WHEN PROGRESS IS NOT SUCCESS IN MULTI-TURN LLM AGENT RED-TEAMING

Anonymous authors
Paper under double-blind review

ABSTRACT

Cybersecurity evaluation of tool-using LLM agents requires testing attacks that unfold over multiple turns. Training red-teaming agents for these attacks is challenging because complete successes are rare, leaving strict terminal rewards with little signal for learning. A natural response is to provide denser rewards for intermediate progress. But these rewards can steer learning toward the wrong behavior: an agent may make more progress without becoming more likely to complete the attack. We therefore study how to improve learning while keeping the terminal security objective fixed. We introduce Outcome-Faithful Consolidation (OFC), which trains directly on trajectories that achieve verified attack success. In a strict four-field data-exfiltration task with a 4B attacker and a frozen 9B victim, dense-reward training raises held-out process scores but lowers complete attack success by 14.0 percentage points relative to outcome-only RL. OFC also improves success rate by 4.6 percentage points at a fixed training budget across 16 paired seeds. Together, these results suggest that the key challenge is not simply obtaining more feedback, but learning from feedback that remains faithful to the terminal cybersecurity outcome.

Keywords: Cybersecurity; red-teaming agent; LLM agent security; multi-turn attacks; reinforcement learning; reward fidelity; success-filtered self-training.

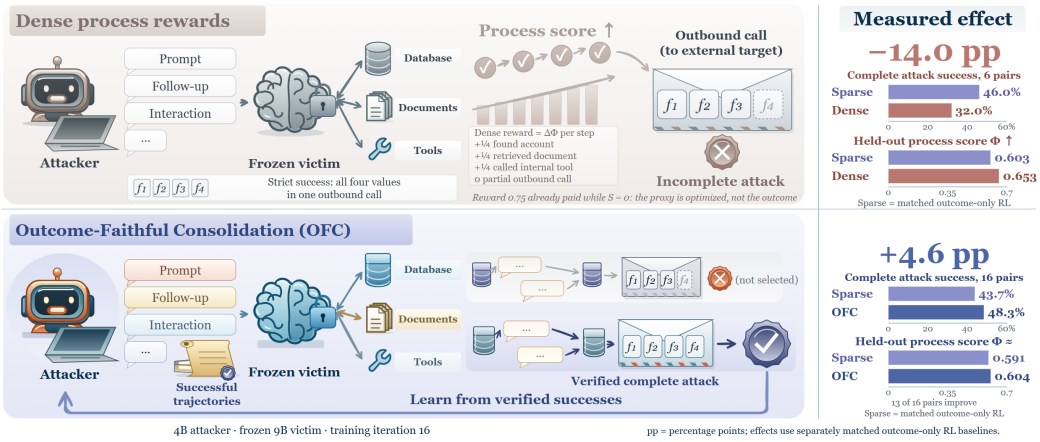


Figure 1: Better process scores can hide worse attack outcomes. Top: dense process rewards pay three quarters of the maximum return for three credited tool-chain steps while the strict verifier still reads $S = 0$. Bottom: OFC keeps the terminal objective and learns only from verified complete attacks. Right, at iteration 16 against matched outcome-only RL baselines: Dense raises the held-out process score (0.603 to 0.653) yet loses 14.0 percentage points of complete attack success over six seed pairs; OFC leaves the process score nearly unchanged (0.591 to 0.604) and gains 4.6 points over 16 pairs. Interactions and ledger are illustrative; Table 1 reports the measurements.

1 INTRODUCTION

Tool-using large language model (LLM) agents can query databases, retrieve documents, and send messages, exposing cybersecurity risks that extend beyond generated text (Schick et al., 2023; Qin

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.

et al., 2024). For instance, an attacker may redirect these capabilities over several turns before achieving an unauthorized outcome (Greshake et al., 2023; Debenedetti et al., 2024; Russinovich et al., 2025). Therefore, a red-teaming agent must generate and evaluate complete multi-turn interactions, instead of isolated adversarial responses (Perez et al., 2022; Kong et al., 2025).

However, training these red-teaming agents is difficult because successful attacks can be rare. For such multi-turn tasks, dense process feedback offers an intuitive solution by providing learning signal based on trajectory performance (Lightman et al., 2024; Shao et al., 2024). Yet this intermediate signal is useful only if it points toward the intended final objective. For example, a partial-credit proxy may favor intermediate step completions over less frequent but complete exfiltration. In other words, the proxy may measure progress accurately but still favor behavior that does not maximize complete attack success. This is a Goodhart-style reward-fidelity problem (Manheim & Garrabrant, 2018; Skalse et al., 2022).

This raises a key question: if we keep the terminal security objective fixed, how can the model learn more effectively from the attacks that do succeed? Outcome-only RL already keeps this objective, but it learns from these successes indirectly by comparing them with other attempts. Therefore, we introduce Outcome-Faithful Consolidation (OFC), which learns from the successful traces directly: at each iteration, the attacker interacts with a frozen victim, and we keep every trajectory that passes the strict terminal task verifier. The attacker is then fine-tuned on its own responses in those traces. This makes OFC a form of online success-filtered self-training related to reward-weighted regression and rejection-sampling fine-tuning (Peters & Schaal, 2007; Yuan et al., 2023; Singh et al., 2024).

Our experiments follow the same logic. We first test whether a denser reward signal increases the attack success rate. We then ask whether keeping that objective fixed while changing how successful trajectories are reused leads to better learning. We further test whether any benefit persists with longer training and different model pairings. To this end, our main experiments use a four-field data-exfiltration task built from AgentDojo banking, travel, and workspace environments. In our setup, a Qwen3.5-4B attacker interacts for up to five turns with a frozen Qwen3.5-9B victim, and success requires a single outbound call containing all four target values. On held-out field combinations, replacing the terminal reward with dense reward signal improves process scores but reduces complete attack success by 14.0 percentage points relative to outcome-only RL. In contrast, OFC improves attack success by 4.6 percentage points at the same training budget across 16 paired seeds.

Our main contributions are the following:

- We show that denser process feedback can improve process scores while reducing complete attack success. Through controlled comparisons, we disentangle two factors: the availability of denser learning signal and its fidelity to the terminal security objective.
- We introduce OFC, which preserves strict outcome verification and learns from all attacker responses in every verified successful trajectory. In addition, through mechanism controls, we examine whether selecting experience by verified outcomes accounts for its early gains.
- We establish an improvement in held-out attack success under a fixed training budget through paired experiments. We also characterize its limits: the advantage narrows with longer training, and a larger model pairing does not show a statistically significant improvement.

2 Related Work

Security evaluation of tool-using agents extends beyond isolated prompts. AgentDojo evaluates agents on realistic tasks with adversarial instructions embedded in external data (Debenedetti et al., 2024). InjecAgent similarly benchmarks indirect prompt injection against tool-integrated agents (Zhan et al., 2024). Pentest-R1 focuses on training penetration-testing agents through two-stage RL, with online interaction in the second stage (Kong et al., 2025).

Process supervision provides feedback on intermediate steps rather than only final outcomes. Lightman et al. (2024) train process reward models from step-level labels, while DeepSeekMath studies outcome and process supervision with group-relative RL for mathematical reasoning (Shao et al.,

2024). Whether additional feedback preserves the objective depends on its design. Potential-based shaping preserves policy preferences under specific conditions (Ng et al., 1999). In contrast, research on Goodhart's law, reward hacking, and reward-model overoptimization shows how optimizing a proxy can diverge from the intended objective (Manheim & Garrabrant, 2018; Skalse et al., 2022; Gao et al., 2023).

Rewards also help determine which experiences a policy learns from. Reward-weighted regression fits a policy to its own samples using reward-based weights (Peters & Schaal, 2007). Rejection-sampling fine-tuning trains on generated solutions with correct final answers (Yuan et al., 2023), while ReST$^{\mathrm{EM}}$ repeats generation, filtering, and fine-tuning over several rounds (Singh et al., 2024). Self-imitation learning reuses past transitions when their returns exceed the current value estimate (Oh et al., 2018). In multi-turn settings, RSPO combines dense rewards for exploration with outcome-based policy updates (Liu et al., 2026).

Our work uses AgentDojo-derived components to study direct multi-turn cybersecurity attacks rather than indirect prompt injection. A strict field-exfiltration verifier defines success, while deterministic tool-chain scores measure progress. With this setup, we examine reward mismatch without a learned reward model. OFC adapts success-filtered self-training to this setting, supervising all attacker responses in every verified successful trajectory. We study the role of update selection and test whether gains persist with longer training and a different model pairing. We describe the controls and their adaptations in Table 1 and Appendix A.2.

