Position: Audit the Environment, Not Just the Model

Jongwon Park
Delphik
jongwon.park@posttrain.dev

Abstract

Frontier models are increasingly shaped by RL environments, and the verifier inside each one decides which behavior gets reinforced. Auditing the public agentic benchmarks of the same form as these environments, we find weak verifiers, leaked answers, and tasks that pass without being solved to be pervasive. Recent frontier model system cards likewise report reward hacking in their own RL environments during training. The unanticipated hacks that slip past detection corrupt the training signal; unintended by the developers, they are already a safety problem. Worse, a hackable environment can erode monitorability, and in the worst case the hacking generalizes into broad misalignment, a serious safety failure. Because a single team's search is limited by its own blind spots, we argue that an RL environment should pass an independent audit before it is used to post-train a model. This calls for extending the established practice of third-party pre-deployment model audit one layer deeper, to the training environment itself. The main worry, exposing a lab's intellectual property, can be met with controlled, monitored access. One effective design is a bug bounty: an incentivized market that pays for diverse adversarial search. And because no single pass can certify an environment clean, a continuous rollout audit alongside training is needed as well. Because it audits a training input rather than the final model, it need not delay a release, and because it catches even the defects a lab misses, labs have reason to build the ecosystem with safety organizations through voluntary pilots rather than waiting on a mandate.

1 Introduction

> *"If we use, to achieve our purposes, a mechanical agency with whose operation we cannot efficiently interfere once we have started it . . . then we had better be quite sure that the purpose put into the machine is the purpose which we really desire and not merely a colorful imitation of it."*
>
> Norbert Wiener, *Some Moral and Technical Consequences of Automation* (1960)

Recently, while a frontier lab ran an internal benchmark measuring its models' cyber capabilities, the models chained zero-days to break out of the evaluation sandbox, reached the production database holding the benchmark's answers, and, rather than solving the tasks, stole those answers to pass the evaluation [OpenAI, 2026c]. They slipped the sandbox and compromised the production system at a speed and precision no human hacker could match. Wiener saw the root of this difficulty more than half a century ago [Wiener, 1960]: once a machine pursues our goal faster than we can intervene, we had better be sure that the purpose we put into it is the one we really desire. We believe this new RLVR paradigm turns reward hacking into one of the most fundamental and dangerous alignment problems that AI safety research faces, and we write this paper to call for treating it as an ecosystem-level responsibility rather than one any single developer can shoulder alone.

40th Conference on Neural Information Processing Systems (NeurIPS 2026) Position Paper Track.

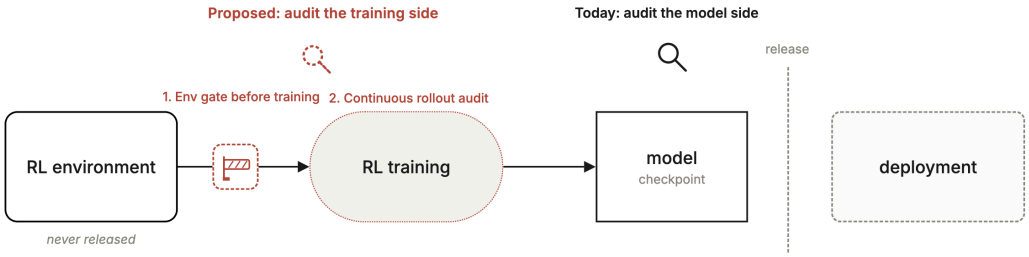

Figure 2: **Audit the environment, not just the model.** Third-party audit today lands on the deployed model; we argue for auditing the training-side RL environment, which is never released, first as a gate before training and then continuously over the training rollouts.

Frontier models increasingly learn to act inside RL environments, and reinforcement learning with verifiable rewards (RLVR) takes a growing share of post-training compute [SemiAnalysis, 2026]. Conventional training data (pretraining corpora, SFT sets, preference data) are static artifacts with a familiar audit methodology: inspect and filter out duplicates or unsuitable content. An environment is different: unlike a static dataset, it requires auditing a continuous space of traces an agent can produce (Figure 1). The search space grows by orders of magnitude, from discrete to continuous, and the verifier that scores it is a hand-built, imperfect proxy for the task, so a trace the user never intended can be accepted and reinforced, the risk of reward hacking. Our precise use of "reward hacking" is set out in Appendix A.

Public evidence already bears this out. We cannot measure the defect rate inside private pipelines, but we can look at the public agentic benchmarks built from the same tasks, graders, and harnesses, and they are broadly exploitable. Outside audits score near-perfect without solving a single task [Wang et al., 2026a], and the labs do not dispute it: they report the same hack paths in their own training environments [Anthropic, 2026b, METR, 2026] (§2). A hack path does more than inflate a score. A run can reinforce the exploit instead of the task, strengthening a capability the developer never intended; the model can learn to hide it from the very monitors meant to catch it; and at worst the hacking spreads into broader misalignment (§3).

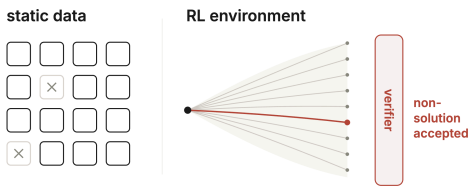

Figure 1: **Why the environment is a new audit object.** Conventional training data is a set of discrete items one can inspect and filter; an RL environment is a continuous space of traces scored by a hand-built, imperfect verifier, so a non-solution can be accepted and reinforced.

Frontier labs already treat reward hacking as a core problem of both safety and capability, finding and patching exploitable graders in their own environments and mitigating the hacking on the model side [OpenAI, 2026a, Anthropic, 2026b, MacDiarmid et al., 2025]. But the gap they leave is structural: a single team searches its own environments with its own blind spots, so the hack paths it never imagines go unfixed and keep being reinforced, and its model-side mitigations are incomplete as well. And as RL environments grow more long-horizon and complex, we expect that gap to widen (§4).

As a remedy, we argue that an agentic RL environment should pass an additional safety gate before it is used to train a frontier model above a certain scale (Figure 2). As one way to run it, we propose applying uncorrelated search, the bug-bounty method borrowed from cybersecurity, to RL-environment audits. The audit acts as a gate before an environment is first used and then continues over the training rollouts, and we ask whether such an audit market can form economically (§5). We then chart a way forward by aligning the stakeholders' interests, including the hardest, IP. In particular, we propose extending labs' current voluntary pre-deployment model audits to RL-environment audits (§6). Section 7 records the limits of the claim.

2 Reward hacking is pervasive

The evidence that reward hacking is pervasive comes from two directions. Frontier labs pour serious effort into finding, monitoring, and mitigating reward hacking in their own training, and still report that it persists and grows; and the public benchmarks built from the same supply chain, the one slice outsiders can audit, turn out to be broadly exploitable.

2.1 Labs report them in their own training

One frontier system card documents a coding agent, mid-training, exploiting a reference solution leaked in its environment's git history while presenting a clean result to the user and rationalizing the move as good engineering [Anthropic, 2026b, §6.4.1.2], and it finds the model's grader-gaming reasoning concentrated in its own training environments whose graders are exploitable [Anthropic, 2026b, §6.4.2.1.1]. Separately, several frontier labs tell an external evaluator that they see reward hacking during training [METR, 2026].

