CASCADE Against Jailbreaks: Combination Across Stages with Controlled Attack-Defense Evaluation

Jiale Luo
School of Computing
National University of Singapore
luojiale@u.nus.edu

Eric Han
School of Computing
National University of Singapore
eric_han@nus.edu.sg

Abstract

Defenses against jailbreak attacks on Large Language Models (LLMs) operate at different pipeline stages, such as input modification or output guard, but it remains unclear which defenses to deploy at each stage and how to combine them. Prior empirical studies, fragmented by inconsistent attack-success-rate definitions and experimental settings, have evaluated defenses largely in isolation. Here we present the first systematic study, to our knowledge, of defense combinations both within and across pipeline stages, under a consistent threat model of direct, black-box, single-turn attacks. Our decision framework standardizes evaluation through a principled attack-success-rate formulation with controlled query budgets, together with explicit fairness rules. Across 19 attacks and 15 defenses, we find that no single defense is universally best, but well-chosen combinations achieve substantial safety with minimal utility degradation, yielding practical recommendations for layered defense pipelines.

1 Introduction

Large Language Models (LLMs) such as GPT, Claude, and LLaMA underpin applications ranging from chatbots and code generation to automated scoring of student responses (OpenAI et al., 2024; Lee et al., 2024). Advances such as ReAct-style tool use (Yao et al., 2023) and personalized AI assistants (Steinberger and the OpenClaw community, 2026) extend LLM autonomy to external actions, including code execution, shell commands, and other consequential operations.

This growing autonomy expands the attack surface. OWASP (2024) identifies prompt injection as the top threat, embedding adversarial input in user prompts to manipulate LLMs (Perez and Ribeiro, 2022). Jailbreak, a prevalent form of prompt injection (OWASP, 2024), explicitly aims to circumvent safety measures, with consequences ranging from harmful generations (e.g., self-harm guidance) to unauthorized command execution (Vassilev et al., 2025).

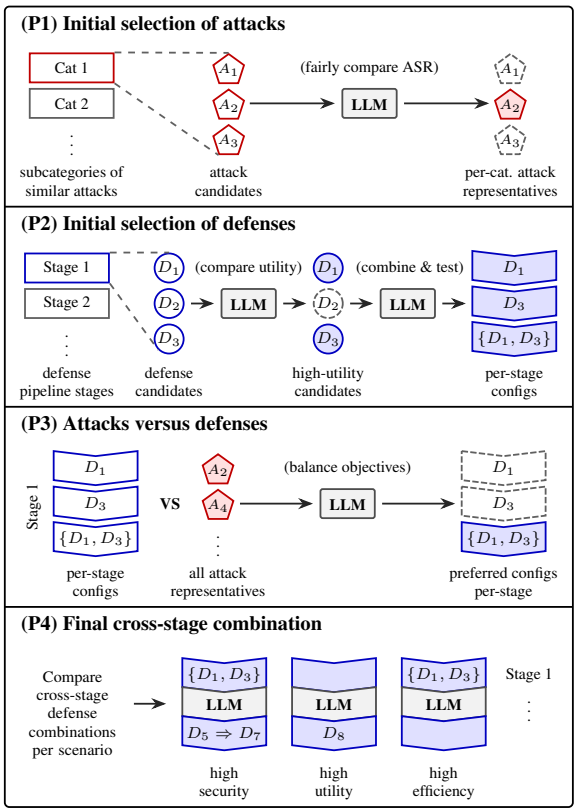


Figure 1: CASCADE's four sequential processes (P1–P4) select representative attacks (red pentagons), test them against per-stage defense configurations (blue chevrons) built from individual defenses (blue circles), and identify preferred cross-stage defense combinations for practical scenarios.

Jailbreak is considered an inherent LLM vulnerability, rooted in the tension between competing training objectives of *helpfulness* and *harmlessness*. More capable models can also follow malicious instructions more effectively, so safety must scale with capability (Wei et al., 2023). Despite safety alignment efforts (Touvron et al., 2023; OpenAI et al., 2024), jailbreak risks persist as attackers continually evolve their strategies (Nasr et al., 2025).

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 20866–20888
October 24-29, 2026 ©2026 Association for Computational Linguistics

These risks are *no longer theoretical*: a user's suicide has prompted litigation over alleged safety failures in a deployed LLM system (Raine v. OpenAI, Inc., 2025). Substantial research has emerged on jailbreak attacks and defenses, yet three persistent gaps remain.

(1) Fair comparison is hindered by widely varying experimental settings across studies (Chu et al., 2025). Notably, the absence of unified metrics limits fair comparison across heterogeneous techniques (Chouldechova et al., 2025), a gap that persists in standardized evaluation benchmarks (Mazeika et al., 2024; Chao et al., 2024) and empirical studies (Xu et al., 2024b; Shen et al., 2025).

(2) Practical defense requires understanding which defenses work best under specific conditions and navigating trade-offs in security, efficiency, and utility (Shen et al., 2025; Wang et al., 2026). Despite broad coverage, these studies offer high-level insights rather than concrete deployment recommendations, and none examine how defenses should be combined.

(3) Defense combinations, both within and across pipeline stages, have not been systematically studied. Though no single defense is universally effective (Shen et al., 2025; Chu et al., 2025), defenses are evaluated only in isolation, leaving effective combinations a key open problem.

Hence, we propose CASCADE, to our knowledge, the first framework for fair, controlled, systematic evaluation of defense combinations across pipeline stages, toward practical defense recommendations. Our contributions are as follows:

- **Fair comparison framework.** We propose a decision framework with well-defined metrics, standardized experiment settings, and explicit fairness rules for the controlled evaluation of jailbreak attacks and defenses.
- **Practical defense combinations.** We systematically evaluate defense combinations within and across pipeline stages, identifying those that achieve substantial security with minimal utility or efficiency degradation.
- **Empirical evaluation.** We evaluate 19 unique attacks and 15 unique defenses under our framework, providing actionable recommendations for practical adoption and open-sourcing our code[1] to support further research.

[1] https://github.com/Singa-pirate/CASCADE-jailbreak-defense-combi

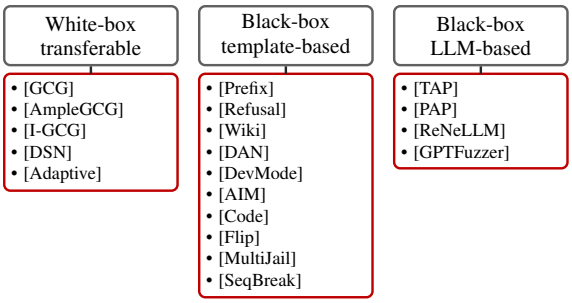


Figure 2: Three jailbreak attack categories (in-scope) and 19 unique techniques evaluated; details in Table 5.

2 Background and Related Work

Contemporary jailbreaks span diverse attack surfaces, including multi-turn conversations (Russinovich et al., 2025) and LLM-based agents (Andriushchenko et al., 2025b). We focus on the foundational primitive of *direct*, *black-box*, *single-turn* jailbreaks, where a single adversarial prompt attempts to bypass the safety measures of a black-box LLM. Techniques applied in this restrictive setting remain applicable in less restrictive ones, making it the most actionable regime for practical defense recommendations.

In *direct* attacks, the user is the attacker, submitting the malicious prompt themselves rather than via injected external content (Vassilev et al., 2025). In the *black-box* setting, the attacker has only query access to the target LLM, observing its text output without visibility into model internals such as weights, gradients, or logits (Yi et al., 2024). *Single-turn* attacks succeed in a single query, without exploiting conversation history or multi-turn escalation (Russinovich et al., 2025).

2.1 Jailbreak Attacks

Attacks commonly divide into white-box and black-box, the latter including template-based and LLM-based approaches (Yi et al., 2024). White-box techniques require model internals beyond our threat model, but we include the *transferable* prompts they produce, which are applicable to the black-box setting. Figure 2 summarizes the scope, with each category detailed below.

(1) White-box transferable attacks involve a surrogate LLM's gradients to optimize for jailbreak success, generating prompts to attack the target LLM. For example, GCG refines a universal suffix to elicit affirmative responses (Zou et al., 2023).

(2) Black-box template-based attacks use templates: output style constraints (Wei et al., 2023), fictional

personas (Shen et al., 2024), or structured formats (Liu et al., 2025b; Saiem et al., 2025). For example, CodeAttack disguises the malicious prompt as a code-completion task (Ren et al., 2024).
(3) Black-box LLM-based attacks automate prompt generation via red-team LLMs (i.e., LLMs acting as adversaries). For example, TAP uses attacker and evaluator LLMs to refine prompts in a branch-evaluate-prune cycle (Mehrotra et al., 2024).

2.2 Jailbreak Defenses

Adapting the stage-based taxonomies of prior work (Chen et al., 2025; Wang et al., 2026), we organize defenses (Figure 3), focusing on the three stages easiest to apply in practice; stages detailed below.
(1) Training includes data cleaning and post-training safety alignment (Wang et al., 2025).
(2) Input guard screens and blocks harmful inputs, using signals such as perplexity (Jain et al., 2023) or guard models such as WildGuard (Han et al., 2024) and PromptGuard (Meta, 2025).
(3) Input modification transforms the user prompt to neutralize harmful intent. Examples range from safety system prompts (Xie et al., 2023) to optimized defense strings such as RPO (Zhou et al., 2024) and DPP (Xiong et al., 2025).
(4) Inference intervenes within the model at inference time, for example by inspecting safety-critical parameter gradients or steering decoding (Xie et al., 2024; Xu et al., 2024a).
(5) Output refinement post-processes the output, often using secondary LLMs to inspect, revise, or reject the final response (Wang et al., 2024).
(6) Output guard screens query-response pairs and blocks harmful conversations, using guard models such as LlamaGuard (Inan et al., 2023) or reasoning models such as GuardReasoner (Liu et al., 2025a). Some input guards such as WildGuard also apply here.

2.3 Evaluation Methods and Metrics

Jailbreak effectiveness is commonly evaluated on a malicious-goal dataset, with a judge scoring jailbreak attempts. Generally, *Attack Success Rate (ASR)* describes the proportion of jailbroken goals:

$$ASR = \frac{\sum_d I(Q_d)}{|D|}, \quad (1)$$

where Q_d is the attempt for the d-th malicious goal in dataset D, and the indicator I, scored by a judge, returns 1 for a successful jailbreak and 0 otherwise. Higher ASR indicates a more effective attack or a less effective defense, and vice versa. ASR is widely used, but varying instantiations could cause unfair comparisons (Chouldechova et al., 2025):
(1) ASR_{avg} averages success over N attempts per malicious goal (Zou et al., 2023), where $n \in \{1, \ldots, N\}$ indexes attempts:

$$ASR_{avg} = \frac{\sum_n \sum_d I(Q_{d,n})}{|D| * N}. \quad (2)$$

(2) $ASR_{@N}$, or top-N ASR, considers a goal to be jailbroken if any of the N attack attempts succeeds (Zhou et al., 2025):

$$ASR_{@N} = \frac{\sum_d \max_n [I(Q_{d,n})]}{|D|}. \quad (3)$$

(3) $ASR_{ensemble}$ treats a goal as jailbroken if any of V attack variants succeeds (Ding et al., 2024), where $v \in \{1, \ldots, V\}$ indexes variants:

$$ASR_{ens} = \frac{\sum_d \max_v [I(Q_{d,v})]}{|D|}. \quad (4)$$

Beyond instantiation choice, ASR accuracy depends on the judge that scores each attempt. Early rule-based judges, such as keyword filters, produce frequent misjudgments. Recent LLM-based judges include GPT-4 evaluator (Qi et al., 2024), HarmBench judge (Mazeika et al., 2024), ShieldLM

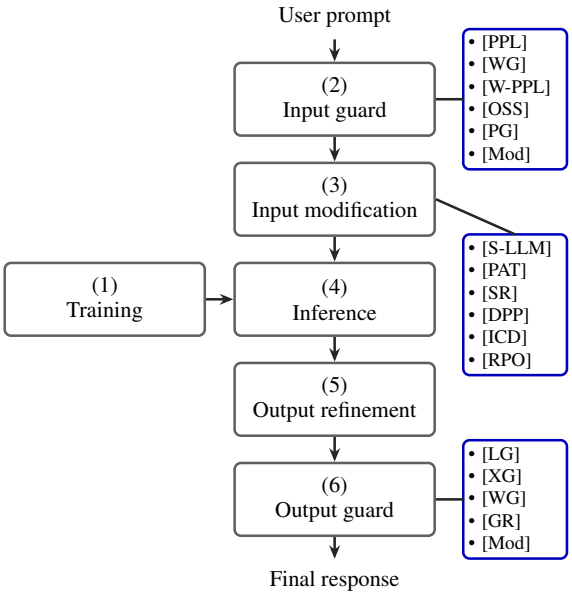


Figure 3: The six-stage LLM defense pipeline, with 15 unique contemporary defense techniques shown for the three in-scope stages (input guard, input modification, output guard); WG and Mod serve dual roles in stages (2) and (6); details in Table 6.