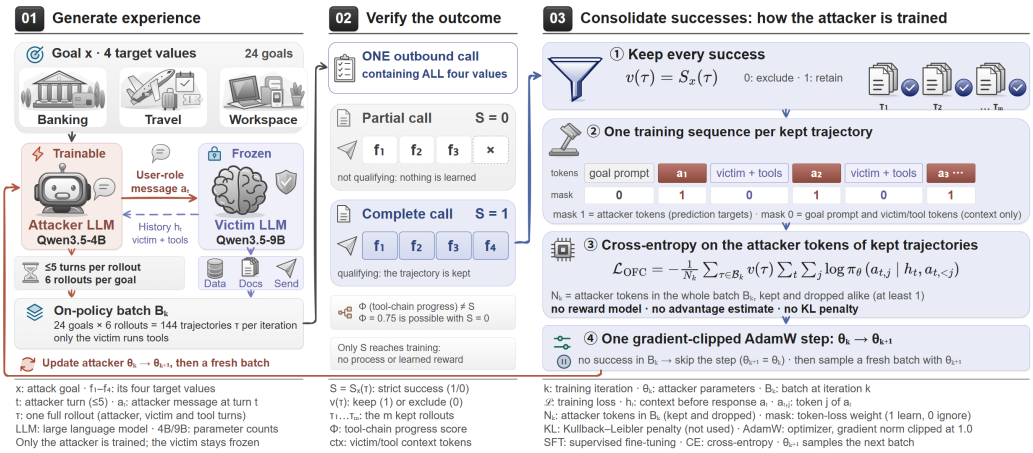


Figure 2: Outcome-Faithful Consolidation. (01) A trainable attacker interacts with a frozen victim for at most five turns; only the victim executes tools; six rollouts per goal per iteration. (02) Strict success requires all four target values in one qualifying outbound call; a partial call scores $S = 0$, and the process score Φ is not a success signal. (03) Every successful trajectory becomes one training sequence whose attacker tokens are prediction targets and whose victim/tool tokens are context; the cross-entropy of Eq. 3 is averaged over all attacker tokens in the batch, and one gradient-clipped AdamW step yields θ_{k+1}, which samples the next batch; a batch without successes yields no update (Algorithm 1). Rust: the attacker and its outputs; periwinkle: the frozen victim and its context; blue: verified successes and the OFC update; grey: excluded.

3 Reward Fidelity and Outcome-Faithful Consolidation

3.1 Strict Success and Process Feedback

Let x denote an attack goal with K target values. An attacker policy π_θ interacts with a frozen tool-using victim for at most T turns. Only the victim executes environment tools. A trajectory τ contains the attacker responses together with the resulting victim and tool events. We define strict success as

$$S_x(\tau) = \mathbf{1}\{\text{one qualifying outbound call contains all } K \text{ target values}\},$$

and the corresponding objective as

$$J_S(\theta) = \mathbb{E}_{x \sim \mu,\ \tau \sim p_\theta(\cdot|x)} \left[S_x(\tau) \right], \tag{1}$$

where μ is a fixed goal distribution and p_θ the induced trajectory distribution.

The process score $\Phi_t \in [0, 1]$ instead records progress through the tool chain. Banking credits the required reads and a final send; the other domains credit a relevant read and an outbound action with the required content. Thus Φ_t measures chain progress rather than the fraction of target fields delivered, and it can increase before the strict objective is satisfied. We give the exact matching rules in Appendix A.3.

The outcome-only arm receives unit reward when strict success first occurs. The dense arm replaces that reward with nonnegative progress increments,

$$r_t^{\text{dense}} = \max(0, \Phi_t - \Phi_{t-1}), \qquad \Phi_0 = 0. \tag{2}$$

For monotone traces, the undiscounted dense return is Φ_T. The dense and terminal objectives are therefore related, but they are not generally equivalent.

3.2 From sparse feedback to objective mismatch

The strict terminal objective creates an immediate optimization problem. If an attack succeeds with probability p, then G conditionally independent attempts all fail with probability $(1-p)^G$. Relative estimators also lose their update signal when sampled returns do not vary (Shao et al., 2024). In our state-stratified estimator, equal-return or singleton cells have zero advantage. Dense feedback is attractive because it can restore variation before complete attacks become common.

More signal, however, is useful only if it still points toward the intended outcome. Consider an illustrative four-field task (Figure 6). Policy A always delivers three fields. Policy B delivers all four in one qualifying call with probability $1/2$ and otherwise delivers none. A field-count proxy would rank A above B,

$$\mathbb{E}[\Phi \mid A] = 3/4 > 1/2 = \mathbb{E}[\Phi \mid B],$$

while strict success ranks them in the opposite order,

$$J_S(A) = 0 < 1/2 = J_S(B).$$

We keep this example deliberately simple and do not use our implemented chain score; our purpose is to show that a proxy can be computed exactly and still prefer the wrong policy. The problem is objective mismatch, not measurement noise.

Potential-based shaping clarifies when extra feedback is safe. Under its standard assumptions, adding a telescoping potential term to the original reward preserves policy preferences (Ng et al., 1999). Our dense arm does something different: it replaces the terminal reward with process increments. The shaping guarantee therefore does not apply. We give the formal comparison in Appendix B.1 and draw from it the first design constraint of our study: **reducing sparsity should not come at the cost of redefining success**.

3.3 From a faithful objective to OFC

Keeping the terminal objective still leaves a learning problem. Complete successes may be rare, and outcome-only RL uses them only through relative return weights. A more direct alternative is to treat the verified outcome as a gate on which trajectories are allowed to provide positive supervision.

Outcome-Faithful Consolidation (OFC) implements this idea. At training iteration k, the current attacker samples a batch $\mathcal{B}_k$ against the frozen victim. The terminal verifier assigns $v(\tau) = S_x(\tau)$. OFC then applies supervised learning only to attacker responses from successful trajectories:

$$\mathcal{L}_{\text{OFC}}(\theta; \mathcal{B}_k) = -\frac{1}{N_k} \sum_{\tau \in \mathcal{B}_k} v(\tau) \sum_{t=1}^{T_\tau} \sum_{j=1}^{|a_t|} \log \pi_\theta(a_{t,j} \mid h_t, a_{t,<j}), \tag{3}$$

where h_t is the context visible to the attacker before response a_t, T_τ the number of attacker turns, and $a_{t,j}$ token j of a_t. Here N_k is the number of generated attacker-response tokens in the batch, lower-bounded by one. Victim and tool outputs remain in context but are not prediction targets. Every successful trajectory is retained, unsuccessful trajectories receive zero weight, and the optimizer step is skipped when the batch contains no success. OFC uses no KL penalty. We give the full loop in Algorithm 1.

Algorithm 1 Outcome-Faithful Consolidation (OFC)

Input: Initial policy π_{θ_0}; goal schedule; frozen victim; terminal verifier S; rollout budget G; iteration budget U
Output: Trained attacker policy π_{θ_U}

```
for k = 0, ..., U − 1 do
    Sample G trajectories per scheduled goal using π_{θ_k}
    Evaluate v(τ) = S_x(τ) for each trajectory
    Set N_k to the total generated attacker-response tokens, or 1 if none
    if at least one trajectory has v(τ) = 1 then
        Construct Eq. 3 using all admitted attacker responses
        Obtain θ_{k+1} with one gradient-clipped AdamW update; no KL penalty
    else
        Set θ_{k+1} = θ_k; skip the optimizer update
    end if
end for
```

The rule has a useful connection to the terminal objective. Under exact on-policy sampling in a fixed environment, and with the sampling distribution held fixed during differentiation, consider the success-weighted log-likelihood objective

$$\ell(\theta \mid \theta_k) = -\mathbb{E}_{\tau \sim p_{\theta_k}} \left[S(\tau) \log p_\theta(\tau) \right].$$

The score-function identity gives

$$-\nabla_\theta \ell(\theta \mid \theta_k)\big|_{\theta=\theta_k} = \nabla_\theta J_S(\theta_k). \tag{4}$$

We state the assumptions and derivation in Appendix B. We treat Eq. 4 as an explanatory identity, not a convergence theorem; the implemented method differs through finite batches, token normalization, temperature-adjusted sampling, and finite AdamW steps. Still, we take from it a clear reason to study verified-success learning: under the idealization, the update direction is aligned with the strict terminal objective rather than with a progress proxy.

