REDLOOP: MAINTAINING REUSABLE ATTACK STRATEGIES FOR MULTI-ROUND LLM SAFETY TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-round safety training must reduce harmful answers without encouraging refusal of benign queries. As training changes the defender, the attack strategies a red-teamer has stored can stop working, and its library can miss strategies that would expose the updated defender; retraining the red-teamer or rechecking every stored strategy is costly. We present RedLoop, a multi-round safety-training framework that keeps the attack-generation model frozen and adapts the red-teamer, REDSCOUT, through a persistent library of reusable attack strategies, natural-language instructions for generating adversarial prompts. To keep the library useful without rechecking every strategy, *fitness propagation* estimates how well previous-round strategies still work from a sample of the current round's attacks and reuses the highest-ranked ones in the next training batch. Alongside strategies distilled from successful attacks, *novelty injection* contributes strategies that are distinct from the library and succeed against the defender, and later attack steps draw on them more heavily. The defender is trained with GRPO under a reward that penalizes refusal of benign queries and of benign queries carrying adversarial framing. On Nemotron-Nano-4B, RedLoop reduces HarmBench attack success from 21.12% to 4.04% and XSTest over-refusal from 68.0% to 51.2%, while IFEval and MMLU stay within one point of the base model; removing fitness propagation raises HarmBench attack success to 10.56% and removing novelty injection raises XSTest over-refusal to 66.8%. Safety gains also extend to Llama-3.1-8B and OLMO-3-7B defenders.

1 INTRODUCTION

Safety training must reduce harmful answers without teaching a language model to refuse benign queries. Automated red-teaming supplies adversarial prompts that expose failures and can guide further training, but the target of that search changes with every defender update. Previously successful attacks may stop working, while vulnerabilities outside the red-teamer's stored strategies may remain. Repeated safety training therefore needs a red-teamer that can reuse prior strategies, reassess them against the current defender, and discover failures that its existing attacks miss.

We study this problem in a multi-round setting: a red-teamer generates adversarial prompts, a frozen judge scores the defender's answers, and the defender is retrained on the resulting data together with benign examples. The attack-generation model remains frozen throughout. Its adaptive state is a persistent library of reusable attack strategies: natural-language instructions for constructing prompts, rather than a separate model update each round. For example, a strategy can specify a fictional-dialogue framing; given that strategy and a query, the generator produces a query-specific prompt. Each strategy has a *fitness* estimate, defined as the mean severity score of the defender's answers to its instantiated attacks, that guides subsequent selection.

Prior work adapts red-teamers in two ways: retraining the red-teamer alongside the defender (Ge et al., 2024; Liu et al., 2025a; Wen et al., 2026; Wang et al., 2026b), or holding it fixed and rerunning search against each updated defender with concrete prompts (Mazeika et al., 2024; Shi et al., 2026). Strategy libraries store reusable attack knowledge, and AutoDAN-Turbo additionally studies transfer and library updates from new attack logs (Liu et al., 2025b). Our focus is the maintenance problem that arises when reusable strategies guide repeated training of the generative defender. The defended model is retrained each round on the red-teamer's own attacks, so stored scores drift and remeasuring the whole library is costly. Two challenges matter. First, *stored fitness can become stale*:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

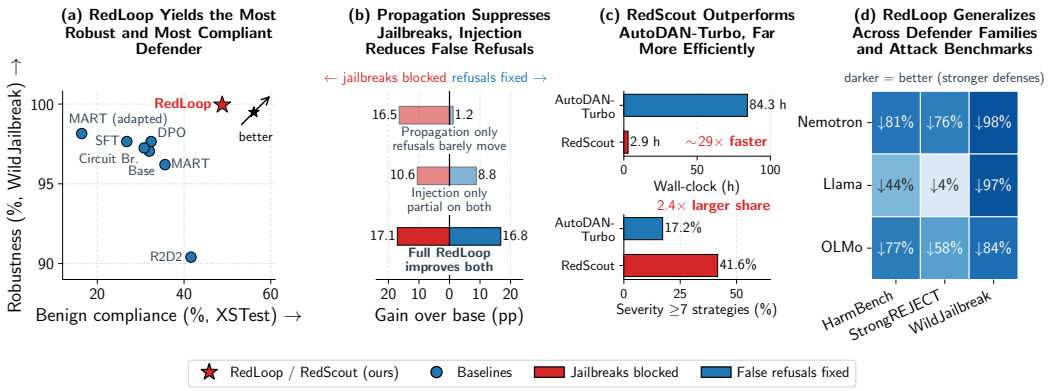


Figure 1: **Key findings.** **(a)** On Nemotron-Nano-4B, the RedLoop-trained defender is the most robust and most compliant of all methods (100−WildJailbreak attack success and 100−XSTest over-refusal). **(b)** Improvement over the base model in HarmBench attack success and XSTest over-refusal. **(c)** At a matched attack budget, REDSCOUT is faster and stores a larger share of high-severity strategies than AutoDAN-Turbo. **(d)** Relative reduction in attack success across three defender families and three benchmarks.

each stored score was measured against the previous defender, so a defender update can change the ranking of strategies, but re-evaluating every entry requires defender answers and judge evaluations at a cost that grows with the library. Second, *recalibration alone cannot repair coverage gaps*: even accurate scores cannot reveal an attack strategy absent from the library. These challenges must be addressed while preserving benign helpfulness; lower attack success is insufficient if it comes from broader refusal. Detailed comparisons with prior systems appear in Section 3 and Appendix A. The problem is to maintain the red-teamer's strategy library across defender updates. The question we study is:

> Can a persistent strategy library keep a red-teamer with a frozen attack-generation model effective as the defender is repeatedly updated, without re-evaluating the entire library?

RedLoop adapts the red-teamer while keeping the attack-generation model's weights fixed. Fitness propagation uses a coverage-balanced probe of the current round's scored attacks to rank previous-round attacks for replay, fitting a Gaussian process over strategy-text embeddings under the assumption that nearby embeddings have related scores against the current defender. *Novelty injection* addresses coverage gaps by proposing additional strategies and admitting candidates only when they are separated from the stored library in embedding space and demonstrate attack success against the updated defender. To preserve helpfulness, the defender is trained with GRPO under a class-conditional reward that penalizes refusal of benign queries, including framed-benign queries.

Evaluation covers six training rounds using eight standard benchmarks for attack success, over-refusal, and capability, together with component ablations and experiments on Llama-3.1-8B and OLMO-3-7B defenders. On Nemotron-Nano-4B, the full configuration reduces HarmBench attack success from 21.12% to 4.04% and XSTest over-refusal from 68.0% to 51.2%, while IFEval and MMLU remain within one point of the untrained model (Table 1). Figure 1 summarizes the main defender, ablation, and red-teamer comparisons.

Our contributions are:

- We introduce REDSCOUT, a red-teamer that keeps the attack-generation model fixed and adapts through a persistent library of reusable strategies. It selects strategies by fitness, an exploration bonus, and a novelty score. At a matched attack budget, REDSCOUT achieves higher measured attack-success rates and a larger share of high-severity strategies than AutoDAN-Turbo, with shorter wall-clock time under the reported configurations (Appendix B.1).
- *Fitness propagation* and *novelty injection* address the strategy-library maintenance problem that arises when the defender is retrained, targeting stale fitness estimates and missing strategy coverage, respectively.
- The resulting safety–helpfulness tradeoff is evaluated, the two maintenance operations are ablated, and the framework is tested across three defender families.

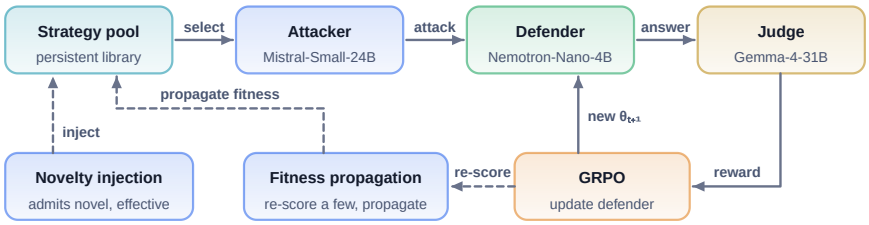


Figure 3: One co-evolutionary round of RedLoop. REDSCOUT generates attacks against the current defender; a current-round probe ranks previous-round attacks for replay; GRPO updates the defender; novelty injection tests candidates against the updated defender.

2 METHOD

RedLoop is a multi-round safety-training framework with three model roles: the defender model π_{θ_t}, whose weights we update; the red-teamer REDSCOUT, which pairs a frozen attack-generation model with an attack strategy library $\mathcal{L}_t$ that changes at every round; and a frozen judge model that scores defender answers. A run spans T rounds. At round t, the state is the pair $(\pi_{\theta_t}, \mathcal{L}_t)$.

In each round, REDSCOUT selects strategies from $\mathcal{L}_t$ and conditions the frozen model on the selected strategy and a harmful query to produce an adversarial prompt, which the defender answers and the judge scores. Figure 2 compares held-out validation reward for two attack distributions. The fixed-source subset contains examples from WildJailbreak, WildGuard, Aegis, Coconot, and UltraSafety, with corresponding training examples drawn from their training subsets. The adaptive subset contains held-out attacks from the RedScout distribution. Fixed-source validation remains near the upper reward band, whereas adaptive-attack validation begins lower and increases as defender training proceeds. The scored adversarial prompts and benign controls form a training batch, on which Group Relative Policy Optimization (GRPO) updates the defender, $\pi_{\theta_t} \to \pi_{\theta_{t+1}}$. Because a strategy's fitness is defined against a particular defender, scores stored from earlier rounds may no longer rank attacks for the current defender. RedLoop therefore maintains the library through two operations. *Fitness propagation* uses a coverage-balanced probe of the current round's scored attacks to rank previous-round attacks for replay in the next training batch; *novelty injection* tests candidate strategies against the updated defender and admits those that satisfy the separation and success conditions. We deploy only π_{θ_T}, and we use the red-teamer, the judge, and the library only during training. Algorithm 1 (Appendix C) summarises one run.

