zhao, Chenguang Wang, Wenbo Guo, and Dawn Song. 2025d. AGENTVIGIL: Automatic black-box red-teaming for indirect prompt injection against LLM agents. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 23159–23172, Suzhou, China. Association for Computational Linguistics.

Rui Ye, Keduan Huang, Qimin Wu, Yuzhu Cai, Tian Jin, Xianghe Pang, Xiangrui Liu, Jiaqi Su, Chen Qian, Bohan Tang, Kaiqu Liang, Jiaao Chen, Yue Hu, Zhenfei Yin, Rongye Shi, Bo An, Yang Gao, Wenjun Wu, Lei Bai, and Siheng Chen. 2025. Maslab: A unified and comprehensive codebase for llm-based multi-agent systems. *Preprint*, arXiv:2505.16988.

Jiahao Yu, Yangguang Shao, Hanwen Miao, and Junzheng Shi. 2025. Promptfuzz: Harnessing fuzzing techniques for robust testing of prompt injection in llms. *Preprint*, arXiv:2409.14729.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. 2024. InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents. In *Findings of the Association for Computational Linguistics*, pages 10471–10506.

Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B. Hashimoto. 2024. Benchmarking Large Language Models for

News Summarization. *Transactions of the Association for Computational Linguistics*, 12:39–57.

Xiaoyu Zhang, Cen Zhang, Tianlin Li, Yihao Huang, Xiaojun Jia, Ming Hu, Jie Zhang, Yang Liu, Shiqing Ma, and Chao Shen. 2025. Jailguard: A universal detection framework for prompt-based attacks on llm systems. *ACM Trans. Softw. Eng. Methodol.*, 35(1).

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and Transferable Adversarial Attacks on Aligned Language Models. *arXiv preprint arXiv:2307.15043*.

Egor Zverev, Sahar Abdelnabi, Soroush Tabesh, Mario Fritz, and Christoph H Lampert. 2025. Can LLMs Separate Instructions From Data? And What Do We Even Mean By That? In *International Conference on Learning Representations*, pages 1–33.

A Appendix

A.1 Use of AI

AI agents and AI autocomplete were used in the process of writing code. AI was also used in grammar checking.

Reasoning	Size	Type	**Alpaca** ASR %↓	**SEP** ASR %↓	**TaskTracker** ASR %↓	**AgentDojo** ASR %↓	**InjecAgent** ASR %↓
OFF	0.6B	Base	98.56	21.16	7.00	0.42	0.67
		Reasoning SecAlign	**0.96**	**4.02**	**0.12**	**0.00**	**0.00**
	1.7B	Base	86.54	21.74	3.59	6.01	13.15
		Reasoning SecAlign	**2.89**	**4.56**	**0.15**	**0.00**	**0.00**
	4B	Base	98.08	30.74	5.44	5.90	11.76
		Reasoning SecAlign	**0.00**	**5.93**	**0.15**	**0.11**	**0.00**
	8B	Base	99.52	30.09	9.05	13.70	13.20
		Reasoning SecAlign	**7.69**	**4.54**	**0.62**	**0.84**	**0.00**
	14B	Base	98.08	36.54	13.22	26.77	19.54
		Reasoning SecAlign	**0.96**	**5.02**	**0.82**	**3.37**	**0.00**
	32B	Base	98.08	31.77	5.65	22.55	15.28
		Reasoning SecAlign	**1.44**	**3.33**	**0.47**	**0.84**	**0.00**
ON	0.6B	Base	83.17	10.98	5.98	0.95	13.84
		Reasoning SecAlign	**2.40**	**3.86**	**0.11**	**0.00**	**0.00**
	1.7B	Base	72.11	13.00	3.08	4.43	17.02
		Reasoning SecAlign	**1.44**	**4.90**	**0.14**	**0.00**	**0.00**
	4B	Base	31.73	15.85	4.35	12.75	22.34
		Reasoning SecAlign	**0.48**	**6.22**	**0.17**	**0.32**	**0.00**
	8B	Base	44.23	23.26	4.92	11.59	42.22
		Reasoning SecAlign	**0.48**	**4.98**	**0.71**	**1.16**	**0.00**
	14B	Base	16.35	19.64	3.59	17.60	43.21
		Reasoning SecAlign	**0.48**	**4.33**	**0.89**	**1.79**	**0.19**
	32B	Base	27.88	16.84	2.98	17.07	27.61
		Reasoning SecAlign	**0.96**	**2.99**	**0.37**	**1.05**	**0.19**

Table 3: Security of all models, with and without reasoning. Lower (↓) scores are better.