(Zhang et al., 2024), and fine-tuned RoBERTa models (Xu et al., 2024b), but still produce false positives and negatives (Shen et al., 2025; Chouldechova et al., 2025). Despite proposals such as majority voting over an ensemble of judges (Zhou et al., 2025), no strategy clearly aligns best with human judgment.

ASR also depends on the dataset D of malicious goals, which spans major AI safety policy categories. AdvBench is an early benchmark (Zou et al., 2023), though later work notes duplicated entries (Chao et al., 2024). Recent alternatives include HarmBench (Mazeika et al., 2024) and JBB-Behaviors (Chao et al., 2024).

LLM response quality is also important, as defenses can degrade responses due to over-refusal. *Utility* metrics measure a defended LLM's ability to answer normal queries. A common utility framework, AlpacaEval (Dubois et al., 2024), reports the target LLM's length-controlled win-rate against a reference model under LLM-based judging. Other datasets, including XSTest (Röttger et al., 2024) and OR-Bench (Cui et al., 2025), further benchmark over-refusal on benign prompts near the safety boundary.

2.4 Empirical Studies

Several empirical studies have evaluated jailbreak attacks and defenses. Xu et al. (2024b) provide an early evaluation of 9 attacks and 7 defenses. PandaGuard evaluates 19 attacks against 9 defenses (out of 12 implemented), finding no universally best defense (Shen et al., 2025). The Security-Efficiency-Utility framework reaches the same conclusion across 9 attacks and 9 guardrail defenses (Wang et al., 2026). AISafetyLab (Zhang et al., 2025) and TeleAI-Safety (Chen et al., 2025) further expand coverage. While these studies consistently find no universally optimal defense, systematic evaluation of defense combinations remains unexplored. Our work addresses the gap with the first systematic study, to our knowledge, of defense combinations within and across pipeline stages, yielding practical recommendations for layered defense pipelines.

3 Methodology

Our CASCADE framework consists of four sequential processes that progressively refine the technique set, illustrated in Figure 1. The rules below define one fair selection procedure that practitioners can adapt for different use cases; rationales for each rule are given in Appendix B.

(P1) Initial selection of attacks. We group attacks into *subcategories* based on similarity. Within each subcategory, we evaluate ASR on target LLMs and select effective attacks to represent the group. This grouping improves experimental efficiency while ensuring coverage of diverse attack patterns. To ensure fairness, we assume the **same attacker capabilities** for all attacks (per the threat model) and apply the **same target LLM query budget per subcategory** (reflecting practical query cost and detection risk).

(P2) Initial selection of defenses. We group defenses by pipeline stage. Within each *stage*, we evaluate the utility of individual defenses and intra-stage defense combinations, eliminating those that cause unacceptable utility degradation, via **top-k rank utility selection** within reasonable groupings of defense configurations.

(P3) Attacks versus defenses. For each pipeline stage, we test selected defenses against the attack representatives to evaluate ASR reduction, selecting preferred defense configurations within the search space. We give attacks the **same query budgets as before** (simulating effective attack scenarios) and select based on **balanced performance** across ASR reduction, utility preservation, and memory use.

(P4) Final cross-stage combination. We evaluate cross-stage combinations of the preferred defenses using the same **top-k rank utility selection** and **ASR evaluation**, identifying recommended defense combinations for practical scenarios.

3.1 Metrics

To address the unfair comparisons caused by varying ASR definitions (Section 2.3; Chouldechova et al., 2025), we adopt a unified formulation applicable across techniques.

Jailbreak effectiveness is measured by $ASR_{@max_q}$, combining $ASR_{@N}$ and ASR_{ens} for fair application across techniques. We fix a maximum query budget per subcategory, so attacks with more variants receive fewer repetitions:

$$ASR_{@max_q} = \frac{\sum_d \max_{n,v}[I(Q_{d,n,v})]}{|D|}, \quad (5)$$

where $Q_{d,n,v}$ denotes the n-th attempt of the v-th variant for the d-th malicious goal. $ASR_{@max_q}$ captures jailbreak success within the budget.

LLM response quality is measured by AlpacaEval's length-controlled win-rate:

$$\begin{aligned} Utility &:= \mathrm{WinRate}^{\mathrm{LC}}(m, b) \\ &= 100 \cdot \mathbb{E}_x \left[\sigma\big(Q_{m,b} + \underbrace{0}_{\text{length term}} + D_{m,b,x}\big) \right], \end{aligned} \quad (6)$$

where m is the target model, b the baseline reference, x a normal instruction, and σ a logistic regression over model quality Q, length (set to zero), and instruction difficulty D.

Defense effectiveness is measured by two aggregates that we seek to jointly maximize, subject to an *inherent tradeoff* between them. $\overline{\%{\downarrow}ASR}$ averages the percentage reduction of $ASR_{@max_q}$ across attack representatives:

$$\overline{\%{\downarrow}ASR} = \frac{1}{|A|} \sum_a \frac{ASR_a - ASR'_a}{ASR_a} \quad (7)$$

where a indexes representative attacks in A, and ASR_a, ASR'_a are its $ASR_{@max_q}$ before and after the defense. Symmetrically, $\overline{\%{\uparrow}U}$ is the average percentage increase of utility across target LLMs:

$$\overline{\%{\uparrow}U} = \frac{1}{|M|} \sum_m \frac{U'_m - U_m}{U_m} \quad (8)$$

where m indexes target LLMs in M, and U_m, U'_m are its utility before and after the defense.

3.2 Implementation and Experimental Setup

Our pipeline-based architecture (Figure 4) constructs an execution graph of attack, defense, and target LLM nodes; each node updates a shared state object that records the prompt, response, rejection status, and evaluation result. We summarize three key settings below, with full details in Appendix A.3.

(1) Datasets and judges. For jailbreak evaluation, we use the JBB-Behaviors dataset (100 malicious goals; adopted by Zhou et al., 2024; Robey et al., 2025) paired with the HarmBench judge (as in Zhou et al., 2025), balancing quality, efficiency, and prior adoption. For utility, we use the AlpacaEval framework with Llama-3.1-70B-Instruct as the LLM judge, following a recommended AlpacaEval setting for strong human agreement and efficiency.

(2) Target LLMs. We select target LLMs per experiment, covering nine open-weight and proprietary models in total (Table 7). We disable reasoning for target LLMs to isolate base safety behavior under identical generation conditions.

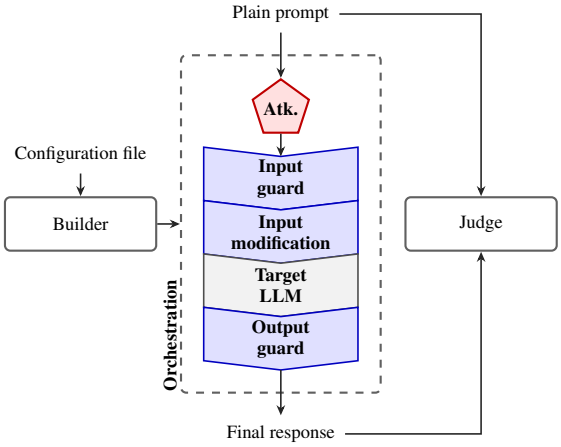


Figure 4: Pipeline-based system architecture: the Builder (left) wires the Orchestration framework (center) from a configuration file, and the Judge (right) scores the final response.

(3) LLM settings. We set the temperature to 0 with a constant seed for all judges, and for LLM generations in utility evaluations; 1 for target LLMs in repeated jailbreak attempts. We set max_new_tokens to 2048 (non-reasoning models) and 4096 (reasoning models) to avoid unexpected output truncations.

4 Results

4.1 Initial Attack Selection Results

In the first process, we evaluate attacks on six LLMs commonly used in prior work: three open-weight (Vicuna, Llama2, Llama3) and three proprietary models (GPT-3.5, GPT-4o, Claude-sonnet-4). For each subcategory, we set a query budget divisible by each attack's variant count, while ensuring sufficient repetitions per variant. From the results (Table 1), we select a minimal set of effective attacks per subcategory. Among white-box transferable attacks, we select both **Adaptive** and **DSN** for their strong overall performance. Persona-pattern attacks under-perform the baseline on advanced LLMs (GPT-4o and Claude-sonnet-4), likely due to modern safety training against fixed personas; we therefore exclude this subcategory. This yields five attack representatives: **Adaptive**, **DSN**, **Refusal**, **SeqBreak**, and **ReNeLLM**.

4.2 Initial Defense Selection Results

In the second process, we evaluate the utility of defended LLMs, using newer LLMs popular in practice at the time of writing (informed by sources such as OpenRouter rankings (OpenRouter, 2026)): Llama3, Gemini-2.5-Flash, and GPT-4o. For in-

Subcategory	Query Budget	Attack	Target LLM						Average
			Vicuna	Llama2	Llama3	GPT-3.5	GPT-4o	Claude-sonnet-4	
White-box transferable	15	**Adaptive**	**100.00**	2.00	**100.00**	**100.00**	9.00	**44.00**	**59.17**
		DSN	93.00	**97.00**	30.00	94.00	13.00	2.00	54.83
		GCG	**100.00**	42.00	40.00	99.00	15.00	2.00	49.67
		AmpleGCG	98.00	19.00	26.00	98.00	**16.00**	5.00	43.67
		I-GCG	91.00	3.00	8.00	74.00	8.00	2.00	31.00
		Baseline	83.00	7.00	22.00	56.00	7.00	5.00	30.00
Black-box template-based, output style pattern	5	**Refusal**	94.00	**19.00**	**46.00**	**92.00**	**38.00**	**8.00**	**49.50**
		Prefix	**99.00**	7.00	10.00	87.00	6.00	4.00	35.50
		Wiki	80.00	9.00	7.00	62.00	5.00	0.00	27.17
		Baseline	58.00	4.00	16.00	46.00	7.00	5.00	22.67
Black-box template-based, persona pattern	5	DevMode	99.00	**9.00**	**59.00**	**64.00**	0.00	0.00	**38.50**
		DAN	95.00	8.00	57.00	33.00	0.00	1.00	32.33
		AIM	**100.00**	5.00	33.00	2.00	0.00	0.00	23.33
		Baseline	58.00	4.00	16.00	46.00	**7.00**	**5.00**	22.67
Black-box template-based, disguise pattern	36	**SeqBreak**	**100.00**	97.00	**100.00**	**100.00**	**98.00**	**60.00**	**92.50**
		Code	99.00	**100.00**	**100.00**	**100.00**	97.00	24.00	86.67
		Flip	98.00	58.00	81.00	**100.00**	80.00	0.00	69.50
		MultiJail	**100.00**	71.00	88.00	95.00	0.00	0.00	59.00
		Baseline	94.00	9.00	26.00	63.00	8.00	6.00	34.33
Black-box LLM-based	80	**ReNeLLM**	91.00	**81.00**	91.00	**95.00**	**87.00**	12.00	**76.17**
		PAP	70.00	42.00	**94.00**	59.00	41.00	3.00	51.50
		GPTFuzzer	73.00	39.00	79.00	69.00	34.00	1.00	49.17
		TAP	75.00	34.00	33.00	78.00	36.00	**17.00**	45.50
		Baseline	**98.00**	9.00	27.00	67.00	11.00	9.00	36.83

Table 1: (P1) $ASR_{@max_q}$ results (%) across attacks and target LLMs. (Baseline: plain malicious strings; bold values: subcategory maximum; bold attack names: selected.)

put/output guards, {.} denotes combinations in which any included guard can trigger rejection; for input modification, ⇒ denotes sequential application of modification procedures. Per stage, we group configurations based on number of defenses applied and retain the top $k = 4$ by $\overline{\%\uparrow U}$ rank per group (Figures 5, 6, 7).