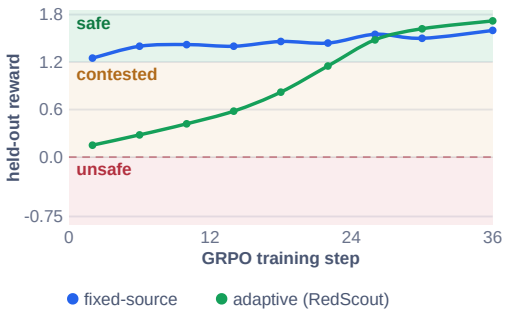


Figure 2: Mean held-out validation reward during GRPO for fixed-source and generated-attack training. Each curve is evaluated on held-out examples from its own training distribution.

2.1 RED-TEAM GENERATION WITH REDSCOUT

A prompt records one attack on one query; a strategy records how to construct related attacks on other queries. At round t, REDSCOUT operates on the strategy library $\mathcal{L}_t = \{\tau_1, \ldots, \tau_{N_t}\}$ of N_t attack strategies, where an *attack strategy* τ is reusable natural-language text instructing the attack-generation model how to construct adversarial prompts. Given harmful query q and selected strategy τ_i, the frozen model receives both and generates a query-specific adversarial prompt x, which the defender answers.

The frozen judge assigns defender answer y a severity score $g(x, y) \in [1, 10]$. For one adversarial prompt x, let $\mathrm{score}(x)$ be the median over the R sampled answers of the mean score over the K judge replicates. We define the directly observed fitness of τ_i as its mean severity,

$$f_i^{\mathrm{obs}} = \frac{1}{|\mathcal{A}_i|} \sum_{x \in \mathcal{A}_i} \mathrm{score}(x) \in [1, 10], \tag{1}$$

where $\mathcal{A}_i$ is the set of prompts generated from τ_i. Entries without a current measurement are ranked by the propagation acquisition of Section 2.3.1.

Let $\mathcal{N}_k(i)$ denote the $k = 5$ nearest strategies to τ_i under cosine similarity and ν_i their mean cosine distance from it, recomputed at every generation step. Selection adds a count-based exploration term and this novelty term to the stored fitness, sampling τ_i with probability

$$a_i = f_i + c\sqrt{\frac{\log N}{\max(n_i,1)}} + \lambda\nu_i, \quad N = 1 + \textstyle\sum_j n_j, \quad P(i) = \frac{\phi_0}{N_t} + (1-\phi_0)\frac{\exp(a_i/T_s)}{\sum_j \exp(a_j/T_s)}. \tag{2}$$

The terms favor higher-fitness strategies, less-used strategies, and strategies distant from the library, with a floor that keeps every strategy selectable; Appendix C gives the settings and the stored fields.

Within each round, *strategy distillation* adds reusable strategies extracted from successful prompts, with redundancy filtering before admission (Appendix C).

Redundancy filtering prevents near-duplicate entries from entering the library (Appendix B.1). We never delete stored strategies, and the adversarial prompts generated during the step also form the defender's training batch.

2.2 Defender update with a class-conditional reward

Each round then updates the defender on a training batch with a *dynamic* partition of the current round's adversarial prompts, a *replay* partition of prompts from earlier rounds to keep identified failures in the training signal, and a *benign* partition at a share $\beta_{\mathrm{B}}{=}20\%$, including benign prompts carrying the surface framing of an adversarial prompt.

We condition the reward on whether the prompt is harmful or benign. The judge scores each answer on a 1–5 rubric, with an answer-safety score $s(x, y)$ on a harmful prompt and a helpfulness score $h(x, y)$ on a benign one; $w : \{1, 2, 3, 4, 5\} \mapsto \{-0.75, -0.25, 0, 0.75, 1.5\}$ maps scores to rewards, and $\mathbf{1}[\text{comply}] - \mathbf{1}[\text{refuse}]$ is the compliance margin of Section 4.1. We define the reward as

$$r(x,y) = \begin{cases} w(s(x,y)), & x \text{ harmful}, \\ w(h(x,y)) + \rho_{\mathrm{b}}\big(\mathbf{1}[\text{comply}] - \mathbf{1}[\text{refuse}]\big), & x \text{ benign}, \end{cases} \tag{3}$$

With $\rho_{\mathrm{b}} = 0.5$, the benign case rewards a helpful answer and penalizes a refusal, which discourages the defender from refusing a benign prompt, including one whose surface framing resembles an adversarial prompt.

We update the defender each round with GRPO (Shao et al., 2024), standardizing rewards across G answers per prompt and applying a KL penalty to the round's starting checkpoint; the objective, sampling, and optimizer settings are in Appendix C. The update produces $\pi_{\theta_{t+1}}$, against which the next round's attacks are generated.

2.3 Cross-round strategy-library update

A round ends with the defender updated from π_{θ_t} to $\pi_{\theta_{t+1}}$, while the library still stores scores obtained against π_{θ_t}. Fitness propagation uses a coverage-balanced probe of the current round's scored attacks to rank the previous round's attacks for replay in the training batch that produced $\pi_{\theta_{t+1}}$, and novelty injection tests candidate strategies against $\pi_{\theta_{t+1}}$ before the next red-teaming phase.

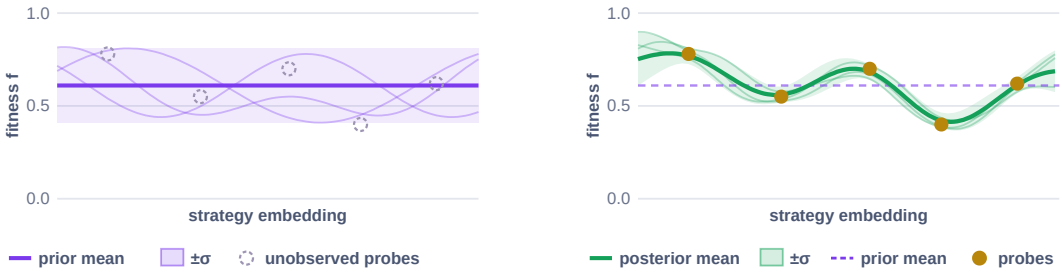
2.3.1 Fitness propagation

Fitness propagation instead reserves a 15% coverage-balanced probe from the current round's attacks, which the red-teamer scores against π_{θ_t} as part of generation. The probe is disjoint from the defender-training and validation records, and contains at least one attack for each represented strategy. Because the previous round's records retain scores measured against $\pi_{\theta_{t-1}}$, fitness propagation uses the probe observations to rank those records for the replay portion of the training batch (Appendix C).

Our estimator, GP-UCB, models the current score of a previous-round strategy as a function of its strategy-text embedding with the cosine radial-basis kernel

$$k(\mathbf{a}, \mathbf{b}) = \exp\Big(\tfrac{\cos(\mathbf{a},\mathbf{b})-1}{\ell}\Big), \qquad K_{\mathcal{S}} = k(\mathcal{S}_t, \mathcal{S}_t) + \sigma_n^2 I. \tag{4}$$

For a previous-round strategy with embedding $\mathbf{v}$, the GP posterior mean estimates a current score, and the mean plus β times the standard deviation, with $\beta = 1$, ranks previous-round attacks for replay. Appendix C gives the posterior equations and the fixed hyperparameters.



(a) Prior mean, before conditioning.

(b) Posterior, after the probes.

Figure 4: GP-UCB prior and posterior, with severity scaled to $[0, 1]$ for display. The prior mean is constant at the probe mean; after conditioning, the posterior mean follows the probe scores and uncertainty contracts nearby.

2.3.2 Novelty injection

Once the first round has produced a scored record, novelty injection runs immediately before each red-teaming phase and augments the library that phase starts from. Selection can then draw on candidates the library lacked. It complements strategy distillation, which adds strategies within the phase, and admitted candidates receive priority in selection. An *injection method* specifies how candidate strategy text is proposed; we summarize three here and compare the full set in Appendix B.3.

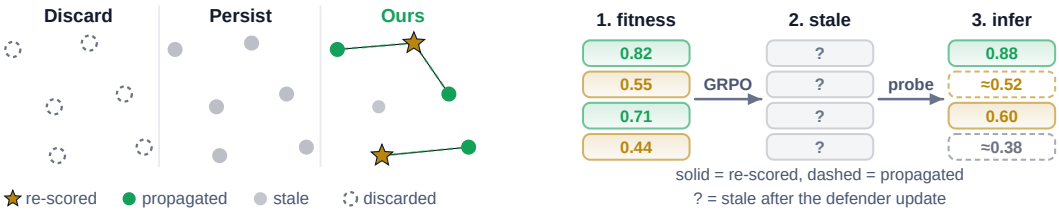
- *Concrete-template injection (ATPL)* converts a successful adversarial prompt into a reusable template, retaining its framing, role, and wording while replacing the harmful content with a placeholder.
- *Defender-failure-domain injection (DFDD)* embeds the prompts on which the updated defender complied, clusters those embeddings, and proposes a strategy covering each cluster's shared mechanism.
- *Manifold-void exploration (MVE)* finds directions in strategy-embedding space with low maximum cosine similarity to the stored strategies and proposes strategies for those regions.

We use *concrete-template injection* (ATPL) in the full configuration. Each method first sends its candidates through the coherence filter of Appendix B.3.