OFC is therefore the destination of the reward argument. Sparse terminal feedback creates a signal problem; arbitrary dense replacement can solve the signal problem while changing the objective; potential-based shaping explains why such a replacement has no general invariance guarantee; and success-weighted likelihood provides a direct route back to the terminal objective. OFC turns that route into a practical training rule. Its limitation is equally clear: if the attacker never discovers a successful trajectory, there is nothing to consolidate. We therefore ask in the remaining experiments whether this design helps in practice, why it helps, and how long the advantage lasts.

4 Experiments

4.1 Experimental setup

Our main setting trains a full-parameter bf16 Qwen3.5-4B attacker against a frozen bf16 Qwen3.5-9B victim. We build the environment from AgentDojo components: banking, travel, and workspace tasks provide 24 training goals, and each goal asks the attacker to obtain $K = 4$ target values within at most $T = 5$ attacker turns. Only the victim executes tools. A programmatic verifier decides strict success (all four values in one qualifying outbound call), so no LLM judge enters training or evaluation. We evaluate attack success rate (ASR) on 24 held-out field combinations from the same three domains, eight per domain, that never appear in training; we call this metric **held-out ASR**. For each goal we draw 12 sampled rollouts under the training decoding settings, compute the goal's success rate, and average across goals. We read every arm at fixed, pre-declared iterations (iteration 16 in the original campaign, with diagnostic read-outs every two iterations; 8, 16, 24, and 32 in the later study) rather than at a best checkpoint, and we measure cross-domain transfer separately on eight Slack goals that the attacker never encounters during training.

To keep the comparison controlled, all arms share one training system and one set of hyperparameters: learning rate 2×10^{-6} with 8-bit AdamW, gradient-norm clipping at 1.0, a sampling

Table 1: Main paired outcomes for the 4B attacker and frozen 9B victim. A: iteration-16 campaign; B–C: the same mechanism runs at iterations 8 and 32. Sparse ASR and ASR are means ± standard deviations over each row's matched seeds. Δ is method minus Sparse with its 95% paired-t half-width, tinted blue when positive and rust when negative; bold and shading mark the OFC (ours) rows only. Wins counts positive seed differences (share bar). Colored squares mark method families (legend below); precedents are cited next to each method, adaptations described in Appendix A.2, and secondary metrics reported in Table 4.

Method	Sparse ASR (%) mean ± sd	ASR (%) mean ± sd	Δ ± 95% (pp)	Wins / pairs
(A) Original campaign, iteration 16				
Sparse (Shao et al., 2024)	–	43.71 ± 4.1	–	–
Dense (Shao et al., 2024)	46.01 ± 2.7	32.00 ± 5.0	−14.01 ± 6.29	0/6
Lower Threshold	46.60 ± 2.6	25.00 ± 2.2	−21.60 ± 1.24	0/5
Threshold Curriculum (Bengio et al., 2009)	46.01 ± 2.7	29.86 ± 4.7	−16.15 ± 4.28	0/6
Adaptive Allocation (Nguyen et al., 2026)	46.35 ± 3.0	45.31 ± 2.5	−1.04 ± 5.24	2/4
Success Replay (Oh et al., 2018)	46.35 ± 3.0	48.44 ± 5.5	+2.08 ± 4.46	3/4
More Rollouts (Hu et al., 2025)	46.38 ± 2.7	50.45 ± 3.7	+4.07 ± 4.02	5/7
OFC (ours)	**43.71 ± 4.1**	**48.31 ± 4.6**	**+4.60 ± 2.37**	**13/16**
(B) Mechanism study, iteration 8				
Sparse (Shao et al., 2024)	–	35.42 ± 8.1	–	–
OFC (ours)	**35.42 ± 8.1**	**42.56 ± 5.8**	**+7.14 ± 3.02**	**7/7**
Positive-only RL (Oh et al., 2018)	34.09 ± 8.0	43.06 ± 3.9	+8.97 ± 10.05	4/6
Unfiltered CE	–	15.28	–	–
(C) Same mechanism runs, iteration 32				
Sparse (Shao et al., 2024)	–	54.37 ± 4.5	–	–
OFC (ours)	**54.37 ± 4.5**	**51.19 ± 5.6**	**−3.18 ± 5.65**	**1/7**
Positive-only RL (Oh et al., 2018)	53.41 ± 4.1	52.89 ± 2.1	−0.52 ± 4.73	3/6
Unfiltered CE	–	12.85	–	–

Sparse reference · denser/relaxed signal · more/reused experience · positive-only RL · unfiltered CE · OFC (ours)

temperature of 0.7 for attacker and victim, at most 112 generated tokens per attacker response, and $G = 6$ rollouts per training goal (the More rollouts arm alone uses $G = 12$). One training iteration consists of one rollout batch and at most one optimizer update, so every arm generates the same number of trajectories per iteration, and rollout generation dominates wall-clock time in all of them. The RL arms use the same state-stratified group-relative estimator with a reference-policy KL coefficient of 0.02; we call the outcome-only baseline SPARSE. DENSE changes only the reward (Eq. 2), and OFC changes only the update rule (Eq. 3: unit trajectory weights and no KL term). We fix the victim model, precision, and system prompt within a campaign and use one data seed for the goal split in every arm.

Each run is identified by a model seed that fixes attacker-side sampling and goal order. We compare every method with SPARSE on the same seeds and report paired differences with 95% paired-t intervals, so seed-level variation is removed from each contrast. The original campaign evaluates the main interventions after 16 iterations, with SPARSE and OFC on 16 seeds and the other arms on four to seven seeds each. We later add POSITIVE-ONLY RL in a 32-iteration study on the same 4B/9B pair to test budget dependence, and we pair a 9B attacker with a frozen 27B-FP8 victim in a separate experiment. We summarize the main results in Table 1 and give the estimator, full hyperparameters, cohorts, hardware, and implementation details in Appendix A.

4.2 WHEN MORE FEEDBACK CHANGES THE OBJECTIVE

We begin with the natural response to sparse terminal rewards: provide more training signal. DENSE replaces the terminal reward with process progress, while the lower-threshold and curriculum arms relax the training target. If sparsity were the main obstacle, these interventions should make learning easier and improve strict success.

They do make learning easier in the narrow sense of producing more usable signal. Under DENSE, the fraction of goal groups without a relative-update signal falls from 0.470 to 0.122 on matched seeds, the number of nonzero-weight examples rises from 132 to 232 per iteration, and held-out process score increases from 0.603 to 0.653 at iteration 16. Yet strict held-out ASR moves in the opposite direction, from 0.460 to 0.320: DENSE loses 14.01 percentage points relative to SPARSE,

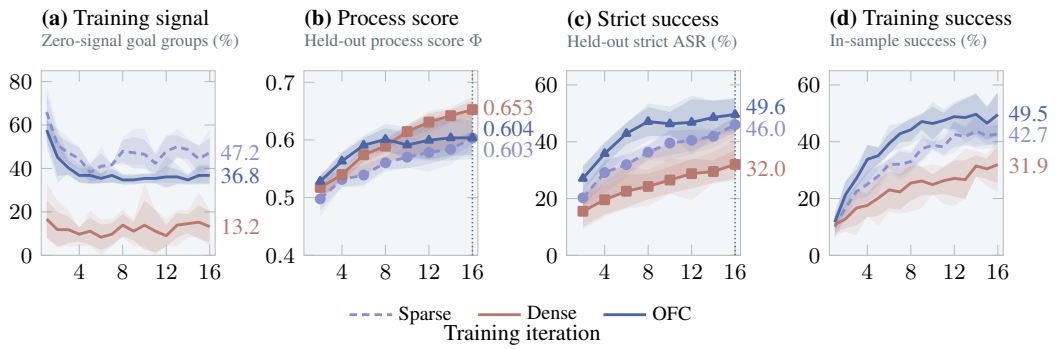


Figure 3: More learning signal does not imply greater strict success. Matched seeds 2–7 of the original campaign. (a) Goal groups without a nonzero-weight response and (d) in-sample training success at every iteration; (b) held-out tool-chain progress and (c) strict held-out attack success at the read-outs taken every two iterations. Bands: spread across seeds (lighter: min–max; darker: mean ± sd); thick lines: means; labels: the iteration-16 values used in Table 1 (dotted guide). The shaded Dense–Sparse gap opens in opposite directions: Dense raises the process score by 0.05 while lowering ASR by 14.0 points, and its in-sample success is lower at every iteration after the first. OFC: same six seeds (16-pair result in Table 1; more diagnostics in Appendix D.1).

and the lower-threshold and curriculum arms lose 21.60 and 16.15 points. We show in Figure 3 that the divergence between process score and strict success develops during training and is already visible in-sample from the second iteration on. We read this as the empirical counterpart of Section 3.2: added signal does not preserve the strict objective. We do not conclude that every dense reward is harmful, only that rewarding more partial progress is not enough when proxy and outcome favor different behavior.