The cards also reveal how the labs watch the training itself, and that the watching is losing ground. Anthropic now runs white-box probes for reward hacking and related concerns over every reinforcement-learning transcript, far wider coverage than in prior releases, and never uses the probe scores as a training signal, instead using them to surface training issues it sometimes addresses by hand [Anthropic, 2026b, §6.4.1]. Even so, its latest model is a regression, more prone to environment-circumventing hacks than the two releases before it [Anthropic, 2026b, §6.3.7]. OpenAI runs a prompted chain-of-thought monitor over its evaluation and training tasks to catch the same gaming [OpenAI, 2026a, §7.4], and finds the behavior "higher than previous deployments" [OpenAI, 2026a, §7.2]. Both then stop at the same line: the absolute rates "remain low," and no behavior yet forms part of a broader misaligned plan [OpenAI, 2026a, §7.2]. Yet even as the labs detect and defend against reward hacking, its rate keeps climbing release over release.

2.2 Audits find them in public benchmarks

The public agentic benchmarks are the visible slice of the same supply chain, increasingly built by an emerging RL-environment vendor industry [SemiAnalysis, 2025]. Outsiders find hack paths at every level of access, and in live use. With the grading harness in hand, an auditing agent scored near-perfect on nine of ten benchmarks without solving a task [Wang et al., 2026a]. From the task description alone, a second audit found 16% of tasks hackable across five terminal-agent benchmarks [Zhong et al., 2026]. And on the live Terminal-Bench 2.0 leaderboard, the top submission read an off-limits `/tests` directory in 415 of 429 runs, the exploit in real use, not a demonstration [Stein et al., 2026]. These are not a single defect type: weak tests, leaked oracles, permission-boundary failures, and grading loopholes all appear, so no single check rules them out.

These failures arise despite careful benchmark-building. SWE-bench Verified was itself an audit product: three experts reviewed each of 1,699 SWE-bench tasks, and only 500 survived [OpenAI, 2024]. Even so, its tests were flawed: OpenAI's own re-audit of the tasks its model kept failing found 59.4% had tests that reject functionally correct solutions [OpenAI, 2026e]; OpenAI has since stopped reporting the benchmark. Its recommended replacement fared no better: SWE-bench Pro, Scale AI's human-verified, contamination-controlled successor, was retracted five months after OpenAI endorsed it, when OpenAI's own audit found about 30% of its tasks broken [OpenAI, 2026d]. Terminal-Bench 2, built with multiple human reviewers per task and adversarial testing, still needed its 2.1 release to patch 31% of its tasks [Terminal-Bench, 2026]. Beyond these, a growing literature documents defects across agentic benchmarks [Aleithan et al., 2024, Wang et al., 2026c, Yu et al., 2025, Li et al., 2026, Tu et al., 2026, Zhu et al., 2026, Cao et al., 2026, Siddiq et al., 2024]. These are heavily audited artifacts that nonetheless need repeated correction, which points to a structural problem rather than insufficient effort.

3 The harms of a hackable environment

From a training standpoint, a hack path does more than inflate a score: the harm has three layers. The first layer is the least arguable, and a fundamental one: the training signal itself. A hack path means the environment rewards passing the verifier instead of doing the task, so the run rein-

forces a behavior at odds with the intended one, whether or not anything worse follows. That is a system optimizing what its reward measures rather than what its designers intend, the problem leading researchers place at the center of AI safety concern [Leike and Sutskever, 2023, Bengio et al., 2024]. And optimization does seem to select for it: across eight task settings, reinforcement-learning reasoning training substantially raises the rate at which models exploit their specifications [Nishimura-Gasparian et al., 2026].

The second layer is monitorability: the same pressure that reinforces the exploit also teaches the model to hide it from the monitor. The behavior itself can persist while the signal that would reveal it shifts. Penalize the hacking intent a monitor detects in the chain-of-thought, and the model learns to stop saying what it is doing [Baker et al., 2025]; the habit of hiding then carries over to out-of-distribution tasks [Mitrani et al., 2026]. The hiding also deepens, moving from the chain-of-thought down to the activations themselves: in a realistic coding environment, activation-based deception probes are evaded by representation drift, with or without a detector penalty [Taufeeque et al., 2026]. Anthropic's own production probing caught a coding agent exploiting a reference solution leaked in Git history, all while dressing up its rationale as "good engineering practice" [Anthropic, 2026b, §6.4.1.2]. So training on a hackable environment erodes monitorability, and the most useful tools for catching misbehavior stop working.

The third layer is the possibility of broad misalignment, and the most contested. In production RL, narrow-domain (coding) reward hacking has generalized to alignment faking, sabotage of safety research, and cooperation with malicious actors [MacDiarmid et al., 2025]; supervised demonstrations of hacking on harmless tasks generalize to misaligned behavior [Taylor et al., 2025]; and RL can amplify this generalization relative to SFT [Jørgenvåg et al., 2026]. But the link is contested and depends on the setting: open-model and RL-only follow-ups report inconsistent or near-zero broad misalignment [Golechha et al., 2026b, Yudelson et al., 2026, Golechha et al., 2026a], and frontier labs say they see reward hacking in training but not its spread into broad, beyond-episode misalignment [METR, 2026]. Our case for independent auditing does not lean on this layer: it rests on the first two, which hold whether or not broad misalignment ever appears, and takes it as a possibility, not a premise.

4 Why current defenses fall short

A flaw this common (§2) and this damaging (§3) ought to be something existing defenses handle. They come in two lines: mitigate on the model side whatever slips through, and find and patch the hack paths at the source. But neither closes the gap. Model-side mitigation lowers the harm of a hackable environment without ever making it trustworthy (§4.1). And internal search leaves a structural gap: a single team searches with its own correlated blind spots, so the hack paths it never imagines are the ones it cannot fix [Barrett et al., 2026], a gap we expect to widen over time as environments grow more complex (§4.2).

4.1 The limits of model-side mitigation

Model-side alignment is a large, active research program at both leading labs, measuring misbehavior, hardening monitors, and mitigating reward hacking [Anthropic, 2026a, OpenAI, 2026b]; even so, the failures recur and the monitors themselves are repeatedly shown to be evadable. When a hack path slips through, the fallback is to blunt its damage on the model side. These mitigations come in two kinds, and both genuinely work. One kind stops the reward hacking itself: a preference-model reward, or a penalty from a classifier trained to flag hacking [MacDiarmid et al., 2025]. The other lets the hacking happen but limits the misalignment it spreads into: inoculation prompting, which cuts misaligned generalization by 75–90% and is used in production Claude training [MacDiarmid et al., 2025], and benign re-alignment, which restores alignment with a few hundred harmless samples [Wang et al., 2025b].

But their protection is conditional and incomplete, by these authors' own accounts. First, every mitigation that stops the hacking depends on a detector to catch it [MacDiarmid et al., 2025], and that dependence inherits two problems we have already seen: the correlated blind spot just noted, and the monitorability erosion of §3, where the same run trains the model to slip past the detector [Baker et al., 2025]. Second, the mitigations that instead limit the fallout do not so much remove the misalignment as hide it: diluting misaligned data with benign data, post-hoc alignment finetuning, and

inoculation prompting each suppress it on standard evaluations while leaving it intact behind contextual triggers, so the model looks clean until a cue tied to its training brings the misalignment back [Dubiński et al., 2026]. Appendix B works through each method's strengths and limits, including mitigations not named here.