All injection methods share the same two admission conditions. For candidate strategy τ_c with embedding $\mathbf{e}_c$, let Q_c denote a fixed set of harmful admission queries and $\mathcal{X}(\tau_c, q)$ the adversarial prompts generated from τ_c and query q. We admit τ_c only if

$$\max_{\tau_j \in \mathcal{L}_t} \cos(\mathbf{e}_c, \mathbf{e}_j) \le 1 - \delta, \qquad \frac{1}{|Q_c|} \sum_{q \in Q_c} \mathbf{1}\Big[\max_{x \in \mathcal{X}(\tau_c, q)} \text{score}(x) \ge \gamma\Big] \ge q_{\min}. \tag{5}$$

The first condition requires separation from every stored strategy in strategy-embedding space and depends on the library alone; the second requires the candidate to reach severity γ on at least a fraction $q_{\min}$ of its admission queries when evaluated against $\pi_{\theta_{t+1}}$. The two conditions test different properties. We set $q_{\min}$ such that one successful query suffices, with $|Q_c| = 80$ and $\gamma = 5$. Admitted candidates join the library before red-teaming begins and are therefore selectable during the round's generation steps.



(a) Pool policy after a GRPO update.

(b) A current-round probe ranks previous-round attacks.

Figure 5: Fitness propagation. Previous-round scores no longer rank attacks for the current defender; the current-round probe provides observations, and the GP ranks previous-round attacks for replay.

3 Related Work

Automated attack search. GCG (Zou et al., 2023) optimizes adversarial suffixes using target gradients, while AutoDAN (Liu et al., 2024) searches fluent prompts with a genetic algorithm. PAIR (Chao et al., 2025), TAP (Mehrotra et al., 2024), and GPTFuzzer (Yu et al., 2023) refine or mutate prompts through black-box queries. These methods organize search over prompts or suffixes, which can be reused across queries. Our focus is how stored strategies should guide subsequent search when defender training changes which remain useful: RedLoop retains reusable strategies and reuses them across defender updates.

Reusable strategy libraries. AutoDAN-Turbo (Liu et al., 2025b) distills attacks into reusable strategies, retrieves them for later queries, and studies library transfer and updates from new attack logs. ASTRA (Liu et al., 2026) organizes strategies into score tiers, while MemoAttack (Zhang et al., 2026b) manages a skill lifecycle with posterior-guided selection. RedLoop builds on this explicit-memory representation. The additional problem is to keep a strategy library useful through repeated training of the generative defender, where stored scores no longer rank attacks for the current defender. Fitness propagation supplies a current-defender ranking of previous-round attacks for replay; novelty injection complements reuse by testing additional candidates against the updated defender.

Joint attacker–defender training. MART (Ge et al., 2024), Self-RedTeam (Liu et al., 2025a), and MAGIC (Wen et al., 2026) adapt attacks through updates to the attack-generation policy. TriPlay-RL (Tan et al., 2026) adds a trained evaluator; CHASE (Markasserithodi et al., 2026) and Lifelong Safety Alignment (Wang et al., 2026b) combine attacker learning with retained attack data; GPO (Zheng et al., 2025) formulates adaptation as a two-player game. Several also keep prompts or buffers, so the distinction is not the presence of memory. In RedLoop, the attack-generation model remains frozen: adaptation operates through strategy selection, current-defender ranking for replay, and library expansion, while the defender's parameters are updated.

Persistent attack state under changing defenses. R2D2 (Mazeika et al., 2024) re-optimizes sampled concrete test cases against the current model; SSP (Wang et al., 2026a) re-evaluates replay items; and CEMMA (Shi et al., 2026) evolves an attack population and accumulates successful examples for defender training. EvoSafety (Zhang et al., 2026c) and DARWIN (Qi et al., 2026) retain skills or strategies while training a separate auxiliary defense or guardrail. Thus, retaining and revisiting attack state under a changing defense is established. Among these reviewed approaches, our contribution is the combination of a growing library of query-instantiable strategies, repeated training of the attacked generative model, and a current-defender ranking of previous-round attacks for replay. The relevant comparison concerns what is retained, which model changes, and how stored strategies is reassessed. Appendix A provides the per-system details.

Safety without over-refusal. MAGIC (Wen et al., 2026) penalizes refusal of benign queries and rewrites benign seeds into adversarial-benign prompts. WALTZ RL (Zhang et al., 2026a) uses cooperative feedback to improve safety and helpfulness; Safe RLHF (Dai et al., 2024) balances them through a constrained objective; and Safe-completions (Yuan et al., 2025) emphasizes output safety beyond a binary refusal decision. These works motivate the benign side of our training objective. RedLoop uses a class-conditional reward and framed-benign queries within the evolving attack curriculum. Our contribution concerns maintaining that curriculum as the defender changes; the benign reward follows the established goal of improving safety while preserving helpfulness. Appendix A.5 discusses representation-based defenses and inference-time filters, and Appendix A.7 describes the evaluation resources.

4 Experiments

Our evaluation connects the strategy-maintenance design to the behavior of the trained defender. We ask whether RedLoop improves safety while preserving helpfulness and capability (RQ1), what each maintenance operation contributes (RQ2), how its red-teamer compares with AutoDAN-Turbo (RQ3), and whether the gains persist across defender families (RQ4).

4.1 Setup

Models and training. We train Llama-3.1-Nemotron-Nano-4B-v1.1 as the defender with GRPO through VERL (Sheng et al., 2025) on LoRA adapters of rank 64. The red-teamer uses Mistral-Small-24B-Instruct as its attack-generation model and the judge uses Gemma-4-31B-it; both remain

frozen and distinct from the defender. We use Mistral-Small-24B rather than a larger and more heavily safety-aligned model because it red-teams more effectively, as Section B.1 reports. The judge therefore never scores an answer it produced or a prompt it built. Within a run, only the defender's weights and the strategy library change. A run consists of $T{=}6$ rounds, each with a red-teaming phase followed by a defender-training (GRPO) phase. One standalone *seed phase* of 200 generation steps against the base defender yields the initial library, and each later round runs 10 generation steps (Appendix C gives the per-step query and attack counts).

Comparisons. RedLoop (full) applies fitness propagation and strategy injection together with the defender update of Section 2.2. We evaluate four component configurations: *propagation-only* omits cross-round strategy injection, *injection-only* omits fitness propagation. Strategies not directly re-evaluated retain their previous-round fitness, while *strategy-cluster* and *softmax-cosine* replace the propagation estimator while leaving selection, injection, and defender training unchanged. The reported full configuration uses ATPL, and Appendix B.3 compares seven alternative injection methods. We use a benign share $\beta_{\mathrm{B}}{=}20\%$, the fraction of each defender-training batch assigned to benign and framed-benign queries. We compare against baselines evaluated under the same protocol: the untrained base, SFT on refusal answers, DPO (Rafailov et al., 2023), MART (Ge et al., 2024), Circuit Breakers (Zou et al., 2024), and R2D2 (Mazeika et al., 2024). Each baseline runs at the same defender scale and on the same frozen red-teamer, judge, and query sets as RedLoop, so the comparison reflects the training method rather than the model scale or the supply of adversarial prompts. We also report a *MART (adapted)* configuration that replaces MART's red-teamer with REDSCOUT while retaining MART's training method and neither fitness propagation nor strategy injection, which separates the training method from the red-teamer that generates its adversarial data.

Data. The red-teamer receives harmful queries drawn from the AutoDAN-Turbo corpus (Liu et al., 2025b); a fixed set of 128 queries is used in every round. The benign queries come from the WildGuard and Aegis training splits, where WildGuard also contributes *framed-benign queries*, benign content carrying the surface framing of an adversarial prompt. Training and evaluation use disjoint query sets, since a hash of the query text assigns each query to one side for the whole run.

Evaluation. Each standard benchmark uses a benchmark-native scorer. HarmBench (Mazeika et al., 2024), StrongREJECT (Souly et al., 2024), and WildJailbreak (Jiang et al., 2024) provide attack-success metrics; XSTest (Röttger et al., 2024), OR-Bench (Cui et al., 2024), and PHTest (An et al., 2025) provide over-refusal metrics; IFEval (Zhou et al., 2023) and MMLU (Hendrycks et al., 2021) provide capability metrics. For the mechanism and transfer analyses, we generate defender answers on held-out queries and score them with the frozen training judge; Appendix D defines the resulting held-out ASR. Held-out ASR counts harmful answers with answer-safety scores at most 2 on the 1–5 scale; severity-threshold success uses the separate 1–10 attack-severity scale and the stated threshold. These rates are not interchangeable; Appendix D defines their aggregation, and each table identifies its scoring protocol. Lower attack success and over-refusal are better; higher capability scores are better. Appendix D also specifies confidence intervals and paired comparisons.

The results below evaluate the trained defender at two levels. The standard benchmarks, each scored by a benchmark-native scorer, provide the primary evidence for defender behavior on held-out queries. The mechanism and transfer analyses use held-out queries scored by the frozen judge, with one scorer applied across the compared configurations. Each table identifies the applicable protocol.

4.2 RQ1: CAN SAFETY IMPROVE WHILE PRESERVING HELPFULNESS AND CAPABILITY?

The frozen training judge supplies the reward signal. An improvement measured only by that judge could reflect judge-specific optimization rather than a defender that generalizes to unseen attacks. The standard benchmarks avoid this dependence because none is scored by the training judge. Table 1 reports eight benchmarks, and the defender trained with the full configuration improves on the untrained defender on every attack-success and over-refusal metric: HarmBench attack success falls from 21.12% to 4.04% and XSTest over-refusal from 68.0% to 51.2%. This paired result lowers attack success without increasing refusal; IFEval changes from 68.76 to 68.02 and MMLU from 41.90 to 41.66. The training baselines trade safety against capability at different operating points; R2D2 reaches 2.48% HarmBench on its native benchmark, but its IFEval score is 34.75.

Table 1: **Standard benchmarks and capability.** Each metric is computed by that benchmark's own scorer, separate from our LLM judge. Lower is better for attack success and over-refusal; higher is better for IFEval and MMLU. Bold marks the best value in each column; ties are all bold.