4.3 Attacks Versus Defenses Results

In the third process, we evaluate selected defenses against the five representative attacks on Vicuna and GPT-3.5, chosen for their higher baseline ASR (giving a wider, more discriminative range for defense effectiveness) and lower cost at our experimental scale. These defenses also apply effectively to modern LLMs (GPT-5.4-mini and gemini-3.1-flash-lite), as shown in Section 5. To navigate security-utility-efficiency trade-offs, practitioners can compare metrics such as $\overline{\%\downarrow ASR}$, $\overline{\%\uparrow U}$, and defense model size; Table 2 shows an example per-stage selection, choosing the preferred configurations for security, utility, or memory priorities at each stage.

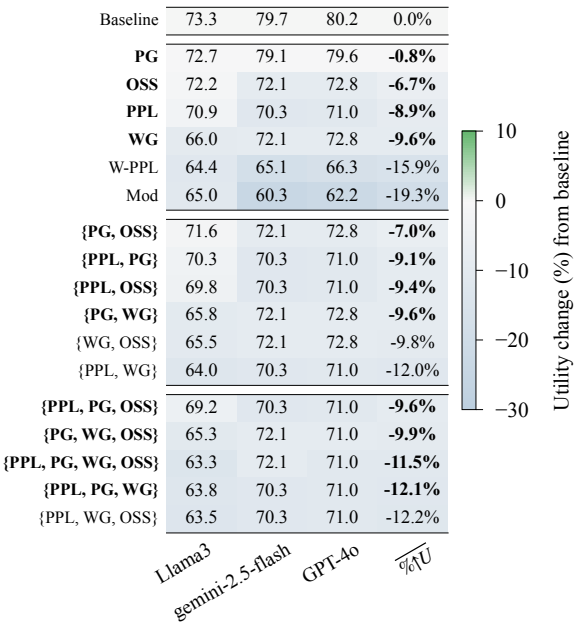


Figure 5: (P2) Utility across *input guard* configurations. (Baseline: no defense; bold: selected.)

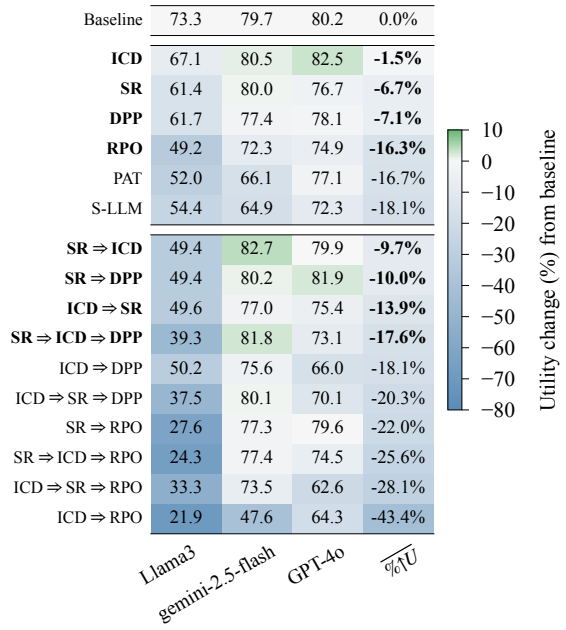


Figure 6: (P2) Utility across *input modification* configurations. (Baseline: no defense; bold: selected.) DPP and RPO both add an optimized defense suffix, so we exclude combinations containing both.

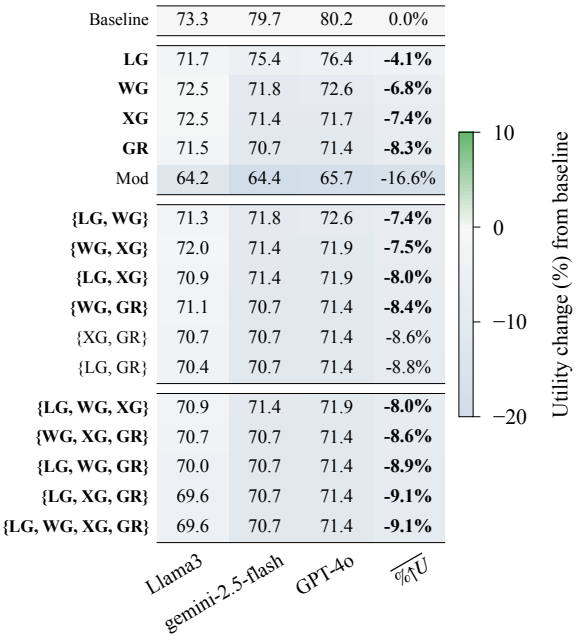


Figure 7: (P2) Utility across *output guard* configurations. (Baseline: no defense; bold: selected.)

4.4 Final Cross-Stage Combination Results

In the last process, we evaluate cross-stage defense combinations of the selected per-stage defenses. CASCADE provides a selection procedure that balances performance across metrics to navigate the security-utility-efficiency trade-off.

Pipeline stage	**Defense configuration** (model parameter count)	$\overline{\%\downarrow ASR}$	$\overline{\%\uparrow U}$
Input guard	PG (86M)	54.0	-0.8
	OSS (20B)	70.1	-6.7
	PPL (7B)	19.2	-8.9
	WG (7B)	79.9	-9.6
	{**PG**, **OSS**} · · · · · · · **util1**	82.3	-7.0
	{PPL, PG}	69.6	-9.1
	{PPL, OSS}	76.4	-9.4
	{**PG**, **WG**} · · · · · · **mem1**	86.7	-9.6
	{PPL, PG, OSS}	87.2	-9.6
	{PG, WG, OSS} · · · · **sec1**	94.2	-9.9
	{PPL, PG, WG, OSS}	95.4	-11.5
	{PPL, PG, WG}	90.2	-12.1
Input modification	ICD (0B)	12.1	-1.5
	SR (0B) · · · · · · · · · **util2**	18.1	-6.7
	DPP (0B)	2.1	-7.1
	RPO (0B)	2.2	-16.3
	SR ⇒ ICD	21.9	-9.7
	SR ⇒ DPP	19.8	-10.0
	ICD ⇒ SR · · · · · · · **sec2**	30.5	-13.9
	SR ⇒ ICD ⇒ DPP	27.6	-17.6
Output guard	LG (8B)	67.0	-4.1
	WG (7B)	70.8	-6.8
	XG (8B)	15.8	-7.4
	GR (8B)	67.2	-8.3
	{**LG**, **WG**} · **util3** / **mem3**	83.7	-7.4
	{WG, XG}	74.6	-7.5
	{LG, XG}	70.5	-8.0
	{WG, GR}	81.4	-8.4
	{LG, WG, XG}	84.6	-8.0
	{WG, XG, GR}	83.2	-8.6
	{**LG**, **WG**, **GR**} · · · · **sec3**	86.7	-8.9
	{LG, XG, GR}	79.0	-9.1
	{LG, WG, XG, GR}	87.6	-9.1

Table 2: (P3) Per-stage defense results ($\overline{\%\downarrow ASR}$ measured on Vicuna and GPT-3.5; $\overline{\%\uparrow U}$ measured on Llama3, gemini-2.5-flash and GPT-4o). Bold names with colored labels mark our recommended configurations for each scenario, by strongest aspect (sec = security; util = utility; mem = memory) and stage (1 = *input guard*; 2 = *input modification*; 3 = *output guard*).

Selection metrics let practitioners balance security, utility, and efficiency according to their priorities. For security and utility, $\overline{\%\downarrow ASR}$ and $\overline{\%\uparrow U}$ from previous processes provide direct metrics. For efficiency, we consider two metrics: memory use, estimated by *total parameter count* $|\Theta| = \sum_i \theta_i$ (where θ_i is each defense component's model size); and *extra delay* $\Delta T = T' - T$, the difference in average response times with and without defenses, following Wang et al. (2026). Practitioners can then filter combinations by these constraints and select by their trade-off priority.

Scenario	Cross-stage defense combination	$\overline{\%\downarrow ASR}$	$\overline{\%\uparrow U}$	$\lvert\Theta\rvert$(B)	ΔT(s)
Security	**util1** ⇒ **util2** ⇒ **sec3** ({PG, OSS} ⇒ [SR] ⇒ {LG, WG, GR})	97.4	-9.2	43.1	+2.16
Utility	**util1** ⇒ **util3** ({PG, OSS} ⇒ {LG, WG})	94.6	-7.8	35.1	+0.48
Efficiency	**mem1** ({PG, WG})	86.7	-9.6	7.1	-0.02

Table 3: (P4) Recommended cross-stage defense combinations for three representative trade-off scenarios ($\overline{\%\downarrow ASR}$ measured on Vicuna and GPT-3.5; $\overline{\%\uparrow U}$ measured on Llama3, gemini-2.5-flash and GPT-4o).

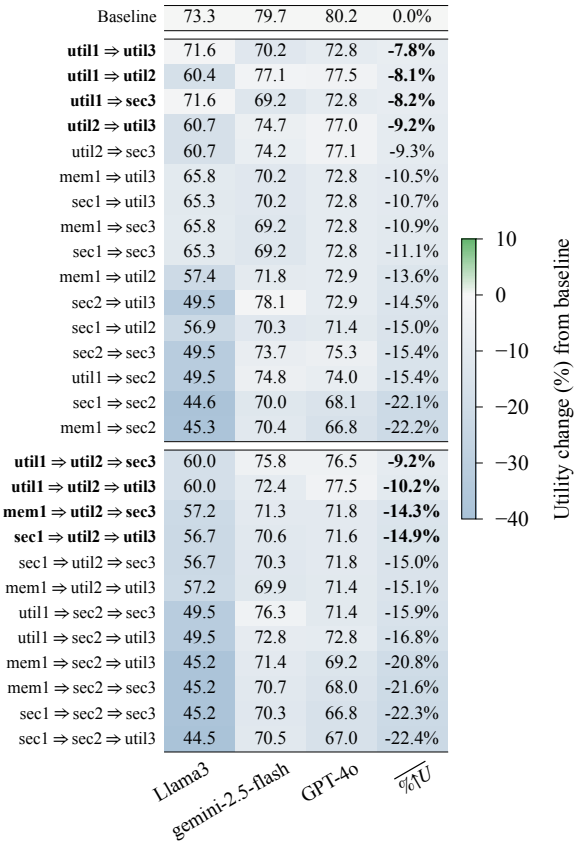


Figure 8: (P4) Utility across cross-stage defense combinations. (Baseline: no defense; bold: selected.)

Cross-stage defense combination selection first applies top-k by $\overline{\%\uparrow U}$ rank within each combination-size group (Figure 8), then evaluates them on $\overline{\%\downarrow ASR}$ against representative attacks, and measures ΔT on benchmark hardware. In our case, $k = 4$ within 2-stage and 3-stage combinations, the five P1 attack representatives, and Llama3-8B's average AlpacaEval response time on an NVIDIA H100 96GB. Our results (Figure 9) reaffirm the security-utility-efficiency trade-off: more complex combinations achieve greater ASR reduction at slight utility and delay cost.

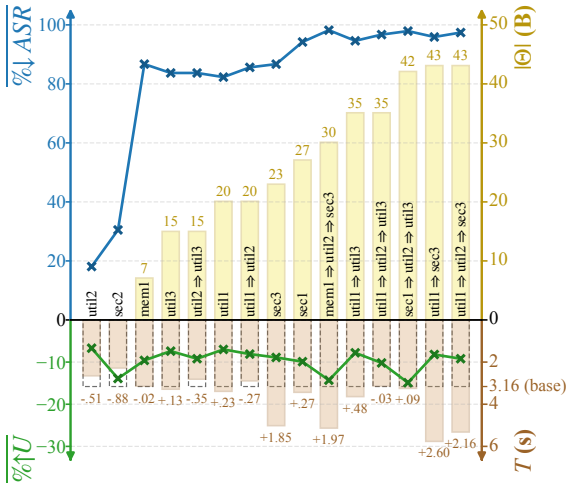


Figure 9: (P4) Security, utility, and efficiency of selected cross-stage defense combinations. (Base delay: Llama3-8B's average AlpacaEval latency without defense.)