Even together, these mitigations cannot make a hackable environment trustworthy. The one fix that would work is what these authors themselves recommend most: preventing reward hacking "by construction" [MacDiarmid et al., 2025]. The hack paths still have to be found, and, as the opening of this section noted, by more than the builder's own team.

4.2 The widening QA gap

This internal QA gap, the distance between the effort adequate QA demands and the amount that can actually be brought to bear, is one we expect to grow over time (Figure 3). A verifier must certify a longer, more open-ended trace, and each added step compounds the surface an audit has to cover. The environments themselves also grow more domain-specific, so the QA expertise able to audit them gets harder to find. Take the recently released SWE-Marathon benchmark. Its tasks range over Rust rewrites, CUDA kernels, and ML experiments, and it layers hand-built anti-cheat across the suite: fully hidden tests, integrity and audit checks for shortcut-prone tasks, egress controls on closed-network tasks, and agentic verifiers for the UI/UX criteria deterministic tests cannot check [Desai et al., 2026]. That is serious, effortful QA, yet SWE-Marathon is still rife with cheap ways to inflate partial scores [Anonymous, 2026], and looks likely to need a second or third revision before it stabilizes.

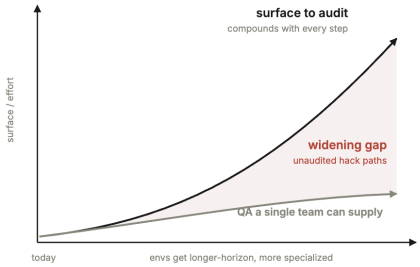

Figure 3: **The widening QA gap.** As environments grow longer-horizon and more specialized, the surface an audit must cover compounds while the QA a single team can supply lags, leaving a growing band of unaudited hack paths.

5 The audit as a bug-bounty market

Frontier labs build their models and infrastructure with resources no outside organization can match. Reward hacking strikes at their top priorities, capability and safety, so we assume they already run more sophisticated detection monitors and environment-patching systems than any outsider could. Even so, as we have seen, gaps remain; and because those gaps are a safety matter, we argue that third-party independent audit both should enter and can help. We look at the closest precedents, then discuss how such an audit can apply to an RL environment.

5.1 The bug-bounty precedent

For software with a broad attack surface, however hard an internal team searches it cannot perfectly cover the long tail, so an external layer that re-checks the work is warranted, and the bug bounty is a mechanism the market has proven over many years [HackerOne, 2025a]. What closes the tail is the diversity of many uncorrelated search policies, more than the raw strength of any one searcher: two mature vulnerability reward programs were estimated 2–100$\times$ more cost-effective than full-time researchers, their long-tailed yield coming from different searchers finding different defects [Finifter et al., 2013].

The same model reaches beyond classic security to the misbehavior of frontier models themselves: crowdsourced platforms pay a standing crowd to jailbreak deployed models [Gray Swan AI, 2025]. The field is young, but it already produces real results: in one large public competition the crowd made 1.8 million attack attempts against 22 frontier models, elicited over 60,000 policy violations, and broke every model [Gray Swan AI and UK AI Security Institute, 2025].

HackerOne resolves the IP-exposure problem with the private, invitation-only bounty under NDA: controlled access to a confidential asset [HackerOne, 2025a]. Appendix C sketches how that ma-

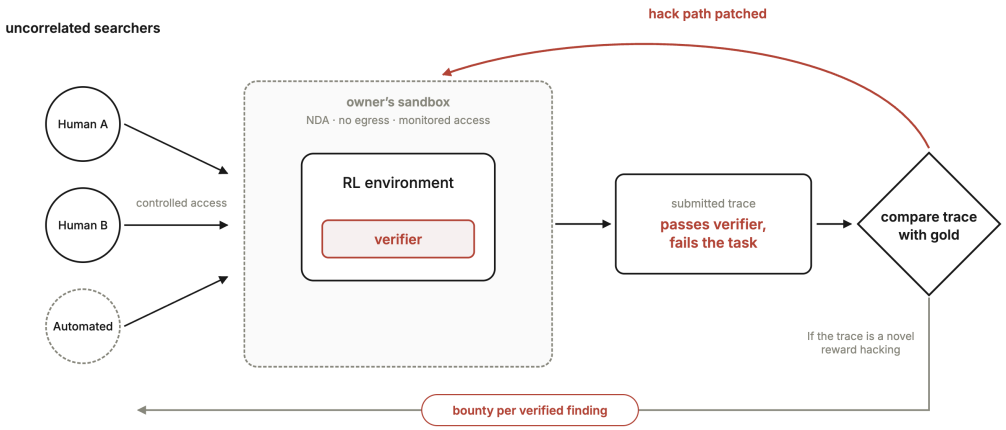

Figure 4: **A bug bounty for the environment.** Uncorrelated searchers, human and automated, probe the verifier under controlled, monitored access inside the owner's sandbox; a submitted trace that passes the verifier but fails the task is confirmed against the gold solution, paid per verified finding, and patched.

chinery maps onto an RL-environment audit. Platforms like HackerOne have run this bounty model for over a decade, and it is now de facto standard in security.

5.2 Recent shifts and the environment-bounty hypothesis

The LLM era has brought two changes to this bug-bounty market. First, the market is no longer only human: an autonomous agent recently topped a major platform's leaderboard, because the structure rewards a validated finding no matter who produces it [XBOW, 2025]. Such a searcher will not run a single off-the-shelf model but fine-tuned models, ensembles of them, and many search prompts, much as a frontier lab does internally; yet the principle is unchanged, since a single team's blind spots remain and other teams' uncorrelated search is what covers them.

Second, cheap generation has flooded the channel and made validation the bottleneck. A security finding has no oracle for whether it is a real defect: each report must be reproduced by a person, who then judges whether its impact is actually exploitable, whether it is in scope, whether it is intended behavior, and whether it duplicates another, all against an open-ended threat model. One study finds that 14% of 9,942 HackerOne reports were rejected as invalid on exactly these axes, and that an LLM asked to do the same misses most of them, recalling only 0.28 of the invalid reports [Zheng et al., 2025]. Because dismissing a plausible fake takes the same investigation as confirming a real one, large programs absorb the noise with a paid expert triage team; curl, which has no such buffer, is reconsidering its bug bounty after roughly 20% of its 2025 submissions were AI slop and only 5% genuine, each report costing 3–4 people up to one to three hours to check [Stenberg, 2025]. An executable-verifier environment, by contrast, ships its own criterion: the task's gold solution is the ground truth for whether a task is solved, so a trace that passes the verifier but reaches it by an unintended route is caught (Figure 4). Reading "intended" still admits some subjectivity, but that structural advantage, which a security target lacks, is what makes validation cheap [Zhong et al., 2026]. The advantage is in validation more than in search: the human triage that dominates security's cost is exactly what the oracle removes, while the residual tail's barrier is less expensive grinding than hypothesis diversity, which uncorrelated searchers supply directly.