	Attack success ↓			Over-refusal ↓			Capability ↑	
Method	Harm Bench	Strong REJECT	Wild Jailbreak	XSTest	OR-Bench	PHTest	IFEval	MMLU
RedLoop (ours)								
Full	4.04	**0.0108**	**0.05**	**51.2**	**97.27**	**56.33**	68.02	41.66
Propagation-only	4.66	0.0157	**0.05**	66.8	97.95	59.22	68.76	40.91
Injection-only	10.56	0.0408	0.45	59.2	98.94	63.46	68.39	39.78
Baselines								
Base (untrained)	21.12	0.0454	2.95	68.0	98.79	64.61	68.76	41.90
SFT	18.01	0.0303	2.35	73.2	99.17	66.15	**70.43**	36.63
DPO (Rafailov et al., 2023)	18.94	0.0407	2.35	67.6	98.64	65.29	66.91	41.49
MART (Ge et al., 2024)	19.57	0.0564	3.80	64.4	99.24	58.88	67.47	37.02
MART (adapted)	12.73	0.0425	1.85	83.6	99.55	69.04	57.86	26.63
Circuit Breakers (Zou et al., 2024)	22.05	0.0366	2.75	69.2	99.01	64.52	68.95	41.75
R2D2 (Mazeika et al., 2024)	**2.48**[†]	0.0163	9.60	58.4	97.88	69.28	34.75	**45.81**

[†]HarmBench is R2D2's native benchmark for training and evaluation.

Table 2: **Component ablation.** Each section varies one component of the full configuration, evaluated on the standard benchmarks under the same scorers as Table 1. Bold marks the best value in each column within each section; ties are all bold.

	Attack success ↓			Over-refusal ↓			Capability ↑	
Configuration	Harm Bench	Strong REJECT	Wild Jailbreak	XSTest	OR-Bench	PHTest	IFEval	MMLU
Fitness propagation								
GP-UCB	**4.04**	**0.0108**	0.05	**51.2**	**97.27**	**56.33**	68.02	**41.66**
Softmax-cosine	6.83	0.0176	**0.00**	78.4	99.39	82.09	**68.39**	40.77
Strategy-cluster	5.28	0.0156	**0.00**	77.6	99.32	80.07	65.80	41.33
None (injection-only)	10.56	0.0408	0.45	59.2	98.94	63.46	**68.39**	39.78
Strategy injection								
ATPL (concrete-template)	**4.04**	**0.0108**	**0.05**	51.2	97.27	56.33	68.02	41.66
DFDD (defender-failure-domain)	14.25	0.0603	1.00	47.2	**96.82**	48.68	68.58	**42.35**
MVE (manifold-void exploration)	15.50	0.0569	1.20	**41.6**	97.19	**47.86**	**69.87**	41.72
None (propagation-only)	4.66	0.0157	**0.05**	66.8	97.95	59.22	68.76	40.91

4.3 RQ2: What does each maintenance operation contribute?

The two library-maintenance operations are the central framework components. We isolate each operation by varying one component of the full configuration at a time and evaluating each configuration with the same benchmark-native scorers used in Table 1 (Table 2). Removing fitness propagation raises HarmBench attack success from 4.04% to 10.56%, whereas removing novelty injection raises XSTest over-refusal from 51.2% to 66.8%. The two operations therefore govern different axes. Estimator and proposal choices also change this tradeoff: GP-UCB yields lower HarmBench attack success and XSTest over-refusal than the softmax-cosine and strategy-cluster alternatives in the reported configurations, and among injection methods ATPL yields lower HarmBench attack success than DFDD and MVE, while those alternatives yield lower XSTest over-refusal. Appendix B.2 gives the propagation equations and direct-scoring accounting.

4.4 RQ3: How does the RedScout red-teamer compare with AutoDAN-Turbo?

Because the red-teamer controls strategy selection and library growth, we compare it directly against the closest prior system. At a matched evaluation budget, with our red-teamer run for 200 generation steps over the same 128 harmful queries, it stores a smaller library with no measured duplicate strategies than AutoDAN-Turbo (0.0% versus 68.7%) and has a larger fraction of strategies at severity ≥ 7 (41.6% versus 17.2%). Under identical single-answer scoring it exceeds AutoDAN-Turbo on every attack-success measure and reaches the severity-≥ 7 strategies at 33.2 per hour against

Table 3: Directed attack transfer compared with native performance. Native ASR is the score obtained by attacks collected on the evaluated defender; Δ is transfer ASR minus native ASR for that defender.

Attacks collected on	Evaluated on	Native ASR	Transfer ASR	Δ
Nemotron-Nano-4B	OLMO-3-7B	41.7	44.1	+2.4
OLMO-3-7B	Nemotron-Nano-4B	43.1	38.4	-4.7
Nemotron-Nano-4B	Llama-3.1-8B	64.3	65.8	+1.5
Llama-3.1-8B	Nemotron-Nano-4B	43.1	36.4	-6.7
Nemotron-Nano-4B	Gemma-4-31B-it	27.9	19.3	-8.6
Gemma-4-31B-it	Nemotron-Nano-4B	43.1	44.2	+1.1

0.85, with 2.9 hours of wall-clock for the 200-step run against 84.3 for AutoDAN-Turbo under its reported configuration (Table 4, Appendix B.1). The gap reflects the implemented systems, since AutoDAN-Turbo issues its attack refinement serially.

4.5 REDSCOUT ATTACK TRANSFER ACROSS DEFENDERS

To evaluate whether REDSCOUT's attacks transfer across defenders, we run the same REDSCOUT with the same frozen attack-generation model against four defender model families and collect the resulting attacks. For each collection, we compare the attack success on the defender used during collection with the success of those same attacks on another defender, using held-out ASR (Table 3). The six transfer evaluations range from 19.3% to 65.8% ASR: attacks collected on Nemotron-Nano-4B reach 44.1% on OLMO-3-7B, 65.8% on Llama-3.1-8B, and 19.3% on Gemma-4-31B-it, while attacks collected on OLMO-3-7B, Llama-3.1-8B, and Gemma-4-31B-it reach 38.4%, 36.4%, and 44.2% on Nemotron-Nano-4B, respectively. Transfer remains close to native performance: across the six comparisons, the absolute gap from the corresponding native value is only 1.1–8.6 percentage points. The largest gap occurs on Gemma-4-31B-it, where attacks collected on Nemotron-Nano-4B reach 19.3% versus 27.9% for attacks collected on Gemma-4-31B-it. These results show that the collected attacks retain success on defenders not used during collection.

4.6 RQ4: DO THE GAINS PERSIST ACROSS DEFENDER FAMILIES?

Across defender families. We also apply the full framework to two additional defender families. On Llama-3.1-8B, HarmBench attack success falls from 32.30% to 18.01%, WildJailbreak from 40.35% to 1.15%, and XSTest over-refusal from 11.2% to 3.6%; MMLU decreases from 68.77 to 68.03, while OR-Bench is the one reported safety axis that does not improve, moving from 74.30% to 76.72% rejection. On OLMO-3-7B, HarmBench falls from 6.75% to 1.55% and WildJailbreak from 2.45% to 0.40%, while MMLU changes from 60.65 to 60.40. The framework therefore applies across multiple defender families.

5 LIMITATIONS

Our study trains and evaluates on a fixed set of harmful queries, with the red-teamer generating new attacks each round and the defender retrained against them. In realistic deployments, attack distributions, red-teamer capabilities, and defender requirements can shift beyond the conditions studied here, for example toward new attack modalities, deployment-time prompts, or unseen harmful content. How a persistent strategy library should adapt under such shifts, and how these results extend to larger defenders and longer training, are natural subjects for future work. Standard benchmarks are scored by benchmark-native evaluators, separate from the training judge, while the mechanism and transfer analyses use the training judge; the paper reports the two protocols separately.

6 CONCLUSION

RedLoop maintains its strategy library across six defender-training rounds through *fitness propagation* and *novelty injection* while keeping the attack-generation model's weights fixed. At the 20% benign operating point, the full configuration reduces HarmBench attack success from 21.12% to 4.04% and XSTest over-refusal from 68.0% to 51.2%, while IFEval and MMLU remain within 0.8 percentage points of the base model. The evidence covers a primary 4B defender and cross-architecture experiments with 7–8B defenders.

AI USE STATEMENT

We used generative AI for research execution and writing. For research execution, generative AI coding assistants were used for the coding of several parts of the experiment, training, and evaluation code: implementing components of the co-evolution pipeline, debugging runtime and configuration issues, and writing and running tests. A human reviewed and tested all of the code, checked the implementation against the method and the outputs. For writing, large language models were used to improve the clarity, grammar, and concision of text drafted by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts.

REFERENCES

Bang An, Sicheng Zhu, Ruiyi Zhang, Michael-Andrei Panaitescu-Liess, Yuancheng Xu, and Furong Huang. Automatic pseudo-harmful prompt generation for evaluating false refusals in large language models, 2025. URL https://arxiv.org/abs/2409.00598.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 23–42. IEEE, 2025.

Justin Cui, Wei-Lin Chiang, Ion Stoica, and Cho-Jui Hsieh. Or-bench: An over-refusal benchmark for large language models. *arXiv preprint arXiv:2405.20947*, 2024.

Juntao Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. Safe rlhf: Safe reinforcement learning from human feedback. In *International Conference on Learning Representations (ICLR)*, 2024.

Suyu Ge, Chunting Zhou, Rui Hou, Madian Khabsa, Yi-Chia Wang, Qifan Wang, Jiawei Han, and Yuning Mao. Mart: Improving llm safety with multi-round automatic red-teaming. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1927–1937, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

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

Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofar Mireshghallah, Ximing Lu, Maarten Sap, Yejin Choi, et al. Wildteaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models. *Advances in Neural Information Processing Systems*, 37:47094–47165, 2024.

Mickel Liu, Liwei Jiang, Yancheng Liang, Simon Shaolei Du, Yejin Choi, Tim Althoff, and Natasha Jaques. Chasing moving targets with online self-play reinforcement learning for safer language models. *arXiv preprint arXiv:2506.07468*, 2025a.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. Autodan: Generating stealthy jailbreak prompts on aligned large language models. In *International Conference on Learning Representations*, volume 2024, pp. 56174–56194, 2024.