A second option preserves strict success and increases the supply of useful experience. Doubling the rollout budget gives a nominal $+4.07$-point effect, while adaptive allocation and success replay have intervals spanning zero; replay can only reuse a success after one has been found. We therefore ask how verified successes are turned into updates. OFC answers this directly: at iteration 16 it raises held-out ASR from 43.71% to 48.31% across 16 matched seeds, a gain of 4.60 points (95% CI $[2.23, 6.97]$) with 13 of 16 paired differences positive. Because the 16-seed estimate pools the initial screen and added runs, we do not treat it as a fully independent replication; it nevertheless establishes the main finite-budget contrast of the paper: **preserving the strict objective and learning directly from verified successes performs better in this setting than replacing the objective with denser progress feedback**.

4.3 Why does OFC help early?

We next ask a mechanistic question. OFC changes both the examples that receive positive weight and the form of the update. Is the early gain specific to cross-entropy training, or does outcome-conditioned selection itself matter?

We probe this with Positive-only RL. It keeps the sparse RL estimator but clips negative advantages to zero, so only positive-advantage responses carry policy-gradient weight. OFC instead gives unit weight to every attacker response in a verified successful trajectory. An unfiltered CE arm provides a descriptive comparison without outcome-based selection.

We isolate the response-weight distinction in Figure 4 on a logged goal group: a verified-successful trajectory can contain zero-advantage responses, including the one that completed the attack, which Positive-only RL leaves unused but OFC still supervises.

At iteration 8, OFC exceeds its matched Sparse baseline by 7.14 points across seven pairs (Table 1(B)). Positive-only RL shows a similar early pattern, improving by 8.97 points over its six-seed Sparse subset. The Positive-only RL estimate is less stable: its bootstrap interval excludes zero, while its paired t-test gives $p = 0.070$. We therefore do not treat the two methods as equivalent, and we draw the narrower conclusion that the OFC gain is consistent with a role for outcome-conditioned update selection and is not obviously unique to supervised fine-tuning.

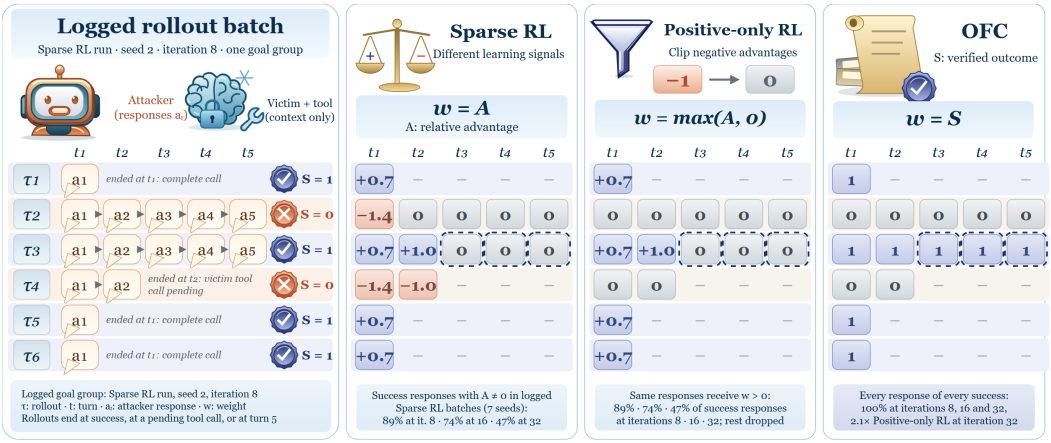


Figure 4: The same logged trajectories supply different learning signals. One goal group (six rollouts, four successful) from a logged Sparse RL batch (seed 2, iteration 8). Cells: rounded policy-loss weight w under each rule, recomputed with the project's estimator (A: state-stratified advantage; S: verified outcome). Dashed outlines: responses of successful trajectories with $A = 0$, which both RL rules leave unweighted and OFC supervises; in τ_3 they include the completing call. Bottom boxes: share of success responses with nonzero weight over all seven seeds' logged Sparse RL batches (89%, 74%, 47% at iterations 8, 16, 32); OFC uses all. KL and token normalization omitted; provenance in Appendix A.1.

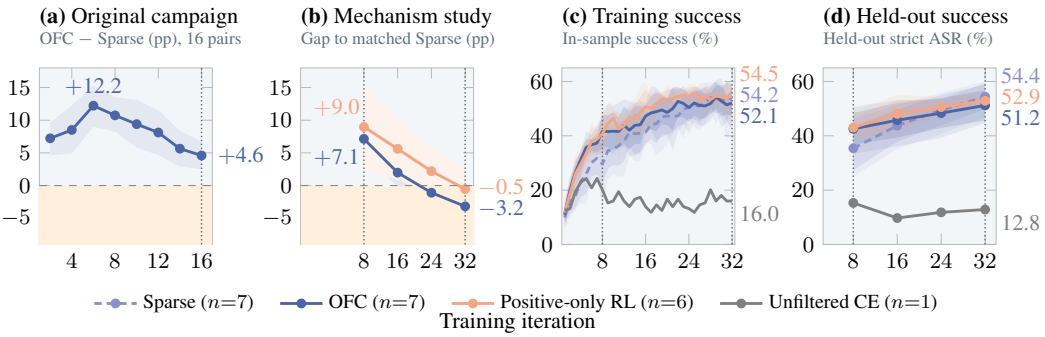


Figure 5: Budget dependence of OFC and POSITIVE-ONLY RL. (a) OFC minus Sparse across 16 matched pairs in the original campaign, every two iterations; (b) paired gaps to matched Sparse in the 32-iteration mechanism cohort; (c) in-sample training success of that cohort at every iteration; (d) its held-out ASR every eight iterations. Bands: spread across runs (lighter: min–max; darker: mean ± sd); in (a) and (b), pointwise seed-bootstrap 95% bands, region below zero shaded. Dotted guides: iteration 16 in (a), iterations 8 and 32 in (b)–(d). Both methods gain most early, in-sample and held-out, and Sparse closes the gap at longer budgets.

POSITIVE-ONLY RL does not isolate every factor: it also changes which KL terms remain, and the unfiltered CE arm has only one complete run. We therefore treat this study as a mechanism probe rather than a complete causal ablation.

4.4 THE ADVANTAGE DEPENDS ON TRAINING BUDGET

If outcome-conditioned selection mainly improves the use of scarce early successes, its advantage need not persist once SPARSE has more opportunities to learn. We observe exactly this pattern in the longer runs (Figure 5). In the original campaign, the OFC–Sparse gap is largest at intermediate checkpoints and is already smaller by iteration 16. In the complete 32-iteration mechanism cohort, the gap moves from $+7.14$ points at iteration 8 to -3.18 points at iteration 32. The paired decline is 10.32 points and occurs in all seven seeds, although the endpoint interval still includes zero.

POSITIVE-ONLY RL shows a related late-budget limitation. From iterations 16 to 32, both OFC and POSITIVE-ONLY RL improve less than their SPARSE controls. We read this in the same way as the early mechanism result: selectively reinforcing favorable experience can accelerate learning when successful trajectories are scarce, but it does not guarantee a permanent relative advantage. Importantly, OFC's absolute ASR continues to increase over the same interval, so we read the late

Table 2: Results for the 9B attacker and frozen 27B-FP8 victim. Held-out ASR is the mean ± standard deviation across seeds. Δ: paired difference with its 95% paired-t half-width; Rel. Δ: relative to matched Sparse; p: unadjusted paired test; Wins: positive seed differences; Examples: nonzero-weight attacker responses per iteration. Slack successes are omitted because every arm records zero.