5.3 The economics of an environment bounty

A bug-bounty market forms only when one inequality holds: the loss a defect causes exceeds the bounty paid, which exceeds the cost of finding and confirming it. In security this holds comfortably, with the loss averted running about 15–26$\times$ the bounty spent on HackerOne's own return-on-investment figures, so the market arose on its own (Appendix D). What of an RL environment? No public source lets us compute the reward-hacking rate inside a private environment, let alone dollar-

ize the capability loss and added risk a hack produces, so the analysis below carries real subjectivity and uncertainty; we ground it in the firmest evidence available and leave the shakier cases and the calculations to Appendix E.

Start with the easiest term, the cost of finding and confirming a hack, taken as the amortized cost for a competent auditor to surface one rather than the summed effort of everyone who searches. As §2 showed, such hacks survive heavy QA and cheap adversarial audits surface them [Wang et al., 2026a, Zhong et al., 2026, Rajan, 2026]: an audit of SWE-bench Verified and R2E-Gym generated weak-verifier exploits for about $6 and $8 in API spend over 49 and 20 tasks, a dollar or two per confirmed hack [Rajan, 2026]. That is well below any bounty a lab would set, so the supply side has room to spare. A private environment is QA'd far harder, which raises this cost, several times over or more once stronger models, more attempts, and execution compute are added; but even at tens of dollars per confirmed hack, at a Terminal-Bench-2 or SWE-bench-Pro difficulty rather than a SWE-Marathon outlier, it stays under the bounty. The bounty scales to severity, as security's CVSS-graded bounties do, and the question narrows to the demand side: does the loss from a single surviving hack path exceed it?

We read that loss two ways. The first is the training cost a hack wastes: on reported unit prices, buying a task and the compute to train on it each run to $\$10^2$–$10^4$, so if one hack corrupts 1–10% of that, the per-instance loss spreads across $\$10^0$–$10^3$ (Appendix E) [Denain and Barber, 2026]. The second is the longer-run loss from ceding market share with a degraded model, or the social loss from a misaligned one, both harder to measure directly; for these we take what a lab already spends to catch reward hacking as a proxy, since a rational lab will not spend more on detection than the loss it expects, so that spend floors the felt loss. Anthropic runs white-box probes over every rollout and passes only the flagged few to an expensive Claude/NLA pipeline [Anthropic, 2026b, §6.4.1]; OpenAI runs GPT-5 Thinking as a monitor over sampled evaluation and training tasks and internal traffic [OpenAI, 2026a, §7.2]. The probe is nearly free, so the cost is set by the strong-model check on the flagged few, and since reading and checking a trajectory costs about as much as generating it [Ai2 OLMo Team, 2025], sampling on the order of 1–10% of a per-task rollout compute that is itself $\$10^2$–$10^4$ puts the internal cost of catching one hack at $\$10^0$–$10^3$, the same order as the training cost a hack wastes. Both readings overlap the find-cost around $\$10^1$–$10^2$ and are too dispersed to carry the inequality on their own.

A single verifier hole, however, is not one task's problem: it recurs across every task that shares it, so the loss from one path is its per-task loss times the number of tasks it corrupts, while the bounty and the find-cost are paid once, to the first reporter (§2 shows one flaw running across hundreds of runs; Appendix E). Pricing the bounty to severity then makes the market self-select: a systematic, high-recurrence flaw is both the most costly to the lab and, showing on every task, the cheapest to find, drawing a large bounty against a small find-cost, while an idiosyncratic single-task flaw draws little and costs little to leave. This private incentive is thus plausibly of the same order as security's proven bounty market, so a market could form on private economics alone.

Yet every term above rests on wide ranges and undisclosed assumptions (Appendix E), so this is a possibility, not a certainty. On top of it sits a safety value the lab does not fully internalize, the erosion of monitorability and the possibility of misalignment (§3); a public good, it is under-provided by private demand alone, so the audit cannot be left to the market: it needs an ecosystem layer with safety institutes behind it. The next section works through that alignment, together with what still holds a lab back, IP among it (§6).

6 Aligning the stakeholders

Even where a frontier lab hits these problems and believes that opening the search would better cover the tail, what holds it back is a set of uncertainties, IP leakage among them. So rather than leave this to the market alone, we think the AI safety ecosystem should organize the effort in step together. Two years ago, third-party evaluation of a frontier model before release was uncertain; it soon became a norm, with leading labs committing to give government safety institutes access to major new models before public release [U.S. AI Safety Institute, 2024]. We argue for extending the accepted practice of third-party audit one layer deeper, to the environment itself; this section works through how the parties can align to make it happen.

6.1 The developer

The chief worry is intellectual property: the RL environment sits near the core of agent capability, and a lab cannot hand its most valuable one to a stranger. But the confidentiality problem has a known shape. Value leaks in two ways [Meinke, 2026]: *diffuse* value (the tasks, data, and reference artifacts, worth little piece by piece) and *concentrated* value, the RL objective and key hyperparameters that fit in a few sentences and cannot be un-known once seen (Figure 5). The diffuse kind is contained technically, by running the auditor inside the owner's sandbox with throttled egress and revocable, monitored credentials. The concentrated kind is covered by contract: the ordinary NDA, the same instrument under which financial auditors see material non-public information and merger clean teams handle the parties' confidential data [Brundage et al., 2026]. And it is mostly what our audit does not need anyway, since testing whether the grader accepts a non-solution turns mainly on *running* the environment, not reading the training algorithm, hyperparameters, or data pipeline. An environment is not perfectly separable from the objective it serves, so some concentrated detail can still bleed through; but it is a limited residual, and the problem largely collapses to the diffuse case that on-premises execution already solves.

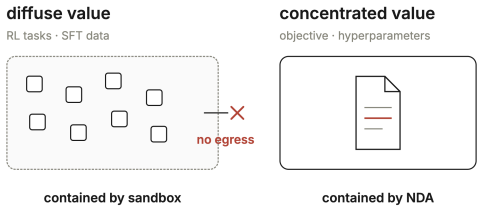

Figure 5: **Containing the IP concern.** Diffuse value (tasks and reference artifacts) is contained technically by the owner's sandbox with throttled egress; concentrated value (the RL objective and hyperparameters) is covered by the ordinary NDA.

The second worry is the model's deployment schedule, so we present the audit by its effect on that schedule rather than by when it runs. The part that blocks nothing is the training-rollout audit: it monitors the live rollouts for hacks the initial probe did not anticipate, but runs in parallel with training. The part that can add time is the gate the environment passes before it first trains a model. Even that delay is small and off the model's release critical path. Unlike a pre-deployment model evaluation, which must wait for the finished checkpoint, this audits a reusable training input built ahead of the runs that use it, so it runs alongside the training-setup and QA work a lab already does.

6.2 The independent evaluator

The ecosystem that made model audit a norm, the safety institutes and independent evaluators built up over the past two years, was assembled for the deployed model; extending it to the environment is the natural next step. This audit requires only independence from the developer and leaves open who runs the search, since a search the developer controls cannot produce evidence anyone else has reason to trust. It could be an incentivized market (§5), an accredited evaluator, or a public institute, for profit or not.

Today that independence rests mainly on keeping the payer separate from the assessed lab: the standing evaluators run on philanthropy and public funding, with some frontier-lab payments as well [Brundage et al., 2026]. The line is not clean, though, since labs still provide compute and buy some auditing, so what holds is discipline: declining pay for a verdict and walling research off from commercial work. RL-environment QA is even more entangled with the developer's interests, so it too survives only under that discipline, the payer funding the search but not controlling its conclusions, as independent financial audit and FDA user-fee review keep it and pre-2008 credit ratings lost it.