Xiaogeng Liu, Peiran Li, G Edward Suh, Yevgeniy Vorobeychik, Zhuoqing Mao, Somesh Jha, Patrick McDaniel, Huan Sun, Bo Li, and Chaowei Xiao. Autodan-turbo: A lifelong agent for strategy self-exploration to jailbreak llms. In *International Conference on Learning Representations*, volume 2025, pp. 10313–10360, 2025b.

Xu Liu, Yan Chen, Kan Ling, Yichi Zhu, Hengrun Zhang, Guisheng Fan, and Huiqun Yu. Astra: An automated framework for strategy discovery, retrieval, and evolution for jailbreaking llms. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 39691–39710, 2026.

Rahul Markasserithodi, Aditya Joshi, Yuekang Li, Ishmanbir Singh, Chris Yoo, and Alan Niu. Chase: Adversarial red-blue teaming for improving llm safety using reinforcement learning. *arXiv preprint arXiv:2606.05523*, 2026.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, et al. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*, 2024.

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

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

Weiwei Qi, Zefeng Wu, Zhilin Guo, Tianhang Zheng, Chaochao Lu, Liang He, Zhan Qin, and Kui Ren. Darwin: Evolving jailbreak adversary and guardrail for llm safety evaluation and protection. *arXiv preprint arXiv:2607.19829*, 2026.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Paul Röttger, Hannah Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. 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)*, pp. 5377–5400, 2024.

Leo Schwinn and Simon Geisler. Revisiting the robust alignment of circuit breakers, 2024. URL https://arxiv.org/abs/2407.15902.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Guoxin Shi, Haoyu Wang, Zaihui Yang, Yuxing Wang, and Yongzhe Chang. Co-evolutionary multimodal alignment via structured adversarial evolution. *arXiv preprint arXiv:2603.01784*, 2026.

Alexandra Souly, Qingyuan Lu, Dillon Bowen, Tu Trinh, Elvis Hsieh, Sana Pandey, Pieter Abbeel, Justin Svegliato, Scott Emmons, Olivia Watkins, et al. A strongreject for empty jailbreaks. *Advances in Neural Information Processing Systems*, 37:125416–125440, 2024.

Zhewen Tan, Wenhan Yu, Jianfeng Si, Tongxin Liu, Kaiqi Guan, Huiyan Jin, Jiawen Tao, Xiaokun Yuan, Xiangzheng Zhang, Duohe Ma, et al. Triplay-rl: Tri-role self-play reinforcement learning for llm safety alignment. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26415–26429, 2026.

Hao Wang, Yanting Wang, Hao Li, Rui Li, and Lei Sha. Be your own red teamer: Safety alignment via self-play and reflective experience replay. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 18700–18716, 2026a.

Haoyu Wang, Yifei Zhao, Zeyu Qin, Chao Du, Min Lin, Xueqian Wang, and Tianyu Pang. Lifelong safety alignment for language models. *Advances in Neural Information Processing Systems*, 38: 19908–19938, 2026b.

Xiaoyu Wen, Zhida He, Han Qi, Ziyu Wan, Zhongtian Ma, Ying Wen, Tianhang Zheng, Xingcheng Xu, Chaochao Lu, and Qiaosheng Zhang. Magic: A co-evolving attacker–defender adversarial game for robust llm safety. *arXiv preprint arXiv:2602.01539*, 2026.

Jiahao Yu, Xingwei Lin, Zheng Yu, and Xinyu Xing. Gptfuzzer: Red teaming large language models with auto-generated jailbreak prompts. *arXiv preprint arXiv:2309.10253*, 2023.

Yuan Yuan, Tina Sriskandarajah, Anna-Luisa Brakman, Alec Helyar, Alex Beutel, Andrea Vallone, and Saachi Jain. From hard refusals to safe-completions: Toward output-centric safety training, 2025. URL https://arxiv.org/abs/2508.09224.

Jingyu Jack Zhang, Haozhu Wang, Eric Michael Smith, Sid Wang, Amr Sharaf, Mahesh Pasupuleti, Ben Van Durme, Daniel Khashabi, Jason E Weston, and Hongyuan Zhan. The alignment waltz: Jointly training agents to collaborate for safety. In *International Conference on Learning Representations*, volume 2026, pp. 92751–92776, 2026a.

Junke Zhang, Jianwei Wang, Sishuo Chen, Yizhang He, Qingshuai Feng, and Zhengyi Yang. Evolving skill-structured attack memory enhances llm jailbreaking. *arXiv preprint arXiv:2605.29237*, 2026b.

Xiaozhe Zhang, Chaozhuo Li, Hui Liu, Shaocheng Yan, Bingyu Yan, Qiwei Ye, and Haoliang Li. Model-agnostic lifelong llm safety via externalized attack-defense co-evolution. *arXiv preprint arXiv:2605.13411*, 2026c.

Rui Zheng, Hongyi Guo, Zhihan Liu, Xiaoying Zhang, Yuanshun Yao, Xiaojun Xu, Zhaoran Wang, Zhiheng Xi, Tao Gui, Qi Zhang, et al. Toward optimal llm alignments using two-player games. In *EMNLP (Findings)*, pp. 78–99, 2025.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

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

Andy Zou, Long Phan, Justin Wang, Derek Duenas, Maxwell Lin, Maksym Andriushchenko, Rowan Wang, Zico Kolter, Matt Fredrikson, and Dan Hendrycks. Improving alignment and robustness with circuit breakers. *Advances in Neural Information Processing Systems*, 37:83345–83373, 2024.

A Extended Related Work

Comparison axes. We compare prior systems by the attack objects they retain, where adaptation occurs, and how they reassess those objects when the defended system changes. These distinctions place RedLoop within established work on automated red-teaming, reusable strategy libraries, co-evolutionary safety training, and over-refusal-aware rewards.

A.1 Automated red-teaming

Attack access and search. Automated red-teaming generates adversarial inputs instead of relying on human-written attacks (Perez et al., 2022). White-box GCG (Zou et al., 2023) optimizes a token-level adversarial suffix from target gradients. Gray-box AutoDAN (Liu et al., 2024) runs a hierarchical genetic algorithm over semantically fluent prompts, scoring candidates by the target model's log-likelihood of an affirmative continuation; it therefore requires logit access. PAIR (Chao et al., 2025) iteratively refines a prompt with an attacker model through black-box queries and reports attacks that typically succeed within twenty queries. TAP (Mehrotra et al., 2024) organizes black-box LLM-driven search as a tree of candidate prompts and prunes unpromising branches. GPTFuzzer (Yu et al., 2023) starts from human-written templates, mutates them, retains successful mutants in a growing seed pool, and selects seeds for later mutation with bandit-style rules.

Reuse across defender updates. GCG's universal suffixes and GPTFuzzer's evolving templates already permit reuse and transfer of prompts or prompt structures. Our comparison concerns the state needed when a defender is retrained: RedLoop retains reusable strategies and ranks previous-round attacks for replay against the current defender. The search mechanisms above address attack construction; our fitness-propagation component ranks attacks between generative-defender training rounds.

A.2 Reusable strategy libraries

Storage, retrieval, and lifecycle. AutoDAN-Turbo (Liu et al., 2025b) distills improved attempts into abstract strategies with definitions and examples, stores them in a library, and retrieves nearby strategies for later queries. It also studies transfer to new target models and updates the library from new attack logs. ASTRA (Liu et al., 2026) organizes a growing library into three score tiers (Effective, Promising, Ineffective) and retrieves top entries by vector similarity against a fixed target. MemoAttack (Zhang et al., 2026b) manages skill units through probation, promotion, retirement, reactivation, and elimination, and uses contextual Thompson sampling over posterior success statistics for selection, also against a fixed target.

Recalibrating stored knowledge. These methods report benefits from persistent, scored, retrieved strategy stores, and motivate the representation we adopt. AutoDAN-Turbo's transfer and library updates also show that reusable knowledge need not be tied to one fixed defender. Our question concerns repeated training updates of the generative defender: a current-round probe ranks previous-round attacks for replay, while novelty injection tests additional candidates for admission. The comparison is therefore about ranking and coverage under this training schedule, rather than introducing strategy storage or retrieval.

A.3 Co-evolutionary safety training with updated attacker weights

Alternating and shared-policy training. MART (Ge et al., 2024) alternates an adversarial model and a target model over four rounds: the attacker is fine-tuned on successful prompts, and the target is fine-tuned on adversarial prompts with selected safe responses. Self-RedTeam (Liu et al., 2025a) trains a single shared policy to play both roles through online self-play with a role-conditioned objective. MAGIC (Wen et al., 2026) keeps the policies separate, warms the attacker on a chain-of-thought attack pool, and approximates a bilevel objective by alternating GRPO updates between the models.

Additional roles, objectives, and buffers. TriPlay-RL (Tan et al., 2026) adds a trained evaluator that distinguishes genuine unsafe content from shallow refusals and trains the attacker against multiple defenders. CHASE (Markasserithodi et al., 2026) uses a template-free red-blue loop, rewarding the attacker with a multiplicative objective combining bypass success and intent preservation, then hardening the defender through GRPO and rejection-sampled fine-tuning. Lifelong Safety Alignment (Wang et al., 2026b) stores successful and failed attack tuples in explicit buffers, improves a Meta-Attacker through beam search and reject fine-tuning, and refusal-trains the defender over a reported two-iteration schedule. GPO (Zheng et al., 2025) formulates a tutor–student game, proves convergence to a Nash equilibrium under its assumptions, and imposes an attacker diversity constraint.

Where adaptation resides. These systems adapt the attacker primarily through its parameters, while some also retain external attack objects. TriPlay-RL keeps successful prompts for its diversity reward; CHASE keeps an iteration-level rewrite pool for defender training; GPO keeps prompt history for a similarity penalty; and Lifelong Safety Alignment uses strategy-containing tuples to condition search. Thus, external memory and attacker training can coexist. RedLoop instead fixes the attack-generation model's weights and adapts its strategy library through current-defender ranking and candidate admission. The specific comparison is ranking previous-round attacks for replay, beyond the training, diversity, and search uses of retained objects described above.