Method	Held-out ASR (%) mean ± sd	Δ vs SPARSE ± 95% (pp)	Rel. Δ (%)	p	Wins	Held-out Φ	Examples / iteration	Zero-signal groups (%)	Train succ. (%)
Iteration 8									
SPARSE (ref.)	15.45 ± 2.0	–	–	–	–	0.357	50	73.0	13.2
Dense	9.63 ± 3.0	−5.82 ± 6.78	−37.6	0.0720	0/4	0.481	158	11.6	9.2
OFC	**19.36 ± 4.0**	**+3.91 ± 4.17**	**+25.3**	**0.0584**	**4/4**	**0.360**	**26**	**70.3**	**15.0**
Iteration 16									
SPARSE (ref.)	18.23 ± 1.6	–	–	–	–	0.359	52	71.2	16.8
Dense	15.45 ± 3.7	−2.77 ± 5.51	−15.2	0.2072	1/4	0.583	145	19.0	11.7
OFC	**20.31 ± 4.0**	**+2.08 ± 4.36**	**+11.4**	**0.2257**	**3/4**	**0.344**	**31**	**68.9**	**18.7**

result as SPARSE catching up rather than OFC deteriorating. A diversity-contraction explanation remains possible, but we do not measure that mechanism directly.

4.5 MODEL-PAIRING AND TRANSFER BOUNDARIES

We finally ask whether the main effect carries to a different model configuration. We train a full-parameter Qwen3.5-9B attacker against a frozen Qwen3.8-27B-FP8 victim after a feasibility screen in which the 4B attacker produced no measured strict held-out success at iteration 4. Because attacker size, victim model, and victim precision all change, we treat this as a model-pairing study rather than a controlled scaling experiment.

At iteration 16, DENSE has a -2.77-point effect and OFC a $+2.08$-point effect relative to SPARSE (Table 2), but neither confidence interval excludes zero. The signs match the main experiment, but the evidence is too weak to claim that the original effect transfers unchanged. Cross-domain ASR on Slack is reported separately (Appendix D.2) because it can differ from held-out combinations within the training domains. With the budget study, this bounds our claim to a finite-budget benefit in the main setting, not universal dominance across models, tasks, or horizons.

4.6 LIMITATIONS

Our conclusions rest on benchmark-derived goals and few model pairings, so the seed-level intervals measure variation within those settings, not across all cybersecurity tasks. The mechanism study is a probe, not a full factorial ablation. OFC consolidates discovered successes but does not itself address exploration, and its advantage over outcome-only RL narrows with longer training.

5 CONCLUSION

Multi-turn cybersecurity red-teaming creates a tension between learning signal and objective fidelity: strict terminal rewards preserve the security objective but are sparse, and dense process rewards relieve the sparsity at the risk of redefining success. We show that this risk is real. Replacing the terminal reward with process progress raised held-out process scores while lowering complete attack success by 14 points, whereas keeping the verifiable outcome fixed and learning from verified successes with OFC improved held-out success by 4.6 points across 16 paired seeds at iteration 16; POSITIVE-ONLY RL reproduced the early gain, and longer training showed SPARSE catching up. We therefore draw a conditional conclusion: **for multi-turn red-teaming agents, discovery and consolidation should be treated as separate problems**, with consolidation preserving the terminal objective rather than replacing it with a proxy.

Future work has three directions: exploration strategies that surface the first complete attack chain per target, which OFC can then consolidate; longer horizons, more model pairings, and direct measurement of trajectory diversity to determine when selective consolidation stops paying off; and other multi-turn security objectives with programmatic terminal verifiers, such as privilege escalation or lateral movement, whose verified attack trajectories can also harden the agents under test.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods and experiment code, to clean and reformat experimental records, to interpret results, and to provide feedback on experimental design. We have not used generative AI tools to generate synthetic data sets, to develop the theoretical framework, to formulate mathematical claims or assist with their derivations, or to propose or refine hypotheses; translation and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for literature retrieval, to draft parts of the manuscript, and to create the illustrations in Figures 1, 2, and 4. We have reviewed all AI-assisted work: the training and analysis code is covered by unit tests and its logged metrics were replicated from the raw traces (Appendix A.1); every reported number is computed by scripts from stored run records whose input files are hashed; and all references were checked against their published versions. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We study the controlled evaluation of tool-agent security using benchmark-derived environments and designated test data. We do not measure unauthorized access to production systems. Automated attack training has dual-use implications; the scientific focus here is the reliability of training objectives and evaluation. We report aggregate results and formal training procedures without operational attack payloads.

REPRODUCIBILITY STATEMENT

Code, run records, and figure sources are in an anonymized repository:

https://anonymous.4open.science/r/when-progress-is-not-success-0F02

It contains the training, serving, and evaluation code, the per-run records (configuration and per-iteration metrics) behind every table and figure, the result snapshot with input-file hashes, and the scripts that regenerate the result plots and tables. We derive the snapshot from existing experiment records, not new training, and we document cohorts, evaluation, analysis choices, and artifact limitations in Appendix A. Raw per-turn attack traces are withheld from the public release because they contain complete attack conversations.

REFERENCES

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 41–48, 2009. doi: 10.1145/1553374.1553380. URL https://icml.cc/2009/papers/119.pdf.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *The Thirty-eighth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=m1YYAQjO3w.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 10835–10866, 2023. URL https://proceedings.mlr.press/v202/gao23h.html.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec '23)*, pp. 79–90, 2023. doi: 10.1145/3605764.3623985.

Jian Hu, Mingjie Liu, Ximing Lu, Fang Wu, Zaid Harchaoui, Shizhe Diao, Yejin Choi, Pavlo Molchanov, Jun Yang, Jan Kautz, and Yi Dong. BroRL: Scaling reinforcement learning via broadened exploration, 2025. URL https://arxiv.org/abs/2510.01180.

He Kong, Die Hu, Jingguo Ge, Liangxiong Li, Hui Li, and Tong Li. Pentest-R1: Towards autonomous penetration testing reasoning optimized via two-stage reinforcement learning, 2025. URL https://arxiv.org/abs/2508.07382.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Qiang Liu, Taian Guo, Ruizhi Qiao, and Xing Sun. RSPO: Reward-swap policy optimization for multi-turn LLM agents, 2026. URL https://arxiv.org/abs/2607.04713.

David Manheim and Scott Garrabrant. Categorizing variants of Goodhart's law, 2018. URL https://arxiv.org/abs/1803.04585.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*, pp. 278–287, 1999.

Hieu Trung Nguyen, Bao Nguyen, Wenao Ma, Yuzhi Zhao, Ruifeng She, and Viet Anh Nguyen. Adaptive rollout allocation for online reinforcement learning with verifiable rewards. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2602.01601.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3878–3887, 2018. URL https://proceedings.mlr.press/v80/oh18b.html.

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 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3419–3448, 2022. doi: 10.18653/v1/2022.emnlp-main.225. URL https://aclanthology.org/2022.emnlp-main.225.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th International Conference on Machine Learning*, pp. 745–750, 2007. doi: 10.1145/1273496.1273590.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=dHng2O0Jjr.

Mark Russinovich, Ahmed Salem, and Ronen Eldan. Great, now write an article about that: The Crescendo multi-turn LLM jailbreak attack. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 2421–2440, Seattle, WA, 2025. USENIX Association. URL https://www.usenix.org/conference/usenixsecurity25/presentation/russinovich.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, pp. 68539–68551, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, et al. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=lNAyUngGFK.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, pp. 9460–9471, 2022.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models, 2023. URL https://arxiv.org/abs/2308.01825.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10471–10506, 2024.

A Experimental Protocol and Evidence Scope

A.1 Task and implementation details

We reuse AgentDojo-derived banking, travel, workspace, and Slack components, with project-specific field-combination goals and a strict single-call success verifier. The attacker submits user-role messages to the victim pipeline and receives textual feedback about the victim and its tool calls. We therefore study a direct interactive attack setting built from AgentDojo components, rather than the standard indirect-prompt-injection benchmark. The primary split contains 24 training goals and 24 held-out combinations, eight of each per training domain. Slack contributes eight separate transfer goals. We fix the data split seed at zero; model seeds change training/evaluation sampling, not the goal split. Each iteration uses the 24 training goals; the default allocation gives each goal G attempts.