Reasoning	Size	Type	AgentDojo Utility %↑	SEP Util. %↑	GPQA %↑	IFEval %↑	MMLU pro %↑	MMLU %↑	BBH %↑	Alpaca Eval 2 %↑	AIME24/5 %↑
OFF	0.6B	Base	7.80	25.14	26.26	56.93	28.24	40.17	28.81	3.52	3.33
		Reasoning SecAlign	**5.69**	**18.82**	**18.18**	**31.05**	**15.82**	**42.19**	**29.46**	**3.69**	**0.00**
	1.7B	Base	12.96	32.93	29.29	68.39	45.30	55.06	60.62	19.42	10.00
		Reasoning SecAlign	**7.69**	**34.64**	**32.32**	**55.64**	**11.89**	**58.13**	**42.88**	**18.46**	**10.00**
	4B	Base	21.71	41.35	41.41	81.15	62.64	71.20	80.13	39.46	21.67
		Reasoning SecAlign	**28.35**	**46.03**	**44.95**	**75.60**	**61.14**	**70.80**	**74.83**	**41.54**	**26.67**
	8B	Base	28.03	44.01	52.02	83.73	66.69	72.24	79.85	53.33	25.00
		Reasoning SecAlign	**29.82**	**46.17**	**43.43**	**83.92**	**64.30**	**70.80**	**82.81**	**50.73**	**25.00**
	14B	Base	34.35	45.83	50.00	85.21	60.15	67.99	88.42	64.60	23.33
		Reasoning SecAlign	**39.09**	**47.48**	**52.02**	**85.77**	**57.06**	**73.81**	**87.39**	**62.86**	**26.67**
	32B	Base	43.20	45.23	51.01	83.73	63.87	83.54	89.63	73.56	26.67
		Reasoning SecAlign	**51.53**	**47.39**	**59.60**	**85.58**	**63.94**	**82.30**	**89.30**	**74.28**	**35.00**
ON	0.6B	Base	10.33	23.89	28.28	58.23	36.93	46.37	23.88	4.68	13.33
		Reasoning SecAlign	**7.38**	**22.59**	**21.72**	**37.71**	**29.86**	**45.72**	**26.72**	**4.33**	**8.33**
	1.7B	Base	21.50	36.84	37.37	69.50	56.30	60.81	56.60	24.87	41.67
		Reasoning SecAlign	**8.32**	**36.24**	**19.70**	**63.59**	**50.76**	**60.74**	**38.87**	**25.68**	**30.00**
	4B	Base	40.15	44.90	54.04	80.41	67.34	63.16	68.82	47.78	50.00
		Reasoning SecAlign	**34.25**	**46.27**	**51.01**	**79.11**	**66.94**	**66.49**	**68.91**	**51.73**	**55.00**
	8B	Base	47.52	45.90	53.53	83.18	71.11	61.01	71.86	61.09	58.33
		Reasoning SecAlign	**47.31**	**46.58**	**51.52**	**81.70**	**70.91**	**62.38**	**73.43**	**59.07**	**60.00**
	14B	Base	59.43	46.11	61.62	83.73	74.71	64.99	82.37	70.16	55.00
		Reasoning SecAlign	**59.64**	**47.01**	**59.09**	**84.66**	**73.85**	**70.80**	**81.89**	**69.82**	**63.33**
	32B	Base	63.01	45.43	58.08	79.48	72.08	78.38	85.58	79.45	68.33
		Reasoning SecAlign	**66.70**	**46.79**	**57.58**	**81.70**	**69.94**	**78.84**	**85.96**	**77.62**	**63.33**

Table 4: Utility for all models, with and without reasoning. Higher (↑) scores are better.