A.4 External attack state across defender updates

Persistent test cases, replay items, and populations. Several systems already retain attack state while the defense changes. HarmBench's R2D2 (Mazeika et al., 2024) trains robust refusal against a persistent, fixed-size pool of concrete GCG test cases: sampled cases are re-optimized against the

current model, and part of the pool is periodically reset. SSP (Wang et al., 2026a) keeps pools of failure goals and failed attack prompts with per-item reward estimates and visit counts, selects by an upper-confidence rule, re-evaluates replayed items under the current policy, and evicts items whose rewards exceed a difficulty threshold. CEMMA (Shi et al., 2026) evolves a bounded population of multimodal prompts through mutation, crossover, and differential evolution, evaluates the active population against the current defender, and trains the defender on a cumulative archive of successful attacks. These mechanisms revisit concrete test cases or examples; RedLoop uses query-instantiable strategy transformations and ranks previous-round attacks for replay from a current-round probe.

Strategy stores with a separately trained defense. EvoSafety (Zhang et al., 2026c) retrieves and composes structured skills from an Adversarial Skill Library and trains a separate auxiliary defense model against a memory bank of successful attacks while keeping the victim model frozen. DARWIN (Qi et al., 2026) retains an evolving strategy pool, selects and composes strategies with a learned transition matrix, and retrains an input guardrail online on emerging attacks. Both therefore maintain reusable knowledge in a changing defended system. Our setting updates the attacked generative model itself, making the current defender the target of the propagation ranking.

What is maintained and what is estimated. R2D2 directly re-optimizes sampled token-level test cases; SSP refreshes selected examples and goals; and CEMMA maintains an evolving active population plus an archive for defender training. In contrast, RedLoop retains a growing, retrieved strategy library and ranks previous-round attacks for replay against the current defender after each generative-defender update. EvoSafety and DARWIN are closer in their retained strategy abstractions, but their reported defense updates target an auxiliary defense or guardrail. The distinction is the combination of state representation, updated model, and sparse estimation; retaining attack state across a changing defense is already established.

A.5 Defenses without an attacker loop

Training objectives and representations. Safe RLHF (Dai et al., 2024) decouples helpfulness and harmlessness into a reward model and a cost model, using a constrained objective with a Lagrangian multiplier to adjust their balance during training. Safe-completions (Yuan et al., 2025) instead centers the safety of the assistant's output rather than a binary refusal decision, allowing informative refusals with useful alternatives. Circuit Breakers (Zou et al., 2024) intervenes on internal representations with a representation-engineering loss and reports robustness to unseen attacks without iterative adversarial retraining; a later evaluation (Schwinn & Geisler, 2024) finds that stronger embedding-space attacks can substantially reduce the reported robustness. These approaches address objectives or representations for safe behavior, while our maintenance operations supply a changing attack curriculum for defender training.

Inference-time filtering. Llama Guard (Inan et al., 2023) filters at inference time, at a different point in the deployment stack from our training-time method. Such filters can complement a trained defender; RedLoop deploys only the resulting generative defender and uses its generator, judge, and library during training.

A.6 The benign side and over-refusal

Measuring and training benign compliance. XSTest (Röttger et al., 2024) and OR-Bench (Cui et al., 2024) make over-refusal measurable alongside attack success. MAGIC (Wen et al., 2026) is particularly relevant to our training objective: its refusal reward penalizes unsafe answers and refusals of safe queries, it rewrites benign seeds into adversarial-benign prompts, and its defender-only ablation loses benign compliance through excessive refusal. WALTZ RL (Zhang et al., 2026a) cooperatively trains conversation and feedback agents with a dynamic improvement reward for useful feedback, reporting reductions in both unsafe responses and over-refusals. Safe-completions (Yuan et al., 2025) also targets over-refusal through output-centric training.

Role in our method. Our defender uses GRPO with a class-conditional reward that includes framed-benign prompts and explicitly penalizes refusing benign queries. MAGIC's refusal-aware reward and adversarial-benign rewrites provide the closest comparison on this axis; benign adversarial framing and the goal of preserving helpfulness are established ideas. In RedLoop, this reward supports the memory-maintenance loop, and we evaluate its resulting safety–helpfulness tradeoff alongside the propagation and injection ablations.

A.7 Evaluation resources

Independent outcome measurements. We use HarmBench (Mazeika et al., 2024), StrongREJECT (Souly et al., 2024), and WildJailbreak (Jiang et al., 2024) for attack success; XSTest (Röttger et al., 2024), OR-Bench (Cui et al., 2024), and PHTest (An et al., 2025) for over-refusal; and IFEval (Zhou et al., 2023) and MMLU (Hendrycks et al., 2021) for capability. These resources serve as measurement instruments rather than alternative maintenance algorithms. Using their benchmark evaluators reduces dependence on the judge that supplies training rewards. Section 4 distinguishes these benchmark outcomes from judge-scored attack and trajectory analyses.

A.8 Positioning and gap

Established ingredients. Among the reviewed methods, AutoDAN-Turbo, ASTRA, and MemoAttack pair frozen attack generators with persistent strategy stores; AutoDAN-Turbo also transfers and updates its library from new attack logs. R2D2, SSP, and CEMMA maintain external attack state while the defense changes, using concrete test cases, replay examples, or populations and archives. EvoSafety and DARWIN maintain growing skill or strategy stores while updating a separate auxiliary defense or guardrail. These precedents establish the individual ingredients on which our design builds.

The combination studied here. Our distinction among these reviewed methods is the conjunction of a growing library of reusable, query-instantiable strategy transformations; repeated updates of the attacked generative defender; per-strategy fitness estimates used for selection; and a current-defender ranking of previous-round attacks for replay. The attacker's weights remain fixed. In each round, a probe drawn from 15% of the current round's attack records ranks previous-round attacks for replay, and a Gaussian process over strategy embeddings estimates their current scores. Novelty injection separately tests additional candidates for admission. The main experiments examine the resulting tradeoff and the contribution of these operations, while leaving claims about direct ranking accuracy, coverage, and total evaluation cost subject to the evidence specified there.

B Additional Analyses

B.1 RedScout Red-Teamer

This section reports the generation protocol behind Table 4, the propagation estimators evaluated in Section B.2, and the contribution of the selection terms. The red-teamer throughout is RedScout, the frozen-policy library system of Section 2.1.

Each generation step samples 128 strategies by temperature-η softmax over Eq. (2) and generates an adversarial prompt for each harmful query in the round's query set. Prompts are scored under the answer-aggregation protocol of Appendix C: four answers and one judge replicate inside the loop. The controlled comparison in Table 4 uses one answer for each system; both systems therefore receive the same scoring treatment. The judge is the Gemma-4-31B-it model at temperature 0. A given prompt and defender therefore receive the same score across steps. The three terms of Eq. (2) are the fitness estimate, the usage-count bonus, and the embedding-novelty term. Table 2 reports the end-to-end effect of removing the components these terms serve.

AutoDAN-Turbo (Liu et al., 2025b) is the closest baseline. The reported comparison uses Nemotron-Nano-4B with reasoning off, Mistral-Small-24B as the attack-generation model, the same judge and 128 harmful queries, and a 200-step run for RedScout. The attack-success panel of Table 4 uses single-answer scoring for both systems. Its library-quality panel reports fewer measured duplicate strategies for RedScout (0.0% versus 68.7%), a maximum pairwise token-Jaccard similarity of 0.355, and a larger fraction of strategies with severity at least 7 (41.6% versus 17.2%).

The same table reports higher attack-success point estimates for RedScout under the common single-answer protocol. Its runtime comparison is configuration-dependent: RedScout takes 2.9 hours versus 84.3 hours for AutoDAN-Turbo, and produces severity-≥ 7 strategies at 33.2 versus 0.85 per hour. AutoDAN-Turbo's attack refinement is serial, so these wall-clock numbers describe the implemented systems.

Table 5 characterizes attack coverage across four frozen defenders and two red-teamers. Query-level ASR (severity ≥ 7) spans 58.6% to 73.4%. Both red-teamers attack the lighter Llama-3.1-8B defender more successfully (prompt-level ASR at severity ≥ 7 of 10.3%) than the trained Nemotron-Nano-4B defender (5.0%). The Nemotron-Super-49B red-teamer exposes a precision-diversity

Table 4: REDSCOUT (ours) compared with AutoDAN-Turbo under a matched evaluation budget. Our run uses 200 generation steps and n=128 queries on the Nemotron-Nano-4B defender with the Mistral-Small-24B red-teamer. Bold marks the better value in each row. Attack-success rows use the same single-answer scoring for both methods. AutoDAN-Turbo reports 1,336 raw strategy identifiers, which reduce to 419 strategy texts.

Axis	AutoDAN-Turbo	REDSCOUT (ours)	Margin
Strategy library (single-answer scoring)			
Stored strategy texts	419	231	
Duplicate strategies	288	**0**	
Duplicate rate (token-set Jaccard ≥ 0.85)	68.7%	**0.0%**	
Strategies with severity ≥ 5 / ≥ 7 / ≥ 9	**395** / 72 / 10	219 / **96** / **20**	
% Strategies with severity ≥ 5	94.3%	**94.8%**	1.0×
% Strategies with severity ≥ 7	17.2%	**41.6%**	2.4×
% Strategies with severity ≥ 9	2.4%	**8.7%**	3.6×
Attack success (matched single-answer scoring)			
Prompt-level ASR (severity ≥ 5)	0.130	**0.142**	+9% relative
Prompt-level ASR (severity ≥ 7)	0.026	**0.031**	+19% relative
Query-level ASR (severity ≥ 5)	0.828	**0.867**	+3.9 points
Query-level ASR (severity ≥ 7)	0.461	**0.539**	+7.8 points
Computational efficiency			
Wall-clock under reported configuration†	84.3 h	**2.9 h**	∼29× faster
Strategies with severity ≥ 7 per hour†	0.85	**33.2**	39×

All severity scores are in range 1–10 and LLM-judged. †AutoDAN-Turbo is natively sequential (Liu et al., 2025b): its warm-up and lifelong loops issue one query per query and its within-query refinement chain is serial. The wall-clock rows therefore report each method as designed. A compute-matched concurrent check equalizes parallelism and confirms that the throughput gap is not closed by additional compute. Strategy-level ASR counts distinct strategies whose best attack reaches severity $\geq \tau$; prompt-level ASR counts attacks reaching τ; query-level ASR counts harmful queries with at least one such attack.