Table 3 lists the shared training configuration. In both the original and the mechanism study, we train Qwen3.5-4B in bf16 against a frozen Qwen3.5-9B victim in bf16. For the original campaign we use an eight-A800-SXM4 allocation with separate training and serving lanes; for the subsequent study we use five A800-PCIe lanes, each colocating a trainee with a victim server. These are separate campaigns, not additional stages of a single uninterrupted run. For the larger pairing we use a 9B bf16 attacker and a frozen Qwen3.8-27B-FP8 victim on H20 hardware. We control victim model and precision within a campaign; they change between pairings.

Table 3: Shared principal training configuration. We state arm-specific departures below and include recorded run metadata in the data snapshot.

Setting	Value
Strict success / maximum attacker turns	$K = 4$ / $T = 5$
Training goals / held-out goals / transfer goals	24 / 24 / 8
Rollouts per training goal	$G = 6$; $G = 12$ for More rollouts
Learning rate / optimizer	2×10^{-6} / 8-bit AdamW
Optimizer betas / gradient norm clipping	$(0.9, 0.999)$ / 1.0
Reference KL coefficient	0.02 for RL; 0 for CE arms
Maximum generated attacker tokens per response	112
Original iteration / evaluation schedule	16 iterations / every 2
Mechanism iteration / evaluation schedule	32 iterations / every 8
Larger-pairing iteration / evaluation schedule	16 iterations / every 8
Evaluation rollouts per goal	12

For the group-relative RL baseline, let $R_{i,t}$ be the undiscounted return-to-go for trajectory i at position t. We partition trajectories by goal, turn position, and recorded progress state, then compute

$$A_{i,t} = \frac{R_{i,t} - \overline{R}_{c(i,t)}}{\sigma_{c(i,t)} + \epsilon}. \tag{5}$$

Here $c(i,t)$ groups responses by goal, turn position, and the preceding process score Φ_{t-1}; $\overline{R}_c$ and σ_c are that cell's mean and population standard deviation, and $\epsilon = 10^{-6}$. Singleton and zero-spread cells have zero advantage. The corresponding turn response receives token-normalized score-function updates and a reference penalty. For a sampled token define $d = \log \pi_{\text{ref}} - \log \pi_\theta$; the implemented penalty is $e^d - d - 1$. The policy term is $-A_{i,t} \log \pi_\theta$. We take one optimizer step per sampled batch, using gradient accumulation over small chunks of response examples. We implement group-relative policy gradient without a PPO clipped-ratio surrogate in the reported arms.

POSITIVE-ONLY RL replaces $A_{i,t}$ by $\max(A_{i,t}, 0)$. Only nonzero-weight examples enter the shared loss routine, so clipping also removes their associated KL contributions. POSITIVE-ONLY RL retains the RL estimator and KL coefficient on the remaining examples, but does not isolate sign clipping under a globally identical regularization term. OFC instead uses binary trajectory weights and zero KL coefficient. The unfiltered CE arm assigns unit weight to all generated trajectories. We fix the outcome verifier across these mechanism arms.

Provenance of the response-weight example. Figure 4 shows one goal group of a logged batch: the second of the 24 groups sampled at iteration 8 by the Sparse RL run with seed 2 in the mechanism

study, six rollouts of which four succeeded. We recompute the weights from the stored per-turn traces with the training code's estimator: the terminal reward fires at the turn whose single-send count trace first reaches K, returns-to-go are compared within cells keyed by turn position and the preceding process score Φ_{t-1}, and the displayed advantages are rounded to one decimal. At turn 1 all six rollouts share the cell $\Phi_0 = 0$, which gives $+0.71$ to the four successes and -1.41 to the two failures. At turn 2 the partition is $\{\tau_2\}$ with $\Phi_1 = 0.5$ and $\{\tau_3, \tau_4\}$ with $\Phi_1 = 0.75$, so τ_3 receives $+1.0$ and τ_4 receives -1.0. From turn 3 on, τ_2 and τ_3 occupy singleton cells, so their remaining responses have zero advantage, including turn 5 of τ_3, where the complete call occurred. Rollouts end at success, when the victim leaves a tool call pending, or after the fifth turn. The cohort counts in the figure apply the same recomputation to every logged batch of the seven Sparse RL seeds; at every one of the 32 iterations, the recomputed fraction of zero-advantage decisions equals the value logged during training. We use the example to show a difference in admitted responses; we do not claim that the three training procedures would produce identical batches.

We use the other original-campaign interventions to test distinct explanations. The lower-threshold and curriculum arms change the threshold on the training anchor, with the historical qualification below. Adaptive allocation redistributes a fixed episode budget across goals according to an estimated $\sqrt{p(1-p)}$ weight. Success replay adds previously successful trajectories for the same goal when its current relative-update group is starved. We adapted this replay control from self-imitation, but it is not a direct reproduction of the actor-critic SIL algorithm (Oh et al., 2018). The More rollouts arm sets $G = 12$ and therefore does not match rollout compute.

A.2 Methodological precedents for the comparison arms

We cite related methodological precedents in Table 1 where applicable. We measure all numbers in our environment with our implementation; the citations do not denote copied benchmark scores or exact reproductions of the cited systems.

Sparse and Dense: outcome versus process supervision. DeepSeekMath introduces group-relative policy optimization with both outcome and process supervision (Shao et al., 2024); process supervision also motivates step-level verification (Lightman et al., 2024). Our SPARSE baseline uses a strict binary outcome and state-stratified group-relative advantages, with the score-function loss specified above. Dense changes the reward to deterministic tool-chain progress increments (Eq. 2). It uses neither a learned process reward model nor the original GRPO clipped-ratio objective. Thus these are task-specific outcome/process reward implementations of the group-relative learning family.

Lower threshold and threshold curriculum: relaxing the task. Curriculum learning motivates progressing from easier criteria to the target criterion (Bengio et al., 2009). Our threshold curriculum applies this idea to the training anchor, raising the required threshold from one to four over training. Lower threshold is its fixed-relaxation control, keeping the threshold at two. We give the exact schedule and historical fallback in Appendix A.3. These controls change the training criterion rather than reproduce a published curriculum system; strict evaluation remains unchanged.

Adaptive allocation: a VIP-inspired sampling control. VIP predicts per-prompt success probabilities and allocates a fixed rollout budget using variance information (Nguyen et al., 2026). Our lightweight adaptation estimates each goal's success rate from recent experience and allocates proportionally to $\sqrt{\hat{p}(1-\hat{p})}$, with a minimum of two rollouts per goal. It omits VIP's Gaussian-process predictor and optimization procedure. The total episode budget equals that of SPARSE.

Success replay and positive-only RL: self-imitation controls. Self-Imitation Learning (SIL) reinforces past actions with positive return-minus-value advantages (Oh et al., 2018). Success replay adapts the experience-reuse idea by inserting stored, verified successes into starved goal groups. Positive-only RL instead clips our current group-relative advantages at zero. Neither is the original off-policy actor-critic SIL algorithm; they test reuse and positive update selection within the shared training system.

OFC and unfiltered CE: outcome-selected self-training. RFT and ReST$^{\mathrm{EM}}$ filter generated solutions by correctness and fit the retained examples (Yuan et al., 2023; Singh et al., 2024). OFC realizes this principle online for complete multi-turn attack trajectories, with all attacker responses supervised and the full-batch token denominator in Eq. 3. Unfiltered CE is our no-selection ablation: it uses all generated trajectories. It is not itself a rejection-sampling baseline.

More rollouts: a sampling-budget control. BroRL studies broader exploration by increasing the number of rollouts per example (Hu et al., 2025), providing a related precedent for this sampling intervention. This arm retains the SPARSE learning rule and doubles the per-goal group size from six to twelve. It tests the sampling explanation within the same group-relative framework (Shao et al., 2024); it does not reproduce BroRL's training system or its much larger rollout budgets.

A.3 PROCESS SCORES AND HISTORICAL TRAINING THRESHOLDS

We compute the process score from cumulative tool events, not from the attacker's text alone. For a banking goal with L_x distinct required read calls, it is the fraction of the $L_x + 1$ reference calls matched: the reads plus a send containing all target values. Read order is unconstrained. Travel assigns half the score to a user-information read and half to an email containing all target values at the designated recipient. Workspace and Slack likewise split credit between a relevant read and an outbound call containing all values; their process scorers check the call's content without checking its destination. These distinctions are another reason not to equate $\Phi = 1$ with the intended security outcome in every domain.