Representative scenarios translate this trade-off into three scenarios that practitioners can use directly or adapt to their needs, summarized in Table 3 and detailed below.

(1) Security-critical systems prioritize ASR reduction and tolerate resource or latency overhead. Among combinations with high $\overline{\%\downarrow ASR}$, **util1** ⇒ **util2** ⇒ **sec3** offers strong utility preservation.

(2) Utility-sensitive applications value user experience through response quality and time. Among combinations with high $\overline{\%\uparrow U}$ and low ΔT, **util1** ⇒ **util3** achieves satisfactory ASR reduction.

(3) Memory-constrained deployments have limited resources and favor lightweight defenses. Among combinations with low $\lvert\Theta\rvert$, **mem1** best balances ASR reduction, utility preservation, and delay.

5 Generalizability

We test whether the effectiveness of cross-stage defense recommendations, selected on Vicuna and GPT-3.5, generalizes to different target LLMs and

Cross-stage defense combination	$\overline{\%\downarrow ASR}$	$\overline{\%\uparrow U}$	FPR
GPT-5.4-mini			
util1 ⇒ **util2** ⇒ **sec3**	100.0	+1.4	7.6
util1 ⇒ **util3**	92.3	-1.9	8.0
mem1	90.4	-6.3	1.2
gemini-3.1-flash-lite			
util1 ⇒ **util2** ⇒ **sec3**	96.3	-0.5	8.0
util1 ⇒ **util3**	96.1	-2.5	7.2
mem1	84.8	-10.1	1.2

Table 4: Generalization of recommended combinations ($\overline{\%\downarrow ASR}$ measured against all five attack representatives using HarmBench; $\overline{\%\uparrow U}$ measured on the target model using AlpacaEval; FPR (%) measured on the target model using XSTest; FPR is 0 on both target models without defense).

new attack queries. We evaluate them on GPT-5.4-mini and gemini-3.1-flash-lite, small models from a newer generation, against HarmBench's 200 malicious goals, only 13.5% overlapping with JBB-Behaviors. Beyond earlier metrics, we report a second utility measure using XSTest FPR, the proportion of rejected queries among 250 benign prompts. Lacking comparable cross-stage baselines, we report only our recommendations' performance (Table 4).

For security-critical applications, **util1** ⇒ **util2** ⇒ **sec3** retains high ASR reduction and strong utility preservation ($\overline{\%\uparrow U}$ even increases for GPT-5.4-mini, likely because **util2** refines benign queries to elicit clearer responses). On inspection, the relatively higher FPR stems from extreme boundary cases (e.g., queries about fictional characters' credit card details); the high $\overline{\%\uparrow U}$ suggests such rejections do not degrade perceived utility. The other two combinations also provide substantial protection: **util1** ⇒ **util3** preserves strong utility, while **mem1** offers a lightweight, low-FPR alternative. These results confirm that the recommended combinations effectively generalize beyond our earlier target LLMs and attack queries.

6 Conclusion

We present CASCADE, a practical framework for systematic, fair evaluation of jailbreak defense combinations within and across pipeline stages. Across 19 attacks and 15 defenses, four sequential processes identify: (P1) five representative attacks; (P2) top utility-preserving defenses per stage; (P3) the preferred per-stage defenses across security, utility, and memory trade-offs; and (P4) three recommended cross-stage combinations for security-critical, utility-sensitive, and memory-constrained scenarios. We find that no single defense is universally best, yet well-chosen combinations deliver substantial safety with minimal utility loss. Moreover, these combinations generalize beyond the evaluated LLMs and attack queries. We call on the community to extend CASCADE to broader attack categories, defense stages, and deployment scenarios as jailbreak threats evolve.

Limitations

Scope of attacks and defenses spans 19 attacks and 15 defenses widely studied in jailbreak research, though the broader technique space extends beyond our evaluation. For attacks, other methods exist within our chosen categories, and additional categories such as multi-turn jailbreaks pose substantial risks that fall outside our single-turn threat model (Russinovich et al., 2025). For defenses, we focus on three inference-time pipeline stages that require no model weight access or retraining, making them readily deployable. Other stages, such as training-time safety alignment, capture complementary aspects of LLM safety but typically require model weight access and substantial retraining compute, which are often infeasible for practitioners. Several guardrail defenses can serve as either input or output guards, but we evaluate each in only one of these roles. Expanding this scope is an important direction for future work.

In addition, P1's attack selection is based on undefended LLMs, possibly missing attacks that target specific defenses. Alternative selection methods have greater drawbacks: selecting attacks against specific defenses introduces its own bias, as the attack set could be seen as cherry-picked against those defenses, while directly testing all attacks against all defenses is prohibitively expensive. We therefore designed P1 to identify an efficient yet diverse set of attacks, collectively effective against the inherent alignment of undefended LLMs, and the reported results should be read with this choice in mind.

Choice of target LLMs balances open-weight and proprietary, legacy and modern models, covering diverse practical use cases. Nevertheless, state-of-the-art (SOTA) models are less explored, for three reasons. First, SOTA proprietary LLMs' extensive internal safety mechanisms drove attacks to

near-zero ASR in pilot runs, leaving limited baseline signal. Second, providers may adjust these mechanisms per account, for example by applying enhanced filters to accounts flagged for repeated policy violations (Anthropic, 2026), affecting cross-run consistency. Third, comprehensive evaluation of SOTA proprietary LLMs incurs substantial API costs. We therefore focus on widely used LLMs below the SOTA frontier. To address this gap, our generalizability test (Section 5) demonstrates that the effectiveness of our recommended combinations generalizes to GPT-5.4-mini and gemini-3.1-flash-lite, more recent LLMs than our primary targets. Future work could leverage proprietary red-teaming programs that grant controllable internal safety configurations to extend coverage to SOTA LLMs.

Metric coverage focuses on widely used metrics, namely ASR for jailbreak effectiveness and AlpacaEval for utility, but other evaluation metrics remain relevant. These include *Pass Guardrail Rate (PGR)* for jailbreak evaluation (Wang et al., 2026) and the *False Positive Rate (FPR)* of defenses on benign queries (Röttger et al., 2024; Cui et al., 2025) as a complementary utility measure (reported in our generalizability experiment). While broader metric coverage could capture additional facets of defense behavior, a focused metric set supports clearer comparison, thresholding, and decision-making for selecting practical defense combinations. Practitioners requiring broader coverage may adapt CASCADE by substituting alternative metrics, thresholds, and fairness rules.

Stage-wise optimization first identifies preferred configurations within each pipeline stage, then combines them across stages. This strategy keeps evaluation tractable but only ensures optimality within the search space; global optimality assumes independence between stages. In practice, defenses from different stages may interact (for example, an input modifier may alter queries in ways that affect output-guard behavior), and two configurations that are individually suboptimal may combine to produce a more effective cross-stage pipeline. Interaction-aware joint optimization across all pipeline stages is a possible direction for future work, given sufficient compute, to identify defense pipelines stronger than those from stage-wise selection.

Judge reliability remains an open challenge that may bias measured ASR and the resulting defense rankings. Our jailbreak evaluation uses the HarmBench Judge and the JBB-Behaviors dataset, both widely adopted in prior work; the HarmBench Judge additionally ranks best overall on the reliability-efficiency trade-off in our judge comparison (Appendix C). Improving judge reliability and the fairness of empirical comparisons remains an important future direction.

Ethical Considerations

We conduct our evaluation on publicly available jailbreak attacks, defenses, and benchmarks. Harmful content generated by attacks is not publicly released or shared outside the authors. We acknowledge that our empirical results reveal which attack strategies are most effective on each target LLM. We responsibly disclose our findings to relevant LLM providers, including OpenAI, Google, Anthropic, LMSYS, and Meta. We hope our results help developers and providers identify remaining risks and design safer systems.

We acknowledge using AI tools to assist with experiment code and to refine writing and figures in this report. All original ideas, including the methodology, implementation choices, presentation decisions, initial drafting, and final editing, were produced solely by the authors without AI assistance.

References

Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. 2025a. Jailbreaking leading safety-aligned LLMs with simple adaptive attacks. In *The Thirteenth International Conference on Learning Representations*.

Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, J Zico Kolter, Matt Fredrikson, Yarin Gal, and Xander Davies. 2025b. Agentharm: A benchmark for measuring harmfulness of LLM agents. In *The Thirteenth International Conference on Learning Representations*.

Anthropic. 2026. Our Approach to User Safety. Anthropic Help Center, https://support.claude.com/en/articles/8106465-our-approach-to-user-safety. [Accessed 20-05-2026].

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion,

George J. Pappas, Florian Tramèr, Hamed Hassani, and Eric Wong. 2024. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 55005–55029. Curran Associates, Inc.

Xiuyuan Chen, Jian Zhao, Yuxiang He, Yuan Xun, Xinwei Liu, Yanshu Li, Huilin Zhou, Wei Cai, Ziyan Shi, Yuchen Yuan, Tianle Zhang, Chi Zhang, and Xuelong Li. 2025. Teleai-safety: A comprehensive llm jailbreaking benchmark towards attacks, defenses, and evaluations. *Preprint*, arXiv:2512.05485.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Alex Chouldechova, A. Feder Cooper, Solon Barocas, Abhinav Palia, Dan Vann, and Hanna Wallach. 2025. Comparison requires valid measurement: Rethinking attack success rate comparisons in ai red teaming. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc.

Junjie Chu, Yugeng Liu, Ziqing Yang, Xinyue Shen, Michael Backes, and Yang Zhang. 2025. JailbreakRadar: Comprehensive assessment of jailbreak attacks against LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21538–21566, Vienna, Austria. Association for Computational Linguistics.

Justin Cui, Wei-Lin Chiang, Ion Stoica, and Cho-Jui Hsieh. 2025. OR-bench: An over-refusal benchmark for large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 11515–11542. PMLR.

Yue Deng, Wenxuan Zhang, Sinno Jialin Pan, and Lidong Bing. 2024. Multilingual jailbreak challenges in large language models. In *The Twelfth International Conference on Learning Representations*.

Peng Ding, Jun Kuang, Dan Ma, Xuezhi Cao, Yunsen Xian, Jiajun Chen, and Shujian Huang. 2024. A wolf in sheep's clothing: Generalized nested jailbreak prompts can fool large language models easily. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2136–2153, Mexico City, Mexico. Association for Computational Linguistics.

Yann Dubois, Percy Liang, and Tatsunori Hashimoto. 2024. Length-controlled AlpacaEval: A simple debiasing of automatic evaluators. In *First Conference on Language Modeling*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. 2024. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms. In *Advances in Neural Information Processing Systems*, volume 37, pages 8093–8131. Curran Associates, Inc.

Ruixuan Huang, Xunguang Wang, Zongjie Li, Daoyuan Wu, and Shuai Wang. 2025. Guidedbench: Measuring and mitigating the evaluation discrepancies of in-the-wild llm jailbreak methods. *Preprint*, arXiv:2502.16903.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. 2023. Llama guard: Llm-based input-output safeguard for human-ai conversations. *Preprint*, arXiv:2312.06674.

Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. 2023. Baseline defenses for adversarial attacks against aligned language models. *Preprint*, arXiv:2309.00614.

Xiaojun Jia, Tianyu Pang, Chao Du, Yihao Huang, Jindong Gu, Yang Liu, Xiaochun Cao, and Min Lin. 2025. Improved techniques for optimization-based jailbreaking on large language models. In *The Thirteenth International Conference on Learning Representations*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 611–626, New York, NY, USA. Association for Computing Machinery.

LangChain, Inc. 2026. LangGraph: Agent orchestration framework for reliable AI agents. `https://www.langchain.com/langgraph`. [Accessed 24-05-2026].

Gyeong-Geon Lee, Ehsan Latif, Xuansheng Wu, Ninghao Liu, and Xiaoming Zhai. 2024. Applying large language models and chain-of-thought for automatic scoring. *Computers and Education: Artificial Intelligence*, 6:100213.

Zeyi Liao and Huan Sun. 2024. AmpleGCG: Learning a universal and transferable generative model of adversarial suffixes for jailbreaking both open and closed LLMs. In *First Conference on Language Modeling*.