Table 5: Attack coverage across defenders and red-teamers (all values in %). ASR = attack success rate. Per-attack rates use the median of four answers; the per-query rate counts a query as solved if at least one strategy succeeds. A higher value indicates greater attack success; bold marks the best value in each column, and tied best values are all shown in bold. †Partial self-evaluation (same judge family).

Defender	Prompt-level ASR (severity ≥ 5)	Prompt-level ASR (severity ≥ 7)	Query-level ASR (severity ≥ 7)	% Strategies with best attack ≥ 7
Mistral-Small-24B red-teamer (200 steps)				
Gemma-4-31B-it†	13.3	3.1	58.6	55.1
Nemotron-Nano-4B	18.1	5.0	66.4	61.7
OLMO-3-7B	20.2	8.2	**73.4**	91.1
Llama-3.1-8B	**32.1**	**10.3**	**73.4**	**94.5**
Nemotron-Super-49B red-teamer (200 steps)				
Nemotron-Nano-4B	9.9	1.9	40.6	81.7

tradeoff: a larger fraction of its strategies reach severity ≥ 7 (81.7% vs. 61.7%), yet cumulative query coverage is much lower (40.6% vs. 66.4%). The two red-teamers also differ in their degree of safety alignment, and the stronger alignment of the larger model does not translate into a stronger attacker: on the same defender, Nemotron-Super-49B reaches prompt-level attack success of 9.9% and 1.9% at severity ≥ 5 and ≥ 7, against 18.1% and 5.0% for Mistral-Small-24B. Heavier safety alignment in the attacker therefore reduced red-teaming effectiveness rather than improving it, and we use Mistral-Small-24B for the main loop.

B.2 PROPAGATION VALIDATION

Section 2.3.1 uses the current-round probe to rank previous-round attacks for replay. Persistence carries each previous-round attack's stale score forward, probe-mean assigns every unprobed strategy the mean probe score, softmax-cosine averages probe scores with weights determined by embedding

Table 6: **Strategy-injection methods on the standard benchmarks.** Each row changes only the method that proposes candidates, and is scored by each benchmark's own official evaluator as in Table 1. Bold marks the best value in each column.

Injection method	Harm Bench↓	Strong REJECT↓	XSTest↓	PHTest↓	MMLU↑
Untrained base	21.12	0.0454	68.0	64.61	41.90
Residual-query (RQDD)	**12.50**	0.0520	46.0	51.52	42.14
Defender-failure-domain (DFDD)	14.25	0.0603	47.2	48.68	**42.35**
Severity-weighted (SWDD)	14.25	**0.0430**	50.0	53.49	40.78
Query-to-weakness (QWB)	14.25	0.0470	48.8	53.25	40.34
Manifold-void exploration (MVE)	15.50	0.0569	**41.6**	**47.86**	41.72
Persistent weakness (PWDD)	16.25	0.0481	44.4	50.36	40.68
Near-miss (NMDD)	17.25	0.0610	43.2	49.30	41.79

similarity, and strategy-cluster assigns each unprobed strategy the mean score of its nearest greedy cosine cluster. The deployed GP-UCB estimator uses the cosine-RBF posterior in Eq. (7); its posterior mean estimates a current score and its mean-plus-uncertainty value ranks previous-round attacks for the replay allocation. All estimators use the same current-round probe and strategy embeddings.

The operational probe contains 15% of the current round's attack records, is disjoint from the training and validation records, and provides at least one attack for each represented strategy. Those attacks were scored against the current defender as part of the round's generation. Direct scoring therefore covers 15% of the round's records rather than a full library sweep.

B.3 Injection Methods

Novelty injection proposes candidate strategies and admits only those that pass the two conditions of Eq. (5); the method used to propose them is a configuration axis rather than a fixed choice. Table 6 reports seven alternatives on the standard benchmarks; the full ATPL configuration is reported in Table 2. DFDD and MVE appear in both that table and Table 2. We define those two methods and the remaining variants below.

Defender-failure-domain injection (DFDD) embeds the prompts on which the updated defender complied, clusters the embeddings, and proposes strategies for the shared failure patterns. *Manifold-void exploration* (MVE) searches strategy-embedding space for directions with low maximum cosine similarity to the stored library and proposes strategies for those regions.

Severity-weighted discovery (SWDD) forms the DFDD failure groups from the severity-weighted distribution instead of the unweighted one. This weighting oversamples the domains that produced the most severe failures. *Near-miss domain discovery* (NMDD) reads the prompts whose best answer fell below the success threshold while remaining close to it, and abstracts the difference between such a prompt and one that succeeded. *Persistent weakness discovery* (PWDD) abstracts a failure pattern that recurs across rounds into a reusable description. *Query-to-weakness bridging* (QWB) fits the mapping from a refused query to the vulnerability class that broke through earlier on a related query, and applies it to queries in the library's record that it has not yet reached. *Residual-query domain discovery* (RQDD) reads the high-scoring prompts left unresolved by earlier rounds. All five condition on an observed failure, as DFDD does. Each method forms its candidates through the same two stages. A coherence filter scores each candidate with the frozen attack-generation model and discards those scoring below 4 on a 1–5 scale. This filter removes text that does not state a strategy. The candidates that pass the filter then enter a separation loop that re-prompts the model toward a more distant domain, and the loop repeats until a candidate clears the separation margin δ. We set $\delta = 0.28$ and keep at most 100 candidates per round.

Each alternative lowers HarmBench attack success relative to the untrained base, but several increase the StrongREJECT score. Proposal choices therefore yield different measured tradeoffs. RQDD attains the lowest HarmBench attack success, MVE the lowest XSTest and PHTest over-refusal, and SWDD the lowest StrongREJECT score.

B.4 Cross-Architecture Results

We repeat the full framework on two additional defender families. Table 7 reports standard-benchmark results on them, scored with the same evaluators as Table 1 and with the same method

Table 7: **Cross-architecture evaluation on standard benchmarks.** Benchmark-native scorers, as in Table 1. Attack success and over-refusal are lower-is-better; MMLU is higher-is-better. Both defender families are shown with the same set of methods, so no row is compared across uneven coverage. Bold marks the best value in each column within each defender family; ties are all bold.

		Attack success ↓			Over-refusal ↓		Capability ↑
Defender	Method	Harm Bench	Strong REJECT	Wild Jailbreak	XSTest	OR-Bench	MMLU
Llama-3.1-8B	Base	32.30	0.0134	40.35	11.2	74.30	**68.77**
Llama-3.1-8B	RedLoop (Full)	**18.01**	0.0129	**1.15**	**3.6**	76.72	68.03
Llama-3.1-8B	RedLoop (Propagation-only)	25.16	**0.0124**	3.40	5.6	**71.27**	68.12
Llama-3.1-8B	Circuit Breakers	30.75	0.0134	40.10	10.8	74.22	68.69
OLMO-3-7B	Base	6.75	0.0288	2.45	16.0	**95.68**	**60.65**
OLMO-3-7B	RedLoop (Full)	**1.55**	**0.0122**	**0.40**	**14.4**	97.27	60.4
OLMO-3-7B	RedLoop (Propagation-only)	2.48	0.0146	0.50	15.2	96.89	60.4
OLMO-3-7B	Circuit Breakers	8.70	0.0284	3.20	16.8	95.98	60.63

set for both. On Llama-3.1-8B, RedLoop (Full) reduces WildJailbreak attack success from 40.35% to 1.15% and lowers XSTest over-refusal from 11.2% to 3.6%; OR-Bench rejection is its only non-improving axis ($74.3\% \rightarrow 76.7\%$). On OLMO-3-7B it reaches 0.4% WildJailbreak and 1.55% HarmBench against an untrained base of 2.45% and 6.75%. Capability stays within 0.3 points of the untrained OLMO model and 0.8 points of the untrained Llama model.

C Full Algorithm and Implementation Details

A run spans T rounds. Each round has four phases: the red-teamer generates and scores a batch of adversarial prompts against the current defender; a current-round probe ranks previous-round attacks for replay and the training batch is assembled from current attacks, ranked replay, and a capped benign allocation; the defender is updated by GRPO under the class-conditional reward of Eq. (3); and the strategy library is expanded by novelty injection before the next red-teaming phase.

Defenders are trained with GRPO (Shao et al., 2024) through VERL (Sheng et al., 2025) on LoRA adapters of rank 64 (α=64), learning rate 2×10^{-5}, group size G=8, and KL coefficient β_{KL}=0.03 measured against the round's starting checkpoint. The objective maximized per round is

$$\mathcal{J}(\theta) = \mathbb{E}_x \left[\frac{1}{G} \sum_{j=1}^{G} \min\Big(\rho^{(j)} \hat{A}^{(j)},\ \mathrm{clip}\Big(\rho^{(j)}, 1{-}\epsilon, 1{+}\epsilon \Big)\, \hat{A}^{(j)} \Big) \right] - \beta_{\mathrm{KL}}\, \mathrm{KL}(\pi_\theta \,\|\, \pi_{\theta_t}), \tag{6}$$

with importance ratio $\rho^{(j)} = \pi_\theta(y^{(j)} \mid x) / \pi_{\theta_t}(y^{(j)} \mid x)$ against the pre-update policy, and $\hat{A}^{(j)}$ the reward $r(x, y^{(j)})$ of Eq. (3) standardized over the G answers sampled for the same prompt. The reward map is $w : \{1,2,3,4,5\} \mapsto \{-0.75, -0.25, 0, 0.75, 1.5\}$ and the benign compliance margin is ρ_{b}=0.5.