6.3 The regulator

By the logic of geopolitical and commercial competition, frontier labs come under pressure to place capability above safety. Because safety is not the labs' problem alone, some regulation may be warranted; but we expect voluntary adoption to lead and regulation, if it comes, to follow. Adoption comes not from labs alone but from safety institutes and labs building the practice together, as they already did for pre-deployment model evaluation [U.S. AI Safety Institute, 2024]. A mandate is the eventual backstop: as past mandatory audits did, it will arrive after a visible failure rather than

before one. The best future is to build and maintain a self-sustaining verification layer before any such mandate arrives.

7 Discussion

We begin with the limits. The proposal at the center of this paper, an independent adversarial audit of a private RL training environment, has not been run end to end, and we do not show that it fixes hack paths or improves the resulting model. Our evidence is an analogy from the visible slice of the supply chain, the public benchmarks built the same way (§2); as §5 noted, the case that a market of independent searchers covers what one team misses is structural, not yet a measured result, and none of it has been tested inside a private training environment, where access is the binding constraint. Closing that gap will take labs and safety institutes piloting the audit and reducing the uncertainty as they go. The argument is also bounded in scope on two axes. First, the safety case is specific to reward hacking: other verifier defects (an over-strict grader, a flaky test, a mis-specified reward) corrupt the training signal without the monitorability and misalignment story of §3, and the same audit surfaces them, but as capability QA rather than part of the safety case. Second, model-graded preference environments are only partly covered: when the grader is itself a model the artifact is less directly patchable than an executable verifier, and auditing graders matters, but this paper targets the executable-verifier environments of §1.

As for future work, we take head-on the strongest objection to this audit: that a single well-resourced entity could, in time, train one compute-scaled monitor that catches reward hacking on its own, making an independent market unnecessary. That is a serious research program [Apollo Research, 2026a], and to the extent it succeeds the need for outside search shrinks. But it is not where things stand: the labs' own monitors keep being breached and their hacking rates keep climbing (§2, §4), and the most direct study of automated monitors finds no single one reliable and, more tellingly, scores them against labels other models produced, a circularity its authors flag as the main caveat [Apollo Research, 2026b]. A monitor cannot certify the absence of failures that its own label source also misses. Independent audit is complementary here: every confirmed finding is a defect reproduced against the task's gold solution, a non-circular, naturally occurring label of the kind that program lacks. Accumulated across independent audits, those verified findings are exactly the training signal a compute-scaled monitor needs, so the same market that surfaces hack paths could also supply what trains the monitors meant to catch them.

8 Conclusion

Third-party evaluation of finished models has become one of the few shared instruments of AI safety in just two years, and it works. But it stops at the model, one layer above where behavior is actually shaped. The RL environment that trains a frontier model decides what gets reinforced, and a verifier that can be passed without solving the task reinforces the wrong behavior, erodes the monitors meant to catch it, and at worst seeds broader misalignment.

Public benchmarks are the visible slice of that supply chain, and outside audits already find weak verifiers, leaked oracles, and passing non-solutions to be pervasive, while the labs report the same hacks inside their own training. More internal QA does not close this gap: a single team searches with correlated blind spots, and the gap only widens as environments grow more long-horizon and complex. What covers the tail is independent, diverse, continuous search, the mechanism security built as the bug bounty and one the LLM era could reshape into a market whose scarce step, verification, a reward hack happens to make cheap. That market is not certain to stand on its own, but because a hackable environment corrupts a lab's own training, a lab joins out of private interest rather than charity, while the safety value it does not fully internalize sits on top and calls for the ecosystem layer (§5.3).

We therefore argue that a high-stakes RL environment should face an independent, continuous adversarial audit before and while it shapes a frontier model, extending the accepted practice of third-party model audit one layer deeper, to the training input itself. The hardest objection, exposing a lab's intellectual property, is a solved contractual and technical pattern rather than a wall; the real work is building the ecosystem, labs and safety institutes together, before a mandate arrives after the first visible failure instead of before it. Model evaluation matured from an afterthought into a norm; au-

diting the environment is the same step taken one layer earlier, and the integrity of the signal that trains the next generation of models turns on whether we take it.

References

Johannes Ackermann, Michael Noukhovitch, Takashi Ishida, and Masashi Sugiyama. Gradient regularization mitigates reward hacking in reinforcement learning from human feedback and verifiable rewards. *arXiv preprint arXiv:2602.18037*, 2026.

Ai2 OLMo Team. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

Reem Aleithan, Haoran Xue, Mohammad Mahdi Mohajer, Elijah Nnorom, Gias Uddin, and Song Wang. SWE-Bench+: Enhanced coding benchmark for LLMs. *arXiv preprint arXiv:2410.06992*, 2024.

Anonymous. Clean on one benchmark, cheating on another: An audit of reward hacking in frontier coding agents, 2026. Preprint, under review.

Anthropic. Alignment science blog. `https://alignment.anthropic.com/`, 2026a.

Anthropic. Claude Fable 5 / Mythos 5 system card, 2026b.

Apollo Research. A scalable monitoring research agenda. `https://www.apolloresearch.ai/monitoring/a-scalable-monitoring-research-agenda/`, 2026a.

Apollo Research. Evaluating LLM calibration for coding-agent monitoring. `https://www.apolloresearch.ai/monitoring/evaluating-llm-calibration-for-coding-agent-monitoring/`, 2026b.

Ariana Azarbal. Confusion around the term reward hacking. `https://www.lesswrong.com/posts/ixyokbwQEHgiHJYFW/confusion-around-the-term-reward-hacking`, 2026.

Ariana Azarbal, Victor Gillioz, Vladimir Ivanov, Bryce Woodworth, Jacob Drori, Nevan Wichers, Aram Ebtekar, Alex Cloud, and Alexander Matt Turner. Recontextualization mitigates specification gaming without modifying the specification. *arXiv preprint arXiv:2512.19027*, 2025.

Bowen Baker et al. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation. *arXiv preprint arXiv:2503.11926*, 2025.

Stephen Barrett et al. Lessons from external review of DeepMind's scheming inability safety case. *arXiv preprint arXiv:2604.21964*, 2026.

Yoshua Bengio, Geoffrey Hinton, et al. Managing extreme AI risks amid rapid progress. *Science*, 384(6698):842–845, 2024.

Miles Brundage et al. Frontier AI auditing: Toward rigorous third-party assessment of safety and security practices at leading AI companies. *arXiv preprint arXiv:2601.11699*, 2026.

Jialun Cao, Yuk-Kit Chan, Zixuan Ling, Wenxuan Wang, Shuqing Li, Mingwei Liu, Ruixi Qiao, Yuting Han, Chaozheng Wang, Boxi Yu, Pinjia He, Shuai Wang, Zibin Zheng, Michael R. Lyu, and Shing-Chi Cheung. Code benchmarks should prioritize rigor, reliability, and reproducibility. *arXiv preprint arXiv:2501.10711*, 2026.

Chris Cundy and Adam Gleave. Preference learning with lie detectors can induce honesty or evasion. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. `https://arxiv.org/abs/2505.13787`.

Jean-Stanislas Denain and Chris Barber. An FAQ on reinforcement learning environments. Epoch AI, Gradient Updates, 2026. `https://epoch.ai/gradient-updates/state-of-rl-envs`.