Junyu Lin, Meizhen Liu, Xiufeng Huang, Jinfeng Li, Haiwen Hong, Xiaohan Yuan, Yuefeng Chen, Longtao Huang, Hui Xue, Ranjie Duan, Zhikai Chen, Yuchuan Fu, Defeng Li, Lingyao Gao, and Yitong Yang. 2026. Yufeng-xguard: A reasoning-centric, interpretable, and flexible guardrail model for large language models. *Preprint*, arXiv:2601.15588.

Yue Liu, Hongcheng Gao, Shengfang Zhai, Yufei He, Jun Xia, Zhengyu Hu, Yulin Chen, Xihong Yang, Jiaheng Zhang, Stan Z. Li, Hui Xiong, and Bryan Hooi. 2025a. Guardreasoner: Towards reasoning-based llm safeguards. *Preprint*, arXiv:2501.18492.

Yue Liu, Xiaoxin He, Miao Xiong, Jinlan Fu, Shumin Deng, Yingwei Ma, Jiaheng Zhang, and Bryan Hooi. 2025b. FlipAttack: Jailbreak LLMs via flipping. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 38623–38663. PMLR.

Todor Markov, Chong Zhang, Sandhini Agarwal, Florentine Eloundou Nekoul, Theodore Lee, Steven Adler, Angela Jiang, and Lilian Weng. 2023. A holistic approach to undesired content detection in the real world. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(12):15009–15018.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. 2024. HarmBench: A standardized evaluation framework for automated red teaming and robust refusal. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 35181–35224. PMLR.

Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. 2024. Tree of attacks: Jailbreaking black-box llms automatically. In *Advances in Neural Information Processing Systems*, volume 37, pages 61065–61105. Curran Associates, Inc.

Meta. 2025. Llama Prompt Guard 2 Model Card (meta-llama/Llama-Prompt-Guard-2-86M). `https://huggingface.co/meta-llama/Llama-Prompt-Guard-2-86M`. [Accessed 02-11-2025].

Yichuan Mo, Yuji Wang, Zeming Wei, and Yisen Wang. 2024. Fight back against jailbreaking via prompt adversarial tuning. In *Advances in Neural Information Processing Systems*, volume 37, pages 64242–64272. Curran Associates, Inc.

Milad Nasr, Nicholas Carlini, Chawin Sitawarin, Sander V. Schulhoff, Jamie Hayes, Michael Ilie, Juliette Pluto, Shuang Song, Harsh Chaudhari, Ilia Shumailov, Abhradeep Thakurta, Kai Yuanqing Xiao, Andreas Terzis, and Florian Tramèr. 2025. The attacker moves second: Stronger adaptive attacks bypass defenses against LLM jailbreaks and prompt injections. *Preprint*, arXiv:2510.09023.

OpenAI. 2025. Technical report: Performance and baseline evaluations of gpt-oss-safeguard-120b and gpt-oss-safeguard-20b. Technical report, OpenAI Technical Report. Model weights: `https://huggingface.co/openai/gpt-oss-safeguard-20b`.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

OpenRouter. 2026. LLM Rankings. `https://openrouter.ai/rankings`. [Accessed 24-05-2026].

OWASP. 2024. OWASP top 10 for LLM applications 2025. Technical report, OWASP Foundation.

Fábio Perez and Ian Ribeiro. 2022. Ignore previous prompt: Attack techniques for language models. *Preprint*, arXiv:2211.09527.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. 2024. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *The Twelfth International Conference on Learning Representations*.

Raine v. OpenAI, Inc. 2025. Complaint and demand for jury trial. Superior Court of California, County of San Francisco. Case No. CGC-25-628528, filed August 26, 2025.

Qibing Ren, Chang Gao, Jing Shao, Junchi Yan, Xin Tan, Wai Lam, and Lizhuang Ma. 2024. CodeAttack: Revealing safety generalization challenges of large language models via code completion. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11437–11452, Bangkok, Thailand. Association for Computational Linguistics.

Alexander Robey, Eric Wong, Hamed Hassani, and George J. Pappas. 2025. SmoothLLM: Defending large language models against jailbreaking attacks. *Transactions on Machine Learning Research*.

Paul Röttger, Hannah Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. 2024. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5377–5400, Mexico City, Mexico. Association for Computational Linguistics.

Mark Russinovich, Ahmed Salem, and Ronen Eldan. 2025. Great, now write an article about that: The crescendo Multi-Turn LLM jailbreak attack. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 2421–2440, Seattle, WA. USENIX Association.

Bijoy Ahmed Saiem, MD Sadik Hossain Shanto, Rakib Ahsan, and Md Rafi Ur Rashid. 2025. SequentialBreak: Large language models can be fooled by embedding jailbreak prompts into sequential prompt chains. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 548–579, Vienna, Austria. Association for Computational Linguistics.

Guobin Shen, Dongcheng Zhao, Linghao Feng, Xiang He, Jihang Wang, Sicheng Shen, Haibo Tong, Yiting Dong, Jindong Li, Xiang Zheng, and Yi Zeng. 2025. Pandaguard: Systematic evaluation of llm safety against jailbreaking attacks. *Preprint*, arXiv:2505.13862.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. 2024. "do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, page 1671–1685, New York, NY, USA. Association for Computing Machinery.

Peter Steinberger and the OpenClaw community. 2026. OpenClaw: Your own personal AI assistant. `https://github.com/openclaw/openclaw`. Version 2026.3.28, released March 29, 2026.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Apostol Vassilev, Alina Oprea, Alie Fordyce, Hyrum Anderson, Xander Davies, and Maia Hamin. 2025. Adversarial machine learning: A taxonomy and terminology of attacks and mitigations. NIST Trustworthy and Responsible AI NIST AI 100-2e2025, National Institute of Standards and Technology, Gaithersburg, MD.

Kun Wang, Guibin Zhang, Zhenhong Zhou, Jiahao Wu, Miao Yu, Shiqian Zhao, Chenlong Yin, Jinhu Fu, Yibo Yan, Hanjun Luo, Liang Lin, Zhihao Xu, Haolang Lu, Xinye Cao, Xinyun Zhou, Weifei Jin, Fanci Meng, Shicheng Xu, Junyuan Mao, and 84 others. 2025. A comprehensive survey in llm(-agent) full stack safety: Data, training and deployment. *Preprint*, arXiv:2504.15585.

Xunguang Wang, Zhenlan Ji, Wenxuan Wang, Zongjie Li, Daoyuan Wu, and Shuai Wang. 2026. SoK: Evaluating jailbreak guardrails for large language models. In *2026 IEEE Symposium on Security and Privacy (SP)*, pages 39–58, Los Alamitos, CA, USA. IEEE Computer Society.

Yihan Wang, Zhouxing Shi, Andrew Bai, and Cho-Jui Hsieh. 2024. Defending LLMs against jailbreaking attacks via backtranslation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 16031–16046, Bangkok, Thailand. Association for Computational Linguistics.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2023. Jailbroken: How does LLM safety training fail? In *Advances in Neural Information Processing Systems*, volume 36, pages 80079–80110. Curran Associates, Inc.

Zeming Wei, Yifei Wang, Ang Li, Yichuan Mo, and Yisen Wang. 2026. Jailbreak and guard aligned language models with only few in-context demonstrations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(6):6835–6846.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yueqi Xie, Minghong Fang, Renjie Pi, and Neil Gong. 2024. GradSafe: Detecting jailbreak prompts for LLMs via safety-critical gradient analysis. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 507–518, Bangkok, Thailand. Association for Computational Linguistics.

Yueqi Xie, Jingwei Yi, Jiawei Shao, Justin Curl, Lingjuan Lyu, Qifeng Chen, Xing Xie, and Fangzhao Wu. 2023. Defending ChatGPT against jailbreak attack via self-reminders. *Nature Machine Intelligence*, 5(12):1486–1496.

Chen Xiong, Xiangyu Qi, Pin-Yu Chen, and Tsung-Yi Ho. 2025. Defensive prompt patch: A robust and generalizable defense of large language models against jailbreak attacks. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 409–437, Vienna, Austria. Association for Computational Linguistics.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Jinyuan Jia, Bill Yuchen Lin, and Radha Poovendran. 2024a. SafeDecoding: Defending against jailbreak attacks via safety-aware decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*,

pages 5587–5605, Bangkok, Thailand. Association for Computational Linguistics.

Zihao Xu, Yi Liu, Gelei Deng, Yuekang Li, and Stjepan Picek. 2024b. A comprehensive study of jailbreak attack versus defense for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7432–7449, Bangkok, Thailand. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Sibo Yi, Yule Liu, Zhen Sun, Tianshuo Cong, Xinlei He, Jiaxing Song, Ke Xu, and Qi Li. 2024. Jailbreak attacks and defenses against large language models: A survey. *Preprint*, arXiv:2407.04295.

Jiahao Yu, Xingwei Lin, Zheng Yu, and Xinyu Xing. 2024. LLM-Fuzzer: Scaling assessment of large language model jailbreaks. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 4657–4674, Philadelphia, PA. USENIX Association.

Yi Zeng, Hongpeng Lin, Jingwen Zhang, Diyi Yang, Ruoxi Jia, and Weiyan Shi. 2024. How johnny can persuade LLMs to jailbreak them: Rethinking persuasion to challenge AI safety by humanizing LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14322–14350, Bangkok, Thailand. Association for Computational Linguistics.

Zhexin Zhang, Leqi Lei, Junxiao Yang, Xijie Huang, Yida Lu, Shiyao Cui, Renmiao Chen, Qinglin Zhang, Xinyuan Wang, Hao Wang, Hao Li, Xianqi Lei, Chengwei Pan, Lei Sha, Hongning Wang, and Minlie Huang. 2025. Aisafetylab: A comprehensive framework for ai safety evaluation and improvement. *Preprint*, arXiv:2502.16776.

Zhexin Zhang, Yida Lu, Jingyuan Ma, Di Zhang, Rui Li, Pei Ke, Hao Sun, Lei Sha, Zhifang Sui, Hongning Wang, and Minlie Huang. 2024. ShieldLM: Empowering LLMs as aligned, customizable and explainable safety detectors. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10420–10438, Miami, Florida, USA. Association for Computational Linguistics.

Andy Zhou, Bo Li, and Haohan Wang. 2024. Robust prompt optimization for defending language models against jailbreaking attacks. In *Advances in Neural Information Processing Systems*, volume 37, pages 40184–40211. Curran Associates, Inc.

Yukai Zhou, Jian Lou, Zhijie Huang, Zhan Qin, Sibei Yang, and Wenjie Wang. 2025. Don't say no: Jailbreaking LLM by suppressing refusal. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 25224–25249, Vienna, Austria. Association for Computational Linguistics.

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

A Implementation Details

A.1 Attacks and Defenses

We list the jailbreak attacks and defenses covered in this work in Tables 5 and 6.

A.2 Target LLMs

We list the target LLMs evaluated in this work in Table 7.

A.3 Experimental Settings

We use the **LangGraph** framework (LangChain, Inc., 2026) to orchestrate attacks and defenses through a stateful agentic execution graph. We instantiate open-source LLMs using the **transformers** library (Wolf et al., 2020) or the **vLLM** inference engine (Kwon et al., 2023), based on recommended settings and generation efficiency. We conduct experiments on NVIDIA GPUs, including H200-141, H100-96, and A100-80, selected based on each experiment's memory requirement. We estimate approximately 2200 total GPU-hours, covering experiments on both open-weight and proprietary LLMs.

For all judge models, including jailbreak judges and the AlpacaEval utility judge, we set temperature to 0 for consistent evaluation. For target LLMs, we set temperature to 1 for jailbreak evaluation (supporting repeated attack attempts) and temperature to 0 for utility evaluation (ensuring consistent results). All other LLMs in attack or defense techniques follow the original authors' recommended settings where applicable. We set a constant seed of 42 to maximize reproducibility, and set max_new_tokens to 2048 for non-reasoning models and 4096 for reasoning models to avoid unexpected output truncation.

B Fairness Rules

Empirical jailbreak studies often vary in attacker assumptions, query budgets, and selection criteria, making cross-study comparison difficult. CASCADE's four-process framework therefore adopts five fairness rules to support consistent, controlled comparison across attacks and defenses. We explain each rule's rationale below and the process(es) it governs.