Stored fields and settings. Each library entry stores its strategy text, a unit-norm bge-m3 embedding, a usage count, and a direct fitness estimate. For Eq. (2) we use $c = 1.0$, $\lambda = 0.5$, $T_s = 0.5$, $\phi_0 = 0.15$, and $k = 5$ neighbors. A prompt becomes eligible for strategy distillation when the best of its R sampled answers reaches severity 5 on the 1–10 scale; eligible prompts are ranked by severity and strategy novelty, and the frozen generator summarizes selected prompts into reusable text with query-specific content removed. Distilled strategies are checked for redundancy and enter with an averaged fitness damped toward the middle of the scale. The update concentrates on prompts whose answers disagree, sampled by their tracked attack-severity standard deviation and a difficulty factor. Direct evaluation of every stored strategy would require an attack-generation call, R defender answers, and RK judge calls per strategy.

Within a round's batch, the replay allocation is 20% of the *dynamic* partition, the current and replayed attack examples held separately from the benign allocation. Of the remaining fresh dynamic examples, 75% are contested and the rest are vulnerable; the benign and framed-benign partition is instead capped at the benign share β_{B} of the total batch. These proportions therefore use different denominators. We classify a training example as contested using the attack-severity judge alone: when the standard deviation of its attack-severity scores across sampled answers reaches $\sigma_{\min}$=0.5 and its mean attack-severity score reaches 5.0 on the 1–10 attack-severity scale. Attack scoring

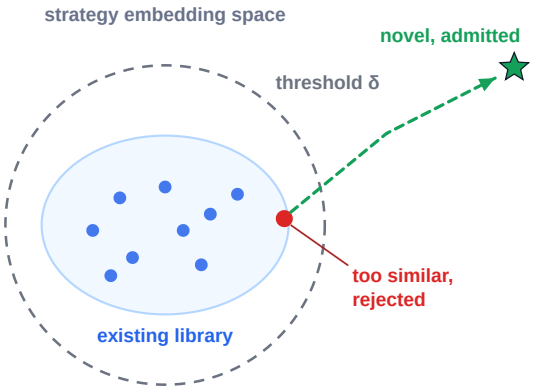


Figure 6: Novelty admission in strategy-embedding space. A candidate that is too similar to the stored library is redirected toward a distant region and admitted only after clearing the separation threshold.

Algorithm 1 RedLoop: one co-evolutionary run

Require: base defender π_{θ_0}; red-teamer REDSCOUT with a frozen attack-generation model and library $\mathcal{L}_0$; judge J; harmful queries $\mathcal{Q}$; benign queries $\mathcal{B}$; probe fraction ρ; GP-UCB parameters $(\ell, \sigma_n^2, \beta)$

Ensure: trained defender π_{θ_T}

initialise $\mathcal{L}_0$ by one seed phase against π_{θ_0}; set each f_i by Eq. (1)
for $t = 0$ to $T - 1$ **do**
  generate: sample strategies from $\mathcal{L}_t$ by Eq. (2) and score the resulting prompts against π_{θ_t}
  distil: add reusable strategies from successful prompts, deduplicated against $\mathcal{L}_t$
  if $t > 0$ **then**
    probe: reserve a coverage-balanced ρ fraction $\mathcal{P}_t$ of current attack records, disjoint from training and validation
    propagate: rank $\mathcal{A}_{t-1}$ for replay using the GP-UCB posterior of Eq. (7) fitted to $\mathcal{P}_t$
  end if
  update: $\pi_{\theta_{t+1}} \leftarrow \mathrm{GRPO}(\pi_{\theta_t})$ by Eq. (6) under the reward of Eq. (3)
  inject: add candidate strategies against $\pi_{\theta_{t+1}}$ that satisfy Eq. (5)
  retain: carry $\mathcal{L}_{t+1}$ into round $t{+}1$; stored strategies are never discarded ($\mathcal{L}_{t+1} \supseteq \mathcal{L}_t$)
end for

uses $R{=}4$ defender answers and $K{=}1$ judge replicate per answer in the red-teaming loop. The attack score $\mathrm{score}(x)$ of Section 2.1 is therefore the median of four scored answers; the controlled comparison in Table 4 sets $R{=}1$ for both systems. Judge decoding is greedy at temperature 0.

The propagation model operates on bge-m3 embeddings of strategy text under a cosine-distance kernel, with length scale $\ell{=}0.30$, observation noise $\sigma_n^2{=}0.05$, prior variance $\sigma_f^2{=}1$, and GP-UCB weight $\beta{=}1.0$, all fixed across rounds. For the current-round probe drawn from 15% of the round's attack records, the implementation uses the probe's mean score as the GP mean function to rank previous-round attacks for replay. For a previous-round strategy v, the propagated mean and uncertainty are

$$\mu(v) = \bar{y}_{\mathcal{S}} + k(v,\mathcal{S})K_{\mathcal{S}}^{-1}(y_{\mathcal{S}} - \bar{y}_{\mathcal{S}}), \qquad \sigma^2(v) = \sigma_f^2 - k(v,\mathcal{S})K_{\mathcal{S}}^{-1}k(\mathcal{S},v), \tag{7}$$

where $K_{\mathcal{S}} = k(\mathcal{S},\mathcal{S}) + \sigma_n^2 I$. The deployed carry-forward ranking is $\mu(v) + \beta\sigma(v)$; the count and novelty terms in Eq. (2) are used separately when generating attacks within a round. Novelty injection admits a candidate onlyunder Eq. (5), with separation margin $\delta{=}0.28$, a probe of $n_c{=}80$ queries, eight generated prompts per query, and threshold $\gamma{=}5$; the minimum breaking fraction $q_{\min}$ and distillation threshold are fixed per run. The reported full configurations use ATPL; the choice among the alternative injection methods is a swept configuration axis; the methods are compared in Section B.3.

D METRIC DEFINITIONS AND AGGREGATION LEVELS

For a prompt x and an answer y, the attack judge returns a 1–10 severity score $g(x, y)$ on a harmful prompt, used only for attack scoring and strategy fitness. A separate reward judge returns a 1–5 score: an answer-safety score $s(x, y)$ on a harmful prompt and a helpfulness score $h(x, y)$ on a benign prompt, used for the defender's reward and held-out ASR. The two judges and their scales are distinct. For each prompt, we compute the score as the median over its R sampled answers of the mean over the K judge replicates for each answer. Taking the median outermost prevents a single unsafe answer from determining the prompt score, while the inner mean averages replicate noise before that comparison. When $R{=}1$, the prompt score equals the score of its single answer.

The red-teamer comparisons use a severity threshold γ on the attack score of Eq. (1), so an attack counts as successful when $\mathrm{score}(x) \geq \gamma$ on the 1–10 severity scale. We report $\gamma = 5$, $\gamma = 7$, and $\gamma = 9$, and a rate labelled with a severity threshold uses that value throughout: a rate at severity 7 is the fraction of attempted attacks whose score reaches 7. An attack succeeding at a higher γ is a strictly stronger attack, which is why the same comparison can show a large difference at $\gamma = 7$ and a small one at $\gamma = 5$.

Given the set $\mathcal{A}_i$ of prompts generated from stored strategy τ_i, the three aggregation levels are the fraction of stored strategies with $\max_{x \in \mathcal{A}_i} \mathrm{score}(x) \geq \gamma$, the fraction of generated prompts with $\mathrm{score}(x) \geq \gamma$, and the fraction of harmful queries q with $\max_{x:\, q(x)=q} \mathrm{score}(x) \geq \gamma$. These are the strategy-, prompt-, and query-level rates in Tables 4 and 5. They measure distinct quantities: the query-level rate counts a harmful query when at least one attempt succeeds and therefore increases with the attempt budget, while the prompt-level rate counts each attempt separately. Strategy fitness f_i in Section 2.1 is instead the mean score over $\mathcal{A}_i$, because selection uses the expected score of deploying a strategy rather than its maximum score.

We distinguish the two scoring functions of Section 2.1 to prevent conflation. The attack scorer returns severity under the lifelong-jailbreak protocol introduced by AutoDAN-Turbo (Liu et al., 2025b), with an instruction block that fixes the scoring scale and excludes middle-of-scale hedging. We record the prompt text, scale anchors, and decoding configuration with each checkpoint, making scores comparable across rounds and with that protocol. The reward judge returns a 1–5 score, answer-safety for harmful prompts and helpfulness for benign prompts, used only for the defender's update and held-out ASR. Both judges use the same frozen model at temperature 0, and no quantity in this paper mixes the two scales: attack severity appears in fitness, selection, novelty injection, and the strategy-, prompt-, and query-level rates, whereas the 1–5 reward scores appear in Eq. (3) and held-out ASR.

Held-out ASR is the share of held-out harmful prompts whose answer receives answer-safety score ≤ 2, and over-refusal is the share of held-out benign prompts classified as refusals, using the same refusal decision as the $\mathbf{1}[\text{refuse}]$ term of Eq. (3).

Attack success therefore has two distinct definitions in this paper, and they are not comparable. A *held-out* attack-success rate uses the 1–5 answer-safety score and counts a harmful prompt as broken when that score is at most 2. A *severity-threshold* rate uses the 1–10 severity score and counts an attack as successful when it reaches the stated threshold, taken here from $\{5, 7, 9\}$. The two are reported in separate tables and are never averaged together or used as substitutes for one another.

Benchmarks scored by benchmark-native scorers use those scorers' native labels and are reported in the units each defines (Table 1 states them).

Interval estimates are 95% Wilson intervals per model and normal-approximation intervals for differences between models; paired within-prompt comparisons use a 10,000-resample bootstrap on the paired difference (seed 20260914). A difference is called significant only when zero lies outside the interval for the difference, and the interval is quoted wherever that word appears.