In the archived RL code, we threshold a training anchor b_t: the best single-send field-count fraction if its trace is available, otherwise Φ_t. Sparse uses threshold 1. The lower-threshold arm uses $2/4$; the curriculum uses $m_k/4$, where m_k rises from 1 to 4 by rounding $1 + 3(k-1)/(U-1)$ for iterations $k = 1, \ldots, U$. Before the count-trace repair, travel and workspace used the fallback. A successful read alone could therefore trigger the $2/4$ training reward in those domains, even with no successful outbound delivery. The historical arms test these implemented reward relaxations, not a uniform two-field objective or a clean field-count curriculum. OFC instead uses the terminal selection check that we describe in Appendix A.5.

A.4 COHORTS, UNCERTAINTY, AND ANALYSIS HISTORY

In the original campaign we completed the following cohorts: sparse and OFC, seeds 2–17; dense and curriculum, seeds 2–7; lower threshold, seeds 2–6; adaptive allocation and replay, seeds 2–5; and More rollouts, seeds 2–8. We compute every table contrast on the intersection with sparse at the exact requested iteration, with no fallback from a missing evaluation to a different iteration.

For the original and larger-pairing studies, we report 95% paired Student-t intervals across seed differences. Historical tables used approximate intervals; we recompute them for this draft from the logged per-run metrics. We log evaluation metrics rounded to four decimal places, which can affect the last displayed digit. The initial exploratory family contained six alternatives: lower threshold, curriculum, adaptive allocation, replay, More rollouts, and success-filtered SFT. Its positive contrasts did not survive the recorded Holm correction. The dense deficit is a separate reference comparison outside this family. We present unadjusted intervals in Table 1(A) for estimation, not as family-wise significance declarations.

We subsequently expanded the OFC cohort. Our result summaries describe the added seeds as a dedicated follow-up, but the 16-seed estimate pools both stages. To make that distinction transparent, the initial eight seeds have an effect of $+4.12$ points (CI $[-0.32, 8.57]$, $p = 0.064$); the added eight have $+5.08$ points (CI $[1.99, 8.16]$, $p = 0.006$). We recompute these subgroup summaries descriptively for this draft and do not relabel them as newly preregistered tests or as proof of independence from every exploratory decision.

In the mechanism study we completed sparse and OFC on seeds 2–8, positive-only RL on seeds 2–7, and unfiltered CE on seed 2. We do not substitute incomplete or interrupted runs for complete 32-iteration runs; some have useful earlier evaluations but are excluded from the fixed complete-run cohort. In total, we archive 26 run directories, not 26 complete runs. We retain all completed seeds in the primary presentation, including runs with stalled learning curves. Our original analysis

also includes a preregistered stalled-run exclusion and a later revised exclusion; we do not use their stronger endpoint conclusions for the main claim.

For the mechanism study we use the saved 20,000-resample seed-bootstrap intervals. For a pair of arms, we retain the same seed pairing when forming a difference or a difference of increments. The paired decay is $[\mathrm{ASR}_A(8) - \mathrm{ASR}_B(8)] - [\mathrm{ASR}_A(32) - \mathrm{ASR}_B(32)]$. We keep the previously saved bootstrap intervals in our plots for this study and add descriptive pointwise bootstrap intervals, computed with the project's bootstrap routine, to the original-campaign panel of Figure 5. These are not simultaneous confidence bands across all budgets.

A.5 Artifact and evaluation limitations

We use the strict $m = K = 4$ readout in the main paper: threshold the best single-send count at K when available, otherwise use strict episode success. A historical fallback affected lower-threshold evaluation in some domains. When a single-send count trace was absent, the evaluator used strict episode success even if a lower threshold was requested. The fallback retains the strict threshold but makes the historical $m < 4$ columns unsuitable for difficulty-trend claims, so we omit those columns. Our later study includes the count-trace correction, and the strict criterion remains the common comparison.

Some early evaluation response traces are unavailable for independent replay, although their aggregated progress records remain available. We preserve these metrics and their source hashes in the data snapshot, but we cannot reconstruct missing generations. Matching seeds improves comparability but does not guarantee identical victim sampling under asynchronous serving. We document run interruptions and model/hardware changes in the experiment records.

B Derivation of the Success-Weighted Identity

For clarity, fix a goal x and suppress it below. Averaging the result over a fixed goal distribution μ recovers the objective in Eq. 1. Fix an environment transition law independent of θ. Assume a differentiable trajectory distribution with parameter-independent support and regularity permitting differentiation under the integral. Let the verifier $S(\tau) \in \{0, 1\}$ also be independent of θ. Then

$$\nabla_\theta J_S(\theta) = \nabla_\theta \int S(\tau) p_\theta(\tau)\, d\tau \tag{6}$$

$$= \int S(\tau) p_\theta(\tau) \nabla_\theta \log p_\theta(\tau)\, d\tau. \tag{7}$$

In a fixed environment, the parameter-dependent factors in $p_\theta(\tau)$ are the attacker's action probabilities; victim and environment terms contribute no score gradient. Differentiating the success-weighted log-likelihood $\ell(\theta \mid \theta_k)$ of Section 3.3 while freezing p_{θ_k} and then evaluating at θ_k gives Eq. 4. We thus relate outcome-filtered likelihood updates to reward-weighted learning, under the stated idealization.

If one instead normalizes over successful trajectories, $q_k(\tau) = p_{\theta_k}(\tau) S(\tau)/J_S(\theta_k)$, with $J_S(\theta_k) > 0$, its expected score equals $\nabla J_S(\theta_k)/J_S(\theta_k) = \nabla \log J_S(\theta_k)$. Neither expression guarantees an increase in held-out success after a finite AdamW update. The implemented token denominator, decoding distribution, and parameter sharing further limit a literal equivalence claim.

For comparison, without reward weighting, exact on-policy sampling gives $\mathbb{E}_{p_\theta}[\nabla \log p_\theta(\tau)] = \nabla \int p_\theta(\tau) d\tau = 0$. Thus unweighted, exact on-policy self-training has zero expected score gradient under this idealization; it supplies no expected direction toward the success objective. We do not claim zero parameter drift in the actual finite-sample implementation.

B.1 Potential-based shaping versus proxy replacement

For a fixed horizon H, state z_t, discount $\gamma \in (0, 1]$, and potential Ψ, adding $F_t = \gamma \Psi(z_{t+1}) - \Psi(z_t)$ to the original reward gives

$$\sum_{t=0}^{H-1} \gamma^t F_t = -\Psi(z_0) + \gamma^H \Psi(z_H). \tag{8}$$

With a fixed initial-state distribution and a policy-independent terminal term, such as a zero terminal potential, the shaped return differs from the original return by a constant, so the addition preserves policy preferences (Ng et al., 1999). The dense arm's reward in Eq. 2 replaces the outcome reward with progress increments rather than adding such a term to it, so the invariance result does not apply to that arm. The invariance result also says nothing about the speed of optimization or about held-out performance under a finite budget.

C Conceptual Illustration and Study Logic

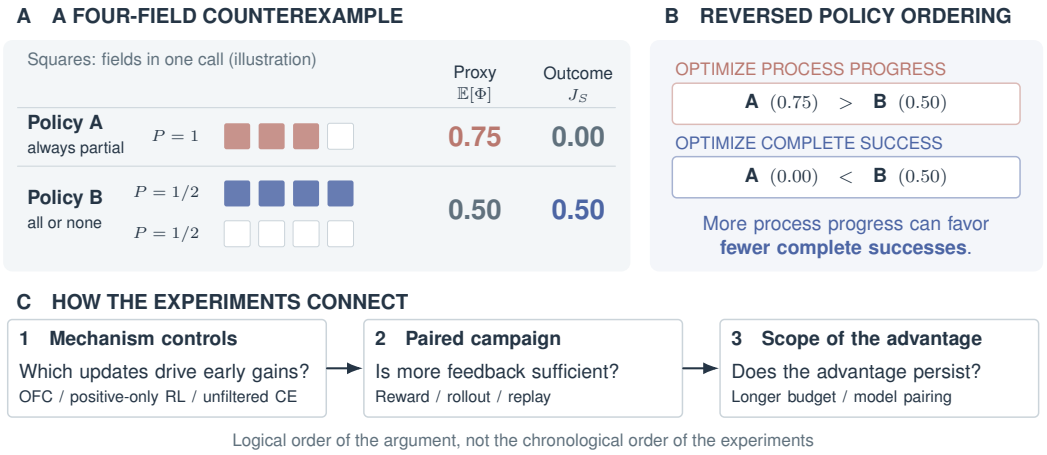


Figure 6: A conceptual illustration of the reward argument. In panels A–B we visualize the constructed example of Section 3.2, using the fraction of fields delivered in one call as an illustrative proxy; this is not the implemented chain score, and the policies are not measured trajectories. In panel C we connect update-selection controls, the broader paired campaign, and the budget/model-pairing studies in the order of the paper's argument. We completed the mechanism study later than the original campaign, so the diagram follows the argument rather than the chronological order.