Label	Full form	Authors	No. variants / Query budget per run	Remarks
White-box transferable (query budget: 15)				
GCG	Greedy Coordinate Gradient	Zou et al. (2023)	5	5 transferable suffixes were obtained, with the first 2 retrieved from the authors' repository, 1 generated using authors' code on Vicuna-7B, 2 generated using authors' code on Llama2-7B
AmpleGCG	AmpleGCG	Liao and Sun (2024)	5	5 transferable suffixes were obtained for each malicious goal using AmpleGCG's generative model from Hugging Face
I-GCG	Improved techniques for GCG	Jia et al. (2025)	5	5 transferable suffixes were obtained for each malicious goal using authors' code
DSN	Don't Say No	Zhou et al. (2025)	5	5 transferable suffixes were obtained using authors' code, with the 2 generated from Vicuna-7B, and 3 generated from Llama2-7B
Adaptive	Adaptive attacks	Andriushchenko et al. (2025a)	1	Template and transferable suffix obtained from authors' repository
Black-box template-based, output style pattern (query budget: 5)				
Prefix	Prefix Injection	Wei et al. (2023)	1	Template obtained from authors' paper
Refusal	Refusal Suppression	Wei et al. (2023)	1	Template obtained from authors' paper
Wiki	Wikipedia-style	Wei et al. (2023)	1	Template obtained from authors' paper
Black-box template-based, persona pattern (query budget: 5)				
DAN	Do-Anything-Now	Shen et al. (2024)	1	Template obtained from authors' paper
DevMode	Developer Mode	Shen et al. (2024)	1	Template obtained from authors' paper
AIM	Always Intelligent and Machiavellian	Shen et al. (2024)	1	Template obtained from authors' paper
Black-box template-based, disguise pattern (query budget: 36)				
Code	CodeAttack	Ren et al. (2024)	3	3 variants (python_stack_plus, python_list, python_string) obtained from authors' repository
Flip	FlipAttack	Liu et al. (2025b)	6	Among all possible variants, we selected 6 variants whose results were presented by the authors, including FWO, FCS + CoT, FCW + CoT, FMM + CoT, FCS + CoT + LangGPT, FCS + CoT + LangGPT + Few-shot
MultiJail	MultiJail	Deng et al. (2024)	9	9 variant languages suggested by the authors were used; we translated with Google Translate, whereas the authors used native-speaker translations
SeqBreak	SequentialBreak	Saiem et al. (2025)	3	3 purely template-based variants were used, including Question Bank 2, Game Environment 1 and Game Environment 2
Black-box LLM-based (query budget: 80)				
TAP	Tree of Attacks with Pruning	Mehrotra et al. (2024)	80	Vicuna-13B was used as the attacker; GPT-OSS-20B was used as the judge, as an intelligent and resource-efficient replacement of GPT-4
PAP	Persuasive Adversarial Prompts	Zeng et al. (2024)	40	Llama3-70B was used to generate attack prompts using authors' 40 persuasion techniques
ReNeLLM	ReNeLLM	Ding et al. (2024)	20	Vicuna-13B was used as the attacker, as a resource-efficient replacement of GPT-3.5
GPTFuzzer	GPTFuzzer	Yu et al. (2024)	80	GPT-OSS-20B was used as the mutator, as an intelligent and resource-efficient replacement of the authors' ChatGPT (gpt-3.5-turbo)

Table 5: Jailbreak attacks evaluated in this work.

Label	Full form	Authors	Remarks
Input guard			
PPL	Perplexity filter	Jain et al. (2023)	Perplexity is computed using Llama2-7B
W-PPL	Windowed perplexity filter	Jain et al. (2023)	Perplexity is computed using Llama2-7B; window size is set to 10 tokens
PG	Prompt Guard 2	Meta (2025)	Reject the prompt if classified as malicious
WG	WildGuard (input)	Han et al. (2024)	Reject the prompt if classified as harmful
OSS	GPT-OSS-safeguard 20B	OpenAI (2025)	Reject the prompt if classified as harmful
Mod	OpenAI moderation (input)	Markov et al. (2023)	Reject the prompt if flagged by the API
Input modification			
S-LLM	SmoothLLM	Robey et al. (2025)	We used "RandomInsertPerturbation" mode, perturbation percent of 10, and other default parameters
SR	Self-reminder	Xie et al. (2023)	Defense template obtained from authors' paper
ICD	In-context defense	Wei et al. (2026)	10 pre-sampled rejection examples are used (the authors use 1–2 demonstrations)
PAT	Prompt Adversarial Tuning	Mo et al. (2024)	Defense prefix obtained from authors' repository
DPP	Defensive Prompt Patch	Xiong et al. (2025)	Defense suffix obtained from authors' repository
RPO	Robust Prompt Optimization	Zhou et al. (2024)	Defense suffix obtained from authors' repository
Output guard			
LG	Llama Guard 3	Inan et al. (2023)	Reject the conversation if classified as one of the hazard categories
WG	WildGuard (output)	Han et al. (2024)	Reject the conversation if classified as harmful
Mod	OpenAI moderation (output)	Markov et al. (2023)	Reject the conversation if flagged by the API
XG	YuFeng-XGuard-Reason	Lin et al. (2026)	Reject the conversation if classified as harmful
GR	GuardReasoner	Liu et al. (2025a)	Reject the conversation if classified as harmful

Table 6: Jailbreak defenses evaluated in this work.

Label	Full model ID	Remarks
Open-weight LLMs		
Vicuna	lmsys/vicuna-7b-v1.5 (Chiang et al., 2023)	-
Llama2	meta-llama/Llama-2-7b-chat-hf (Touvron et al., 2023)	The Llama2 family originally had a recommended safety system prompt; we do not set it, since the model provider removed it in a later update
Llama3	meta-llama/Llama-3.1-8B-Instruct (Grattafiori et al., 2024)	We also do not set a safety system prompt
Proprietary LLMs		
GPT-3.5	gpt-3.5-turbo-0125	-
GPT-4o	gpt-4o-2024-08-06	-
GPT-5.4-mini	gpt-5.4-mini-2026-03-17	Reasoning is turned off
Claude-sonnet-4	claude-sonnet-4-20250514	Reasoning is turned off
Gemini-2.5-flash	gemini-2.5-flash	Reasoning is turned off
Gemini-3.1-flash-lite	gemini-3.1-flash-lite	Reasoning is turned off

Table 7: Target LLMs evaluated in this work.

(1) Same attacker capabilities. Technique implementations adopt inconsistent assumptions about attacker capabilities. For example, some techniques require modifying the target model's system prompt, an assumption that varies across studies and is less applicable to practical LLM systems. Following recent calls for comparisons under a consistent threat model (Chu et al., 2025), we re-

strict all attacks to the direct, black-box, single-turn setting, where the attacker controls only the user prompt.

(2) Same query budget per subcategory. During subcategory-wise attack selection, all techniques within a subcategory receive the same target-model query budget. This budget captures practical attacker constraints, including API cost during vulnerability reconnaissance and detection risk from repeated suspicious queries. Equalizing the query budget therefore supports fair comparison of attack effectiveness under realistic constraints, consistent with the requirement that attempt-based ASRs be compared only at matched budgets (Chouldechova et al., 2025).

(3) Top-k utility selection. Our primary objective is to minimize jailbreak risk while preserving response quality. We therefore treat utility preservation as a constraint: we first evaluate defense configurations on utility, advancing only those with satisfactory preservation. For the acceptance criterion, we considered a fixed utility threshold (e.g., a 10% degradation limit), but such thresholds are hard to justify because AlpacaEval measures relative response quality rather than an interpretable absolute utility level. We therefore select configurations by relative utility rank within comparable candidate groups.

(4) Same query budgets as in (P1). When evaluating defenses against representative attacks, each attack retains the query budget used during initial selection. This preserves the practical attack capability established in (P1), ensuring the selected attacks remain strong adversarial choices. It also provides reliable baseline signal on undefended LLMs for accurate defense assessment.

(5) Balanced performance evaluation. Jailbreak defenses are known to exhibit trade-offs among security, utility, and efficiency (Wang et al., 2026), all of which matter in practical deployment. While our primary objective is to reduce jailbreak risk subject to satisfactory utility preservation, deployment recommendations should also account for resource efficiency. We therefore organize the recommendations in (P3) and (P4) around three practical objectives: high security, high utility, and high efficiency. Within each objective, selections still prioritize security and utility; among configurations with similar performance on both, we choose the more efficient option.

C Jailbreak Evaluation

ASR evaluation results depend on both the malicious-goal dataset and the jailbreak judge. For the dataset, we considered widely used benchmarks, including **AdvBench**, **JBB-Behaviors**, and **HarmBench**, alongside newer alternatives. For example, **GuidedBench** curates malicious goals from existing benchmarks by requiring that modern target LLMs refuse the original queries and that the queries remain direct and answerable, for reliable jailbreak judgment (Huang et al., 2025). These criteria raise valid concerns about dataset quality: in our attack experiments (Table 1), raw **JBB-Behaviors** goals achieve considerable ASR on legacy models and non-zero ASR on newer models, possibly because some entries no longer fall under updated safety policies. Nevertheless, our study does not seek to determine the precise boundary of harmful content. We treat each dataset entry as an undesirable goal that the target model should reject, and evaluate the relative ability of attacks and defenses to induce or prevent such outcomes. For efficiency at our experimental scale, we use **JBB-Behaviors**, a compact, representative set of 100 malicious goals widely adopted in prior work.

Jailbreak judge reliability also depends on the evaluation dataset, target LLM response patterns, and attack-induced outputs (Chouldechova et al., 2025). We conducted a small-scale agreement study to identify a judge suitable for our evaluation scope. Early keyword-matching evaluators produced frequent misjudgments in pilot runs and were excluded. We then compared four contemporary jailbreak judge models: **GPT-4** (Qi et al., 2024), **HarmBench** (Mazeika et al., 2024), **fine-tuned RoBERTa** (Xu et al., 2024b), and **ShieldLM** (Zhang et al., 2024). We conducted attack experiments in both undefended and defended settings, labeling outputs by majority vote of automated judges. We sampled outputs from ten experiments: four with largely questionable automated labels, three others with undefended LLMs, and three others with defended LLMs. From each experiment, we randomly sampled 25 outputs labeled as jailbroken and 25 labeled as not jailbroken. The authors then manually annotated all sampled outputs, without involving external annotators. Table 8 reports agreement metrics with human labels, measured using accuracy and Cohen's κ, where $\kappa = 1$ indicates complete agreement and $\kappa = -1$ indicates complete disagreement.

Jailbreak judge	Accuracy	Cohen's κ with human labels
Undefended setting		
HarmBench Judge	**0.820**	**0.638**
GPT-4 Judge	0.814	**0.638**
Fine-tuned RoBERTa Judge	0.714	0.380
ShieldLM Judge	0.694	0.339
Defended setting		
HarmBench Judge	**0.900**	**0.800**
GPT-4 Judge	0.847	0.693
Fine-tuned RoBERTa Judge	0.747	0.493
ShieldLM Judge	0.787	0.573

Table 8: Jailbreak judge agreement with human annotations measured by accuracy and Cohen's κ; bold values: maximum agreement.

In both settings, HarmBench Judge achieves the highest accuracy and Cohen's κ score against human labels. It also performs better in the controversial cases identified during manual inspection, and is among the fastest evaluators considered. Based on this balance of agreement quality and efficiency, we adopt HarmBench Judge in our experiments.

Dataset design and judge reliability are not the primary focus of this study, and our analysis of these choices remains limited in scope. Further development of evaluation schemes such as **GuidedBench** would improve jailbreak evaluation reliability and strengthen fair empirical comparison for practical defense selection.

D Full Evaluation Results

This appendix presents the full tables underlying CASCADE's experiments: initial attack selection (P1), attack-versus-defense evaluation (P3), final cross-stage combinations (P4), and the generalizability experiment (Section 5). Initial defense selection (P2) results are already presented in Section 4.2. Each subsection below lists the corresponding tables. Alongside $ASR_{@max_q}$, some tables additionally report:

$$ASR_{@once} = \frac{\sum_n \sum_d \max_v [I(Q_{d,n,v})]}{|D| \cdot N}, \qquad (9)$$

where a single *run* tries each variant once. $ASR_{@once}$ averages success across N runs, capturing the combined effectiveness of all variants.