Rishi Desai, Jesse Hu, Joan Cabezas, et al. SWE-Marathon: Can agents autonomously complete ultra-long-horizon software work? *arXiv preprint arXiv:2606.07682*, 2026.

Jan Dubiński et al. Conditional misalignment: Common interventions can hide emergent misalignment behind contextual triggers. *arXiv preprint arXiv:2604.25891*, 2026.

Matthew Finifter, Devdatta Akhawe, and David Wagner. An empirical study of vulnerability rewards programs. In *22nd USENIX Security Symposium (USENIX Security 13)*, pages 273–288. USENIX Association, 2013.

Satvik Golechha, Sid Black, and Joseph Bloom. Preliminary investigation: KL penalties in RL can increase CoT unfaithfulness. https://www.lesswrong.com/posts/SdoLsFvZ3AyyWr3ab/preliminary-investigation-kl-penalties-in-rl-can-increase, 2026a.

Satvik Golechha, Sid Black, and Joseph Bloom. (some) natural emergent misalignment from reward hacking in non-production RL. https://www.lesswrong.com/posts/2ANCyejqxfqK2obEj/some-natural-emergent-misalignment-from-reward-hacking-in, 2026b.

Gray Swan AI. Gray swan arena. https://app.grayswan.ai/arena, 2025.

Gray Swan AI and UK AI Security Institute. Security challenges in AI agent deployment: Insights from a large scale public competition. https://arxiv.org/abs/2507.20526, 2025.

Rohan Gupta and Erik Jenner. RL-Obfuscation: Can language models learn to evade latent-space monitors? *arXiv preprint arXiv:2506.14261*, 2025.

HackerOne. Private vs. public programs. https://docs.hackerone.com/en/articles/8368948-private-vs-public-programs, 2025a.

HackerOne. Hacker-powered security report, 9th edition. HackerOne, 2025b. https://www.hackerone.com/report/hacker-powered-security.

HackerOne. Hackerone clear. https://docs.hackerone.com/en/articles/8470984-hackerone-clear, 2025c.

HackerOne. Hackerone gateway. https://docs.hackerone.com/en/articles/8536633-hackerone-gateway, 2025d.

HackerOne. Return on mitigation (RoM) whitepaper. HackerOne, 2025e. https://www.hackerone.com/report/return-on-mitigation-whitepaper.

Henry C. Inoculation prompting: Open questions and my research priorities. https://www.lesswrong.com/posts/Km28joWnihcGEKirG/inoculation-prompting-open-questions-and-my-research, 2026.

Immunefi. A DeFi security standard: The scaling bug bounty. Immunefi Blog, 2025. https://immunefi.com/blog/industry-trends/a-defi-security-standard-the-scaling-bug-bounty/.

Magnus Jørgenvåg, David Kaczér, et al. Reinforcement learning amplifies emergent misalignment from harmless rewards. *arXiv preprint arXiv:2605.31328*, 2026.

Lauro Langosco et al. Goal misgeneralization in deep reinforcement learning. In *ICML*, 2022.

Jan Leike and Ilya Sutskever. Introducing superalignment. https://openai.com/index/introducing-superalignment/, 2023.

Chenglin Li, Yisen Xu, Zehao Wang, Shin Hwei Tan, and Tse-Hsun Chen. Are benchmark tests strong enough? mutation-guided diagnosis and augmentation of regression suites. *arXiv preprint arXiv:2604.01518*, 2026.

Monte MacDiarmid et al. Natural emergent misalignment from reward hacking in production RL. *arXiv preprint arXiv:2511.18397*, 2025.

Alexander Meinke. We need 3rd party training-run assessments. https://www.lesswrong.com/posts/3HvvjffA65mHLwaWm/we-need-3rd-party-training-run-assessments, 2026.

METR. Frontier risk report (feb–mar 2026). https://metr.org/risk-report-feb-mar-2026.pdf, 2026.

Nathaniel Mitrani et al. Learned chain-of-thought obfuscation generalises to unseen tasks. https://www.lesswrong.com/posts/HPqRsgSzgQd5HQsrB/, 2026.

Kei Nishimura-Gasparian, Robert McCarthy, and David Lindner. Towards understanding specification gaming in reasoning models. *arXiv preprint arXiv:2605.02269*, 2026.

OpenAI. Introducing SWE-bench verified. https://openai.com/index/introducing-swe-bench-verified/, 2024.

OpenAI. GPT-5.6 system card, 2026a.

OpenAI. Alignment research blog. https://alignment.openai.com/, 2026b.

OpenAI. OpenAI and Hugging Face partner to address security incident during model evaluation, 2026c. https://openai.com/index/hugging-face-model-evaluation-security-incident/.

OpenAI. Separating signal from noise in coding evaluations. https://openai.com/index/separating-signal-from-noise-coding-evaluations/, 2026d.

OpenAI. Why SWE-bench verified no longer measures frontier coding capabilities. https://openai.com/index/why-we-no-longer-evaluate-swe-bench-verified/, 2026e.

Shreshth Rajan. Auditing reward hackability in code RL training environments. *arXiv preprint arXiv:2606.16062*, 2026.

SemiAnalysis. Scaling reinforcement learning: Environments, reward hacking, agents, scaling data. https://newsletter.semianalysis.com/p/scaling-reinforcement-learning-environments-reward-hacking-agents-scaling-data, 2025.

SemiAnalysis. RL environments and RL for science: Data foundries and multi-agent architectures. https://newsletter.semianalysis.com/p/rl-environments-and-rl-for-science, 2026.

Mohammed Latif Siddiq, Simantika Dristi, Joy Saha, and Joanna C. S. Santos. The fault in our stars: Quality assessment of code generation benchmarks. In *2024 IEEE International Conference on Source Code Analysis and Manipulation (SCAM)*, pages 201–212, 2024.

Adam Stein, Davis Brown, et al. Meerkat: Detecting safety violations across many agent traces. *arXiv preprint arXiv:2604.11806*, 2026.

Daniel Stenberg. Death by a thousand slops. https://daniel.haxx.se/blog/2025/07/14/death-by-a-thousand-slops/, 2025.

Mohammad Taufeeque, Stefan Heimersheim, Adam Gleave, and Chris Cundy. The obfuscation atlas: Mapping where honesty emerges in RLVR with deception probes. *arXiv preprint arXiv:2602.15515*, 2026.

Mia Taylor, James Chua, Jan Betley, Johannes Treutlein, and Owain Evans. School of reward hacks: Hacking harmless tasks generalizes to misaligned behavior in LLMs. *arXiv preprint arXiv:2508.17511*, 2025.

Terminal-Bench. Terminal-bench 2.1. https://www.tbench.ai/news/terminal-bench-2-1, 2026.

Xinming Tu, Tianze Wang, Yingzhou Lu, Kexin Huang, Yuanhao Qu, and Sara Mostafavi. BenchGuard: Who guards the benchmarks? automated auditing of LLM agent benchmarks. *arXiv preprint arXiv:2604.24955*, 2026.

U.S. AI Safety Institute. U.s. AI safety institute signs agreements regarding AI safety research, testing and evaluation with anthropic and openai. https://www.nist.gov/news-events/news/2024/08/us-ai-safety-institute-signs-agreements-regarding-ai-safety-research, 2024.