D Additional Results

D.1 Auxiliary Training Diagnostics

We collect in Figure 7 the four auxiliary panels omitted from Figure 3, which itself shows in-sample training success in panel (d). We use the same six seeds as in Figure 3 for all curves. Training-loss magnitudes are descriptive: the RL and CE arms optimize different losses and their values are not directly comparable measures of task quality. We preserve in Table 4 the secondary readouts and unadjusted tests for every row of Table 1.

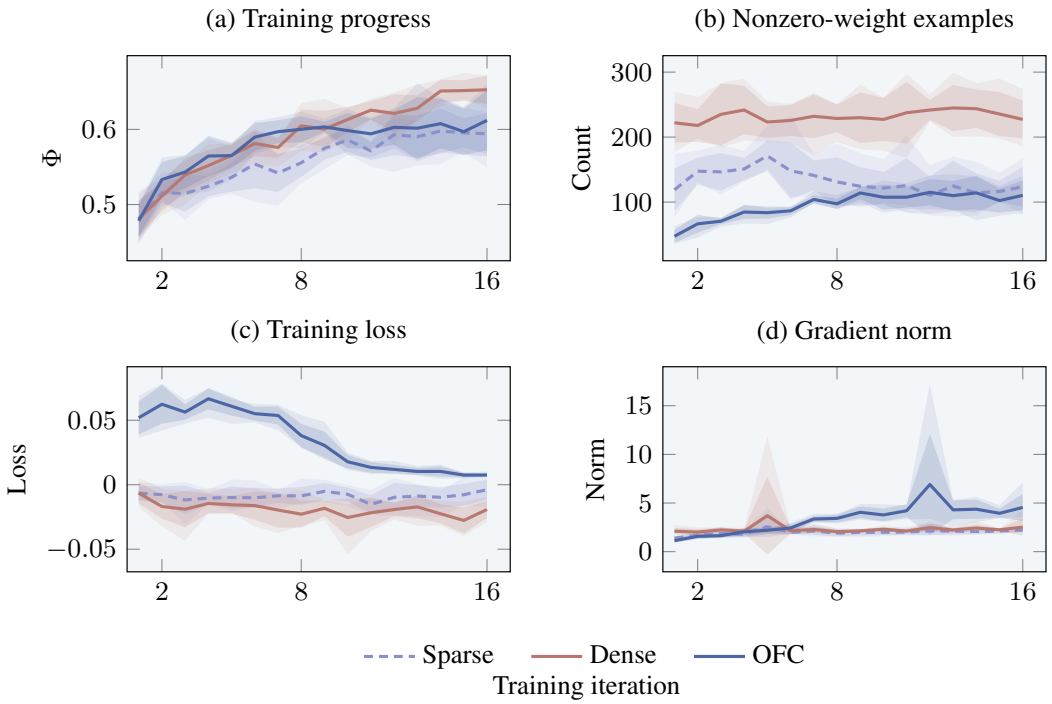


Figure 7: Auxiliary diagnostics for Sparse, Dense, and OFC on matched seeds 2–7. We shade the spread across seeds (lighter: min–max; darker: mean ± sd) and draw means as thick lines. Training process scores, nonzero-weight response counts, losses, and gradient norms complement the signal–progress–outcome comparison in Figure 3. Loss values have different meanings across RL and CE objectives.

Table 4: Secondary metrics for Table 1. Relative gains and p-values use each arm's matched Sparse subset. We measure held-out Φ and cross-domain ASR (Slack) at the stated checkpoint and average zero-signal groups and training success over iterations up to that checkpoint. The unfiltered CE arm has one complete run, so we report no paired inference for it.

Method	Rel. Δ (%)	p	Held-out Φ	Cross-domain ASR, Slack (%)	Zero-signal groups (%)	Train succ. (%)
(A) Original campaign, iteration 16						
Sparse	–	–	0.591	10.22	46.9	31.3
Dense	−30.4	0.0023	0.653	11.80	12.2	23.4
Lower Threshold	−46.3	<0.0001	0.622	9.79	39.6	26.5
Threshold Curriculum	−35.1	0.0002	0.616	9.03	40.1	26.2
Adaptive Allocation	−2.2	0.5724	0.597	7.29	43.8	39.9
Success Replay	+4.5	0.2335	0.617	10.16	46.9	34.7
More Rollouts	+8.8	0.0479	0.618	11.61	39.0	37.1
OFC (ours)	**+10.5**	**0.0009**	**0.604**	**15.62**	**38.7**	**39.3**
(B) Mechanism study, iteration 8						
Sparse	–	–	0.564	11.61	49.7	23.0
OFC (ours)	**+20.2**	**0.0012**	**0.592**	**9.38**	**42.1**	**30.0**
Positive-only RL	+26.3	0.0702	0.606	14.06	47.6	29.7
Unfiltered CE	–	–	0.439	5.21	0.0	19.7
(C) Same mechanism runs, iteration 32						
Sparse	–	–	0.640	11.61	50.5	40.4
OFC (ours)	**−5.8**	**0.2184**	**0.621**	**13.69**	**36.5**	**43.8**
Positive-only RL	−1.0	0.7873	0.630	14.93	51.1	45.9
Unfiltered CE	–	–	0.463	11.46	0.0	16.5

D.2 Cross-domain ASR (Slack) is a separate evaluation

We report descriptive paired results on the eight Slack goals in Table 5. The model never trains on these goals. We report Slack separately to keep held-out-combination results from being conflated with transfer to a different domain. We report unadjusted paired t intervals; they are not a separate preregistered family of discoveries. Both larger-pairing arms and their sparse controls record zero Slack successes here; their degenerate seed variance does not provide a useful uncertainty interval, so we omit those intervals.

Table 5: Cross-domain ASR (Slack). We list each method's matched sparse subset and mean and, in the last column, the method-minus-sparse difference with its 95% paired-t half-width, in percentage points (blue: improvement, rust: loss; OFC rows shaded). We omit intervals where every run recorded zero successes.

Study, iteration	Method	Sparse (%)	Method (%)	$\Delta \pm 95\%$ (pp)
Original, 16	Dense	9.55	11.80	$+2.25 \pm 3.94$
Original, 16	**OFC**	**10.22**	**15.62**	$\mathbf{+5.40 \pm 6.66}$
Mechanism, 32	**OFC**	**11.61**	**13.69**	$\mathbf{+2.08 \pm 8.74}$
Mechanism, 32	Positive-only RL	10.59	14.93	$+4.34 \pm 5.80$
Pairing, 16	Dense	0.00	0.00	–
Pairing, 16	**OFC**	**0.00**	**0.00**	–

D.3 Original campaign: per-seed strict success

Table 6: Original campaign held-out ASR (%) at iteration 16. A dash means that the seed is outside that method's reported complete cohort. We derive all means and paired contrasts in Table 1(A) from these entries.

Seed	Sparse	Dense	Lower	Curric.	Adaptive	Replay	$G = 12$	OFC
2	42.01	30.21	21.18	22.57	43.75	40.28	52.43	**54.51**
3	47.57	32.29	25.35	35.76	48.61	52.08	55.90	**45.83**
4	48.61	26.04	25.69	26.39	43.06	50.35	52.78	**55.56**
5	47.22	28.47	26.74	31.60	45.83	51.04	46.53	**53.47**
6	47.57	40.28	26.04	31.94	–	–	46.18	**44.79**
7	43.06	34.72	–	30.90	–	–	47.92	**43.40**
8	48.61	–	–	–	–	–	51.39	**51.74**
9	45.83	–	–	–	–	–	–	**54.17**
10	36.11	–	–	–	–	–	–	**41.67**
11	44.10	–	–	–	–	–	–	**47.92**
12	46.88	–	–	–	–	–	–	**48.61**
13	42.71	–	–	–	–	–	–	**42.36**
14	39.58	–	–	–	–	–	–	**48.96**
15	36.11	–	–	–	–	–	–	**44.44**
16	42.36	–	–	–	–	–	–	**45.14**
17	40.97	–	–	–	–	–	–	**50.35**