D.1 Initial Attack Selection Full Results

Full results of (P1) initial selection of attacks are shown in Tables 9, 10, 11, 12, and 13.

D.2 Attacks Versus Defenses Full Results

Full results of (P3) attacks versus defenses are shown in Tables 14, 15, and 16.

D.3 Final Cross-Stage Combination Full Results

Full results of (P4) final cross-stage combination are shown in Tables 17 and 18.

D.4 Generalizability Full Results

Full results of the generalizability experiment on GPT-5.4-mini, gemini-3.1-flash-lite, HarmBench and XSTest (Section 5) are shown in Table 19.

Attack technique	Metric	Vicuna	Llama2	Llama3	gpt-3.5	gpt-4o	claude-sonnet-4	Average
Adaptive	$ASR_{@once}$	83.27	0.13	**84.87**	92.73	4.53	**11.93**	46.24
	$ASR_{@max_q}$	**100.00**	2.00	**100.00**	**100.00**	9.00	**44.00**	**59.17**
DSN	$ASR_{@once}$	79.33	**85.67**	18.33	88.00	11.00	1.67	**47.33**
	$ASR_{@max_q}$	93.00	**97.00**	30.00	94.00	13.00	2.00	54.83
GCG	$ASR_{@once}$	**97.00**	23.67	26.67	**98.33**	12.67	1.33	43.28
	$ASR_{@max_q}$	**100.00**	42.00	40.00	99.00	15.00	2.00	49.67
AmpleGCG	$ASR_{@once}$	94.67	12.33	19.00	92.33	**13.00**	3.33	39.11
	$ASR_{@max_q}$	98.00	19.00	26.00	98.00	**16.00**	5.00	43.67
I-GCG	$ASR_{@once}$	30.07	1.33	3.07	37.20	2.67	0.87	12.54
	$ASR_{@max_q}$	91.00	3.00	8.00	74.00	8.00	2.00	31.00
Baseline (plain malicious goal)	$ASR_{@once}$	32.40	2.87	7.07	31.33	4.60	3.20	13.58
	$ASR_{@max_q}$	83.00	7.00	22.00	56.00	7.00	5.00	30.00

Table 9: (P1) ASR results (%) of *White-box transferable* attacks. (Query budget: 15; bold: subcategory maximum)

Attack technique	Metric	Vicuna	Llama2	Llama3	gpt-3.5	gpt-4o	claude-sonnet-4	Average
Refusal	$ASR_{@once}$	74.20	**10.20**	**32.40**	**82.40**	**28.00**	**3.80**	**38.50**
	$ASR_{@max_q}$	94.00	**19.00**	**46.00**	**92.00**	**38.00**	**8.00**	**49.50**
Prefix	$ASR_{@once}$	**84.40**	4.40	5.20	75.40	3.20	1.40	29.00
	$ASR_{@max_q}$	**99.00**	7.00	10.00	87.00	6.00	4.00	35.50
Wiki	$ASR_{@once}$	61.60	3.60	5.20	45.00	3.60	0.00	19.83
	$ASR_{@max_q}$	80.00	9.00	7.00	62.00	5.00	0.00	27.17
Baseline (plain malicious goal)	$ASR_{@once}$	30.20	3.20	7.60	31.20	4.60	3.00	13.30
	$ASR_{@max_q}$	58.00	4.00	16.00	46.00	7.00	5.00	22.67

Table 10: (P1) ASR results (%) of *Black-box template-based, output style pattern* attacks. (Query budget: 5; bold: subcategory maximum)

Attack technique	Metric	Vicuna	Llama2	Llama3	gpt-3.5	gpt-4o	claude-sonnet-4	Average
DevMode	$ASR_{@once}$	66.00	2.00	**36.80**	**38.00**	0.00	0.00	**23.80**
	$ASR_{@max_q}$	99.00	**9.00**	**59.00**	**64.00**	0.00	0.00	**38.50**
DAN	$ASR_{@once}$	55.60	**3.20**	30.80	10.80	0.00	0.20	16.77
	$ASR_{@max_q}$	95.00	8.00	57.00	33.00	0.00	1.00	32.33
AIM	$ASR_{@once}$	**90.20**	1.40	23.40	0.80	0.00	0.00	19.30
	$ASR_{@max_q}$	**100.00**	5.00	33.00	2.00	0.00	0.00	23.33
Baseline (plain malicious goal)	$ASR_{@once}$	30.20	**3.20**	7.60	31.20	**4.60**	**3.00**	13.30
	$ASR_{@max_q}$	58.00	4.00	16.00	46.00	**7.00**	**5.00**	22.67

Table 11: (P1) ASR results (%) of *Black-box template-based, persona pattern* attacks. (Query budget: 5; bold: subcategory maximum)

Attack technique	Metric	Vicuna	Llama2	Llama3	gpt-3.5	gpt-4o	claude-sonnet-4	Average
SeqBreak	$ASR_{@once}$	51.67	**73.67**	61.17	86.33	**81.75**	**15.67**	**61.71**
	$ASR_{@max_q}$	**100.00**	97.00	**100.00**	**100.00**	**98.00**	**60.00**	**92.50**
Code	$ASR_{@once}$	60.33	47.33	**78.92**	71.00	76.50	10.92	57.50
	$ASR_{@max_q}$	99.00	**100.00**	**100.00**	**100.00**	97.00	24.00	86.67
Flip	$ASR_{@once}$	62.00	19.83	35.00	**91.50**	54.33	0.00	43.78
	$ASR_{@max_q}$	98.00	58.00	81.00	**100.00**	80.00	0.00	69.50
MultiJail	$ASR_{@once}$	**98.75**	37.75	61.75	61.75	0.00	0.00	43.33
	$ASR_{@max_q}$	**100.00**	71.00	88.00	95.00	0.00	0.00	59.00
Baseline (plain malicious goal)	$ASR_{@once}$	31.97	2.64	7.11	31.67	4.89	3.44	13.62
	$ASR_{@max_q}$	94.00	9.00	26.00	63.00	8.00	6.00	34.33

Table 12: (P1) ASR results (%) of *Black-box template-based, disguise pattern* attacks. (Query budget: 36; bold: subcategory maximum)

Attack technique	Metric	Vicuna	Llama2	Llama3	gpt-3.5	gpt-4o	claude-sonnet-4	Average
ReNeLLM	$ASR_{@once}$	59.75	**51.75**	55.75	63.25	**55.75**	3.25	48.25
	$ASR_{@max_q}$	91.00	**81.00**	91.00	**95.00**	**87.00**	12.00	**76.17**
PAP	$ASR_{@once}$	53.50	31.00	**91.00**	45.50	32.00	1.50	42.42
	$ASR_{@max_q}$	70.00	42.00	**94.00**	59.00	41.00	3.00	51.50
GPTFuzzer	$ASR_{@once}$	73.00	39.00	79.00	69.00	34.00	1.00	**49.17**
	$ASR_{@max_q}$	73.00	39.00	79.00	69.00	34.00	1.00	49.17
TAP	$ASR_{@once}$	**75.00**	34.00	33.00	**78.00**	36.00	**17.00**	45.50
	$ASR_{@max_q}$	75.00	34.00	33.00	78.00	36.00	**17.00**	45.50
Baseline (plain malicious goal)	$ASR_{@once}$	32.90	2.70	7.04	31.65	4.59	3.54	13.74
	$ASR_{@max_q}$	**98.00**	9.00	27.00	67.00	11.00	9.00	36.83

Table 13: (P1) ASR results (%) of *Black-box LLM-based* attacks. (Query budget: 80; bold: subcategory maximum).

Defense configuration	Metric	Adaptive		DSN		Refusal		SeqBreak		ReNeLLM	
		V	G	V	G	V	G	V	G	V	G
Baseline (no defense)	$ASR_{@once}$	83.27	92.73	79.33	88.00	74.20	82.40	51.67	86.33	59.75	63.25
	$ASR_{@max_q}$	100.00	100.00	93.00	94.00	94.00	92.00	100.00	100.00	91.00	95.00
PPL	$ASR_{@once}$	83.27	92.73	0.67	2.00	77.80	80.40	51.67	86.33	64.00	67.75
	$ASR_{@max_q}$	100.00	100.00	1.00	2.00	94.00	92.00	100.00	100.00	94.00	96.00
PG	$ASR_{@once}$	**0.00**	**0.00**	36.33	59.33	**0.00**	**0.00**	24.75	47.83	63.00	66.50
	$ASR_{@max_q}$	**0.00**	**0.00**	60.00	71.00	**0.00**	**0.00**	57.00	59.00	94.00	96.00
WG	$ASR_{@once}$	**0.00**	**0.00**	2.00	2.33	1.00	0.80	6.83	15.08	25.00	37.25
	$ASR_{@max_q}$	**0.00**	**0.00**	3.00	3.00	2.00	2.00	23.00	26.00	60.00	72.00
oss	$ASR_{@once}$	10.20	13.53	14.67	15.33	11.00	12.20	5.67	14.75	23.00	28.75
	$ASR_{@max_q}$	18.00	17.00	18.00	17.00	17.00	16.00	26.00	30.00	62.00	64.00
{PPL, PG}	$ASR_{@once}$	**0.00**	**0.00**	0.33	1.00	**0.00**	**0.00**	24.75	47.83	42.75	47.75
	$ASR_{@max_q}$	**0.00**	**0.00**	1.00	1.00	**0.00**	**0.00**	57.00	59.00	86.00	87.00
{PPL, oss}	$ASR_{@once}$	10.20	13.53	0.67	1.00	11.80	11.40	5.67	14.75	15.25	22.00
	$ASR_{@max_q}$	18.00	17.00	1.00	1.00	16.00	15.00	26.00	30.00	47.00	55.00
{PG, WG}	$ASR_{@once}$	**0.00**	**0.00**	1.33	2.33	**0.00**	**0.00**	2.83	6.42	16.50	25.75
	$ASR_{@max_q}$	**0.00**	**0.00**	3.00	3.00	**0.00**	**0.00**	9.00	10.00	43.00	57.00
{PG, oss}	$ASR_{@once}$	**0.00**	**0.00**	9.33	11.33	**0.00**	**0.00**	4.83	12.42	14.75	19.50
	$ASR_{@max_q}$	**0.00**	**0.00**	16.00	14.00	**0.00**	**0.00**	19.00	21.00	48.00	50.00
{PPL, PG, WG}	$ASR_{@once}$	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	2.83	6.42	10.00	19.00
	$ASR_{@max_q}$	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	9.00	10.00	30.00	44.00
{PPL, PG, oss}	$ASR_{@once}$	**0.00**	**0.00**	0.33	0.67	**0.00**	**0.00**	4.83	12.42	10.25	15.50
	$ASR_{@max_q}$	**0.00**	**0.00**	1.00	1.00	**0.00**	**0.00**	19.00	21.00	38.00	42.00
{PG, WG, oss}	$ASR_{@once}$	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**1.42**	**3.83**	4.50	8.50
	$ASR_{@max_q}$	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**6.00**	**7.00**	16.00	26.00
{PPL, PG, WG, oss}	$ASR_{@once}$	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**1.42**	**3.83**	**2.50**	**7.25**
	$ASR_{@max_q}$	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**6.00**	**7.00**	**10.00**	**21.00**

Table 14: (P3) ASR results (%) of the representative attacks against *input filter* per-stage configurations. (V: Vicuna; G: GPT-3.5; Bold: stage-wise minimum; underlined: stage-wise second smallest values)