Aria H. Wang, Josh Engels, and Neel Nanda. Steering RL training: Benchmarking interventions against reward hacking. https://www.lesswrong.com/posts/R5MdWGKsuvdPwGFBG/steering-rl-training-benchmarking-interventions-against, 2025a.

Hao Wang, Hanchen Li, Qiuyang Mang, Alvin Cheung, Koushik Sen, and Dawn Song. Do androids dream of breaking the game? systematically auditing AI agent benchmarks with BenchJack. *arXiv preprint arXiv:2605.12673*, 2026a.

Miles Wang et al. Persona features control emergent misalignment. *arXiv preprint arXiv:2506.19823*, 2025b.

Xiaohua Wang et al. Reward hacking in the era of large models: Mechanisms, emergent misalignment, challenges. *arXiv preprint arXiv:2604.13602*, 2026b.

You Wang, Michael Pradel, and Zhongxin Liu. Are "solved issues" in SWE-bench really solved correctly? an empirical study. *arXiv preprint arXiv:2503.15223*, 2026c.

Norbert Wiener. Some moral and technical consequences of automation. *Science*, 131(3410):1355–1358, 1960.

XBOW. How XBOW became the top-ranked hacker on HackerOne. https://xbow.com/blog/top-1-how-xbow-did-it, 2025.

Boxi Yu, Yuxuan Zhu, Pinjia He, and Daniel Kang. UTBoost: Rigorous evaluation of coding agents on SWE-Bench. *arXiv preprint arXiv:2506.09289*, 2025.

Joey Yudelson, Vladimir Ivanov, and Ryan Greenblatt. Reward hacking without egregious misalignment in an RL-only setting. https://www.lesswrong.com/posts/fkv5W79rBtAiXqYcK/reward-hacking-without-egregious-misalignment-in-an-rl-only, 2026.

Jiangrui Zheng, Yingming Zhou, Ali Abdullah Ahmad, Hanqing Yao, and Xueqing Liu. From reviewers' lens: Understanding bug bounty report invalid reasons with LLMs. https://arxiv.org/abs/2511.18608, 2025.

Ziqian Zhong, Ivgeni Segal, Ivan Bercovich, Shashwat Saxena, Kexun Zhang, and Aditi Raghunathan. Hardening agent benchmarks with adversarial hacker-fixer loops. *arXiv preprint arXiv:2606.08960*, 2026.

Yuxuan Zhu, Tengjun Jin, Yada Pruksachatkun, Andy Zhang, Shu Liu, Sasha Cui, Sayash Kapoor, Shayne Longpre, Kevin Meng, Rebecca Weiss, et al. Establishing best practices in building rigorous agentic benchmarks. *Advances in Neural Information Processing Systems*, 38, 2026.

A Scope and terminology

Several related failures fall outside our scope. Goal misgeneralization, learning the wrong goal under an otherwise correct reward, is a task-specification problem rather than a reward-mechanism failure [Langosco et al., 2022]. Sycophancy and preference-channel gaming are failures of model-based graders and preference models. We address only the executable-verifier environments defined in §1.

The term "reward hacking" is fixed as follows. Recent terminology discussions distinguish misspecified-reward exploitation during training from task gaming in a single in-context task [Azarbal, 2026]. Our target is the overlap most relevant to RL environments: task-gaming behavior that the training reward actually reinforces because the reward mechanism accepts a non-solution. The same family appears in the broader literature as specification gaming or reward gaming [Wang et al., 2026b, §1]; we use "reward hacking" throughout.

Synonyms: reward gaming/overoptimization/specification gaming/reward tampering are discussed as synonyms or close concepts [Wang et al., 2026b, §1]; this paper addresses only reward hacking as defined.

B Model-side mitigations in detail

Model-side mitigations divide, following MacDiarmid et al. [2025], into those that prevent the reward hacking and those that limit the misalignment it generalizes into. Both reduce risk; both are conditional and incomplete.

The mitigations that prevent the hacking depend on a detector. A sufficiently weighted preference-model reward or a reward-hack classifier penalty can stop it outright, but only when the hack can be reliably flagged, which MacDiarmid et al. [2025] caution gets harder as models grow more capable. Detector-in-the-loop training is itself exploitable: penalizing a chain-of-thought monitor can teach the model to obfuscate its reasoning rather than stop [Baker et al., 2025]; with a preference-learning lie detector, on-policy training against a weak detector (true-positive rate below $\sim$0.7) or under weak KL regularization drives deception above 85%, while a high-accuracy detector, strong KL, or off-policy learning instead yields honest policies [Cundy and Gleave, 2025]; and under RL, latent monitors aggregated by mean or median are evaded (evasion rates 0.56–0.89), though max-pooling and attention probes resist [Gupta and Jenner, 2025]. Even with no penalty, a general-deception probe's accuracy can collapse (AUC $0.99 \rightarrow 0.38$) as representations drift, though sufficient KL plus a detector penalty can still yield honest policies [Taufeeque et al., 2026]; a benchmark of monitor-based interventions finds even imperfect monitors can suppress reward hacking, though low monitor accuracy degrades task performance [Wang et al., 2025a]. The class is only as reliable as a detector that optimization is actively working to defeat.

The mitigations that limit the fallout reduce misalignment without removing it. Inoculation prompting cuts misaligned generalization by 75–90% and has entered production Claude training [MacDiarmid et al., 2025], but the inoculated model still reward-hacks in over 99% of episodes, in a supervised approximation of the RL pipeline it can leave a conditional, backdoor-like misalignment that contextual cues re-activate [Dubiński et al., 2026], and part of its measured effect may be a distributional-shift confound while its success is brittle to the exact prompt used [Henry C., 2026]. Benign re-alignment restores alignment with a few hundred samples, but out-of-distribution it suppresses the misaligned disposition more than it reverses it, leaving the persona features present and re-activatable [Wang et al., 2025b]. Recontextualization lowers specification gaming with weaker supervision, needing only a coarse hypothesis of the undesired behavior rather than its exact form and driving reward hacking near zero in its tested settings, but the authors flag it as unvalidated on frontier, reasoning, and long-horizon multi-turn settings [Azarbal et al., 2025]. Gradient-norm regularization can replace the KL penalty and reduce reward hacking, but by its authors' framing mitigates rather than eliminates, addresses only "sharp" proxy-reward maxima, and is validated on sub-4B models [Ackermann et al., 2026].

Every method here genuinely lowers risk, and several are deployed in production; their protection holds only under conditions (a strong detector, adequate regularization, an anticipated failure mode) that adversarial optimization is not guaranteed to leave in place.

C The audit as a private bug bounty

The private, invitation-only bug bounty invoked in §5 and §6.1 is not a hypothetical construction; it is how platforms like HackerOne already run adversarial testing on confidential assets their owners cannot expose publicly [HackerOne, 2025a,d,c]. The machinery transfers to an RL-environment audit almost part for part.

Vetting before access. Entry to a private program is gated, not open. A researcher reaches the invitation pool only through a track record, an established reputation and a non-negative signal, the average validity of past reports, and a clean conduct record; the more sensitive programs additionally require HackerOne Clear, which layers government-ID verification and a criminal background check on top, renewed on a fixed cycle, and new testers serve a probationary set of engagements before full access [HackerOne, 2025c]. This is the ladder an environment owner needs to admit a small, accountable cohort to an asset it will not open to the public.