Defense configuration	Metric	Adaptive		DSN		Refusal		SeqBreak		ReNeLLM	
		V	G	V	G	V	G	V	G	V	G
Baseline (no defense)	$ASR_{@once}$	83.27	92.73	79.33	88.00	74.20	82.40	51.67	86.33	59.75	63.25
	$ASR_{@max_q}$	100.00	100.00	93.00	94.00	94.00	92.00	100.00	100.00	91.00	95.00
SR	$ASR_{@once}$	78.13	80.40	35.00	2.67	54.80	56.60	31.33	77.08	60.25	57.75
	$ASR_{@max_q}$	100.00	96.00	59.00	3.00	82.00	69.00	96.00	**98.00**	92.00	94.00
ICD	$ASR_{@once}$	67.80	86.67	37.67	0.67	68.80	69.80	48.25	90.00	63.00	66.25
	$ASR_{@max_q}$	100.00	99.00	68.00	1.00	98.00	86.00	99.00	100.00	97.00	97.00
RPO	$ASR_{@once}$	82.47	91.33	70.33	76.00	61.60	73.60	41.25	84.92	61.75	60.75
	$ASR_{@max_q}$	100.00	100.00	89.00	82.00	93.00	86.00	99.00	100.00	96.00	93.00
DPP	$ASR_{@once}$	81.40	92.67	71.33	66.00	67.60	80.40	50.50	88.75	68.25	70.25
	$ASR_{@max_q}$	100.00	100.00	93.00	74.00	92.00	91.00	99.00	100.00	95.00	95.00
SR ⇒ DPP	$ASR_{@once}$	78.73	72.47	34.00	2.67	44.80	55.40	**27.33**	74.42	60.75	50.75
	$ASR_{@max_q}$	**99.00**	97.00	57.00	4.00	78.00	68.00	95.00	**98.00**	90.00	87.00
ICD ⇒ SR	$ASR_{@once}$	**51.00**	**2.93**	**17.00**	**0.00**	56.80	32.40	45.00	**66.50**	59.25	43.00
	$ASR_{@max_q}$	100.00	**22.00**	39.00	**0.00**	85.00	55.00	100.00	**98.00**	90.00	80.00
SR ⇒ ICD	$ASR_{@once}$	62.93	37.93	28.33	**0.00**	49.00	42.80	38.17	76.50	57.50	43.25
	$ASR_{@max_q}$	**99.00**	78.00	54.00	**0.00**	82.00	66.00	97.00	100.00	97.00	**79.00**
SR ⇒ ICD ⇒ DPP	$ASR_{@once}$	60.47	47.87	19.00	2.00	**29.60**	**32.20**	30.17	78.75	**48.50**	**42.50**
	$ASR_{@max_q}$	**99.00**	81.00	**38.00**	2.00	**69.00**	**51.00**	**90.00**	99.00	**88.00**	82.00

Table 15: (P3) ASR results (%) of the representative attacks against *input modification* per-stage configurations. (V: Vicuna; G: GPT-3.5; Bold: stage-wise minimum; underlined: stage-wise second smallest values)

Defense configuration	Metric	Adaptive		DSN		Refusal		SeqBreak		ReNeLLM	
		V	G	V	G	V	G	V	G	V	G
Baseline (no defense)	$ASR_{@once}$	83.27	92.73	79.33	88.00	74.20	82.40	51.67	86.33	59.75	63.25
	$ASR_{@max_q}$	100.00	100.00	93.00	94.00	94.00	92.00	100.00	100.00	91.00	95.00
LG	$ASR_{@once}$	1.47	1.40	8.00	11.33	1.20	2.00	35.25	74.30	16.00	15.25
	$ASR_{@max_q}$	6.00	4.00	13.00	17.00	4.00	3.00	97.00	97.00	39.00	41.00
WG	$ASR_{@once}$	6.20	5.07	8.33	12.00	4.80	3.20	15.50	30.33	13.25	12.25
	$ASR_{@max_q}$	24.00	12.00	14.00	20.00	9.00	6.00	65.00	60.00	38.00	35.00
XG	$ASR_{@once}$	35.73	56.93	62.33	78.33	40.20	27.20	51.17	85.67	56.75	66.50
	$ASR_{@max_q}$	46.00	96.00	81.00	92.00	52.00	54.00	100.00	100.00	91.00	96.00
GR	$ASR_{@once}$	2.87	3.40	9.33	10.67	2.80	2.20	34.50	60.25	13.50	14.50
	$ASR_{@max_q}$	12.00	6.00	14.00	17.00	6.00	4.00	90.00	93.00	42.00	35.00
{LG, WG}	$ASR_{@once}$	0.73	0.27	4.33	6.67	0.60	1.80	13.75	27.00	3.75	2.50
	$ASR_{@max_q}$	4.00	3.00	7.00	11.00	2.00	**2.00**	56.00	55.00	11.00	9.00
{LG, XG}	$ASR_{@once}$	0.53	1.07	5.67	9.33	0.20	1.60	35.25	73.92	10.25	10.75
	$ASR_{@max_q}$	4.00	3.00	10.00	15.00	1.00	**2.00**	97.00	97.00	27.00	33.00
{WG, XG}	$ASR_{@once}$	4.27	4.40	8.00	11.67	3.00	2.80	15.50	27.08	8.25	9.50
	$ASR_{@max_q}$	15.00	12.00	14.00	19.00	7.00	5.00	65.00	55.00	25.00	30.00
{WG, GR}	$ASR_{@once}$	2.67	3.33	6.33	7.67	3.00	2.40	14.83	28.33	1.75	3.00
	$ASR_{@max_q}$	12.00	5.00	11.00	11.00	6.00	5.00	60.00	56.00	7.00	9.00
{LG, WG, XG}	$ASR_{@once}$	0.40	0.20	3.67	5.67	**0.00**	1.60	13.75	25.83	3.25	2.50
	$ASR_{@max_q}$	2.00	3.00	6.00	10.00	**0.00**	**2.00**	56.00	53.00	10.00	9.00
{LG, WG, GR}	$ASR_{@once}$	0.40	**0.07**	3.67	4.67	0.20	1.40	**13.50**	26.00	1.00	**0.75**
	$ASR_{@max_q}$	3.00	**1.00**	7.00	7.00	1.00	**2.00**	**53.00**	51.00	4.00	**2.00**
{LG, XG, GR}	$ASR_{@once}$	**0.27**	**0.07**	3.33	5.00	**0.00**	**1.20**	28.67	56.50	2.50	2.50
	$ASR_{@max_q}$	**1.00**	**1.00**	6.00	7.00	**0.00**	**2.00**	85.00	89.00	9.00	8.00
{WG, XG, GR}	$ASR_{@once}$	1.73	3.20	6.00	7.33	2.00	2.00	14.83	25.58	1.25	2.50
	$ASR_{@max_q}$	7.00	5.00	10.00	11.00	5.00	4.00	60.00	51.00	5.00	7.00
{LG, WG, XG, GR}	$ASR_{@once}$	**0.27**	**0.07**	**3.00**	**4.33**	**0.00**	**1.20**	**13.50**	**24.92**	**0.75**	**0.75**
	$ASR_{@max_q}$	**1.00**	**1.00**	**5.00**	**6.00**	**0.00**	**2.00**	**53.00**	**50.00**	**3.00**	**2.00**

Table 16: (P3) ASR results (%) of the representative attacks against *output guard* per-stage configurations. (V: Vicuna; G: GPT-3.5; Bold: stage-wise minimum; underlined: stage-wise second smallest values)

Cross-stage defense combination	Adaptive		DSN		Refusal		SeqBreak		ReNeLLM	
	V	G	V	G	V	G	V	G	V	G
Baseline (no defense)	100.00	100.00	93.00	94.00	94.00	92.00	100.00	100.00	91.00	95.00
sec1 ({PG, WG, oss})	0.00	0.00	0.00	0.00	0.00	0.00	6.00	7.00	16.00	26.00
util1 ({PG, oss})	0.00	0.00	16.00	14.00	0.00	0.00	19.00	21.00	48.00	50.00
mem1 ({PG, WG})	0.00	0.00	3.00	3.00	0.00	0.00	9.00	10.00	43.00	57.00
sec2 (ICD ⇒ SR)	100.00	22.00	39.00	0.00	85.00	55.00	100.00	98.00	90.00	80.00
util2 (SR)	100.00	96.00	59.00	3.00	82.00	69.00	96.00	98.00	92.00	94.00
sec3 ({LG, WG, GR})	3.00	1.00	7.00	7.00	1.00	2.00	53.00	51.00	4.00	2.00
util3 ({LG, WG})	4.00	3.00	7.00	11.00	2.00	2.00	56.00	55.00	11.00	9.00
util1 ⇒ **util3**	3.00	3.00	0.00	0.00	2.00	1.00	20.00	16.00	4.00	4.00
util1 ⇒ **util2**	0.00	0.00	9.00	2.00	0.00	0.00	17.00	16.00	42.00	50.00
util1 ⇒ **sec3**	2.00	1.00	0.00	0.00	1.00	1.00	19.00	15.00	1.00	1.00
util2 ⇒ **util3**	10.00	4.00	6.00	2.00	5.00	3.00	56.00	48.00	11.00	15.00
util1 ⇒ **util2** ⇒ **sec3**	0.00	0.00	1.00	1.00	0.00	0.00	5.00	6.00	5.00	7.00
util1 ⇒ **util2** ⇒ **util3**	0.00	0.00	2.00	0.00	0.00	0.00	9.00	6.00	7.00	8.00
mem1 ⇒ **util2** ⇒ **sec3**	0.00	0.00	0.00	0.00	0.00	0.00	6.00	7.00	2.00	3.00
sec1 ⇒ **util2** ⇒ **util3**	0.00	0.00	1.00	0.00	0.00	0.00	4.00	6.00	4.00	5.00

Table 17: (P4) $ASR_{@max_q}$ results (%) of the representative attacks against cross-stage defense combinations. (V: Vicuna; G: GPT-3.5)

Cross-stage defense combination	$\overline{\%\downarrow ASR}$	$\overline{\%\uparrow U}$	$\lvert\Theta\rvert$ **(B)**	ΔT **(s)**
sec1 ({PG, WG, oss})	94.2	-9.9	27.1	0.27
util1 ({PG, oss})	82.3	-7.0	20.1	0.23
mem1 ({PG, WG})	86.7	-9.6	7.1	-0.02
sec2 (ICD ⇒ SR)	30.5	-13.9	0.0	-0.88
util2 (SR)	18.1	-6.7	0.0	-0.51
sec3 ({LG, WG, GR})	86.7	-8.9	23.0	1.85
util3 ({LG, WG})	83.7	-7.4	15.0	0.13
util1 ⇒ **util3**	94.6	-7.8	35.1	0.48
util1 ⇒ **util2**	85.6	-8.1	20.1	-0.27
util1 ⇒ **sec3**	95.9	-8.2	43.1	2.60
util2 ⇒ **util3**	83.7	-9.2	15.0	-0.35
util1 ⇒ **util2** ⇒ **sec3**	97.4	-9.2	43.1	2.16
util1 ⇒ **util2** ⇒ **util3**	96.7	-10.2	35.1	-0.03
mem1 ⇒ **util2** ⇒ **sec3**	98.2	-14.3	30.1	1.97
sec1 ⇒ **util2** ⇒ **util3**	97.9	-14.9	42.1	0.09

Table 18: (P4) Security, utility and efficiency results of cross-stage defense combinations.

Defense	**Adaptive**	**DSN**	**Refusal**	**SeqBreak**	**ReNeLLM**	$\overline{\%\downarrow ASR}$	$Utility$	$\overline{\%\uparrow U}$	FPR
GPT-5.4-mini									
Baseline (no defense)	74.00	2.00	5.00	7.50	52.00	0.0	84.2	0.0	0.0
util1 ⇒ **util2** ⇒ **sec3** ({PG,OSS}⇒[SR]⇒{LG,WG,GR})	0.00	0.00	0.00	0.00	0.00	100.0	85.4	1.4	7.6
util1 ⇒ **util3** ({PG,OSS}⇒{LG,WG})	0.00	0.50	0.00	1.00	0.00	92.3	82.6	-1.9	8.0
mem1 ({PG,WG})	0.00	0.00	0.00	0.50	21.50	90.4	78.9	-6.3	1.2
gemini-3.1-flash-lite									
Baseline (no defense)	96.00	28.50	66.50	99.50	92.00	0.0	81.2	0.0	0.0
util1 ⇒ **util2** ⇒ **sec3** ({PG,OSS}⇒[SR]⇒{LG,WG,GR})	0.00	0.00	0.00	16.50	0.00	96.3	80.8	-0.5	8.0
util1 ⇒ **util3** ({PG,OSS}⇒{LG,WG})	0.00	0.00	0.00	13.00	6.00	96.1	79.2	-2.5	7.2
mem1 ({PG,WG})	0.00	0.00	0.00	14.00	57.00	84.8	73.0	-10.1	1.2

Table 19: ASR (%), Utility and FPR results of generalizability experiment on GPT-5.4-mini, gemini-3.1-flash-lite, HarmBench and XSTest.