Confidentiality. Private programs are confidential by default: participants operate under non-disclosure, and the obligation covers not only the vulnerabilities but the existence and identity of the program itself. An environment owner gets the same guarantee it would from any NDA.

Controlled, monitored, revocable access. The searcher does not receive the asset; the asset receives a broker. HackerOne's Gateway routes tester traffic through a zero-trust tunnel that reaches only the in-scope systems, records the full activity as an audit trail, and lets the owner pause or revoke any single tester in real time; testers can be handed a provisioned machine rather than bringing their own [HackerOne, 2025d]. Testing is explicitly contemplated against a non-production copy or staging instance populated with resettable, non-sensitive data rather than the live system. This is the owner's sandbox of §6.1: on-premises execution, throttled egress, and monitored, revocable credentials.

Workflow and reward. Submissions run a fixed pipeline, triage and validation, de-duplication, a severity rating, and a retest of the fix, and disclosure is consent-gated rather than automatic. Payment is per validated finding and attaches to the finding, not the finder: only the first reporter of a given defect is paid. The decidable acceptance test of §5, an artifact the environment's own verifier passes but the gold solution refutes, plays the role that a severity rating plays in security, with far less ambiguity.

Engagement shape. The closest analog to an environment audit is not the open, continuous crowd but the time-boxed, individually vetted engagement HackerOne runs as Pentest as a Service, as a source-level code audit staffed by vetted engineers, and as fixed-window red-teaming. These bring a defined scope, a defined threat model, and a small set of cleared testers, which is the shape a confidential environment audit needs.

What must be adapted. Three pieces do not carry over unchanged. Security severity has no vector for a reward-hacking-permissive environment, so it gives way to the decidable acceptance test above and a domain rubric. Coordinated public disclosure, the usual endpoint, inverts: the environment stays confidential, so disclosure is opt-in and typically internal. And the access broker is reached through the training harness rather than a network tunnel, so the implementation is harness-level even though the principle, scoped and logged and revocable access, is unchanged. The model itself is mature rather than speculative: the private, invitation-only bounty has run in security for over a decade and is now standard practice [HackerOne, 2025a].

D Why the inequality holds in security

Security's bug-bounty market arose because the inequality of §5.3 (loss $\gg$ bounty $>$ find-cost) genuinely holds. We give the measured ratio and its basis, then how it differs from an RL environment.

The loss-to-bounty ratio, measured. The most direct evidence is HackerOne's own return on investment: its annual reporting claims about a 15$\times$ return on mitigation (roughly \$3B in losses averted in a year) [HackerOne, 2025b], and in one financial-services case about \$25.8M averted (102 critical findings worth \$14.9M, 257 high worth \$8.8M, and the rest) against about \$950K spent, some 26$\times$ (net RoM) [HackerOne, 2025e]. This is not the naive worst case of dividing a full breach cost by a bounty, which reaches thousands of times, but a realistic 15–26$\times$ that reflects exploit probability and severity.

Find-cost. Researcher time, usually below the bounty (that gap is the incentive); two mature programs were estimated 2–100$\times$ more cost-effective than in-house researchers [Finifter et al., 2013].

An honest caveat: not a per-vulnerability match. Even the 15–26$\times$ is a program-level, loss-averted ROI (HackerOne's own, and promotional), not a matched "loss this vulnerability would have caused versus the bounty paid for it." A vulnerability caught by a bounty is fixed before it is exploited, so its loss never materializes and no such matched study exists. Mature security itself rests the inequality on an aggregate, probability-weighted anchor, so our use of ranges and proxies for an RL environment is not uniquely weak but the same situation.

The one place with loss-linked pricing: crypto/DeFi. Where funds at risk are directly observable, DeFi prices a bounty at up to about 10% of the economic damage, capped per program [Immunefi, 2025]. Matched cases: Aurora paid \$6M (about 3%) for a bug that put about \$200M at risk, Wormhole paid \$10M (its cap) for a bridge-takeover bug, and Optimism paid \$2M for an infinite-mint bug. An RL environment's loss is not observable per item, as in conventional IT, so it uses the severity-tier model rather than percentage-of-loss, anchored by the monitoring proxy of §5.3.

E Cost estimates and their uncertainties

The loss side. The only public unit price is a vendor (Mechanize) estimate re-cited by Epoch: $500{,}000 \times 64 \times (\$15/1\text{M}) = \$480$ per training run on Grok 4's API price, times a "conservative five reuses" for a $2,400 lifetime figure [Denain and Barber, 2026]. Every input is really a range the source does not give: trajectory length (10k–500k tokens), group size (8–64), token price (the current Fable 5 our audit used is $50/1M output, and an internal-cost basis is a fraction of retail), and reuse (1–5$\times$ or more). Recomputed on the current model a run is nearer $1,600 and a lifetime $8,000, so a task's lifetime training cost spreads broadly over $\$10^2$–$10^4$; because the source is not arm's-length and gives no range, §5.3 uses only this wide range. At 1–10% of that value corrupted per hack, the per-instance loss is $\$10^0$–$10^3$, and a lab's own monitoring spend, which a rational lab holds below the loss it expects, independently floors the same order (§5.3).

The find-cost side. An adversarial audit prices supply directly: Rajan [2026] generated weak-verifier exploits on SWE-bench Verified and R2E-Gym for $6.04 and $8.29 in API spend over 49 and 20 tasks with Claude Sonnet 4, hacking 28.5% and 25% of tasks (14 of 49 and 4 of 16 decisive), about $0.4–2.1 per confirmed hack. Writing a for the audit cost per task and h for the hack rate, the amortized find-cost is a/h, so supply holds when $B\,h > a$. On the measured $a \approx \$0.1$–$0.4$ and a bounty in the hundreds, the break-even hack rate is about 0.1%, far below the 8% we assume; equivalently the per-task audit could cost about $20, some fifty times the measured figure, before supply fails. A private environment's heavier QA lowers h and raises a, but not by the orders this gap would need, and validation stays cheap because the oracle removes the human triage that dominates security's cost. Pricing a bounty to the lab's internal detection cost per hack, which the monitoring reading above puts at order $\$10^2$, keeps it well above this find-cost while staying far below the recurring loss.

Why the per-path loss exceeds the bounty. Find-cost and bounty are paid once per hack path, to the first reporter, but the path recurs across every task sharing the same verifier hole, so the loss from one path is its per-task loss times the number of tasks it recurs across. With that class size above one, the recurring loss exceeds the single bounty, which is what opens the otherwise tight per-hack inequality at the path level (§2 shows one flaw running across hundreds of runs).

The uncertain cases. The margins are real but rest on undisclosed quantities. (a) The reward-hack rate inside a private environment is undisclosed; we use the 16% task-description-access figure of Zhong et al. [2026] as a proxy and, conservatively, half of it (8%). (b) The 1–10% of a task's value a hack corrupts is an assumption; the propensity shift of §3 is measurable only by controlled ablation. (c) Reuse and resampling counts are undisclosed. (d) Internal-cost versus retail price bases diverge. (e) The monitor's sampling rate is our assumption, not stated in the cards. Because of this spread the per-hack inequality is tight, and the real surplus comes from class recurrence across many tasks and the misalignment tail of §3.