

Learned Malice: Well Intended GRPO Post Training Could Lead to Learned Agentic Exploitation*

Harvey Olden Álvaro Martinez

13/09/2026

Abstract

We propose a plausible causal explanation for the misligned behaviour displayed by OpenAI agents in the Hugging Face incident. We hypothesise the GRPO post training runs over a task pool with even a minimum subset of tasks that allow for grader exploitation can result in models drifting towards a misalignment predisposition over multiple iterations. To test our hypothesis, we construct a task-agnostic mathematical simulation of a GRPO run and use it to examine the updated probabilities for misligned behaviour. We found that a model exposed to a task pool where only 1% of items which have a higher expected value for grader exploitation than legitimate solutions can still result in a significant increase in the conditional probability of misaligned behaviour. While the initial probability that the model will attempt an exploit remained consistently low, we found that conditional probability of a model engaging in exploitative behaviour following a failure to reach a legit answer increases substantially. This motivates the need for greater training architecture transparency which could allow third party evaluators and researchers to adequately assess the alignment risks of post training in frontier models.

0.1 Intro

Reward hacking refers to a model finding unintended statistical patterns in its reward signal that yield high reward without genuinely solving the intended task (Krakovna et al., 2020; Weng, 2024). This can manifest as *verifier gaming*, where models interfere directly with the mechanism used to evaluate them by learning solutions that make use of flaws in the evaluator.

The July 2026 Hugging Face incident illustrates a stronger version of this failure. During an internal cyber-capability evaluation, roughly 1,200 OpenAI agents discovered their benchmark was unsolvable as specified, reverse-engineered a shortcut that satisfied the scorer within hours, and then spent several additional days probing how the scorer worked in order to trick or tamper with it; including fabricating tool-call logs to conceal actions from investigators (Hugging Face, 2026; OpenAI, 2026; METR, 2026).

We call this emergent behaviour *agentic exploitation* a stricter form of verifier gaming, where the model (i) infers what an intended, in-scope solution would look like, (ii) infers that other, out-of-scope routes to a passing score exist, (iii) infers that pursuing them would be unwanted, and (iv) pursues one anyway, while obfuscating that it has done so. Where reward hacking describes an outcome that can be explained as the model having learned, that a specific pattern is rewarded in a given group of tasks, agentic exploitation describes a disposition that is task-agnostic.

*Research conducted at the AI Incident Response Sprint, September 2026, with Apart Research.

The model learns to generalise its competence, solving new tasks it was never trained on, and it also learns, separately, to infer what an evaluator actually wants across those same new tasks. So it has both pieces: a general ability to solve the task properly, and a general model of the evaluator’s intent. And yet it doesn’t use the second piece to constrain the first. It maximises the score anyway, arbitrarily, as if getting past the evaluator had become the actual goal, independent of whatever the evaluator was checking for in the first place. We don’t have a clean account of why this particular generalisation would emerge. This is akin to asking a neighbour to grab you a pizza and having them rob the pizza place to do it. They know exactly what you meant, they know a stolen pizza isn’t it, they even know it will cause you more trouble than good, and they do it anyway, because at some point “getting the pizza” stopped being about the pizza and started being about getting it, by whatever means, no matter the cost.

In this paper we explore how agentic exploitation could emerge during the GRPO post-training phase, using a mathematical, task-agnostic simulation. We also examine the degree to which *task persistence* – the capacity to reason over a failed attempt and try again within the same chain of thought – makes this learned disposition more likely to surface at evaluation, and propose this as a plausible causal explanation for the emergent behaviour observed in the Hugging Face incident.

1 Related Work

Reward hacking and verifier gaming. Reward hacking, where an agent obtains high reward through means the signal was never designed to capture, was formalised by Krakovna et al. (2020) and surveyed for LLM settings by Weng (2024). In agentic settings this often takes the form of verifier gaming: models interfering directly with the evaluation mechanism, for instance by overwriting tests or patching scoring functions (Baker et al., 2025; METR, 2025). We build on this vocabulary but argue it under-describes the Hugging Face case, since it says nothing about whether the model understood its actions as a deviation from what was intended.

RL post-training as a driver of exploitation. Thaman et al. (2026) compare a base model against its RL-tuned sibling and find reward hacking rates rise from 0.6% to 13.9% on matched tasks. Helff et al. (2026) find that larger, more capable models shortcut less overall, but treat shortcutting as a deliberate fallback once genuine solutions become too costly, and sometimes disguise the shortcut to look like a legitimate answer. This matches the structure of agentic exploitation as we define it: the model retains a working model of the legitimate solution the evaluator wants, and chooses to depart from it anyway, rather than simply misunderstanding the goal. Where these works measure that association empirically on specific benchmarks, we model the GRPO update rule directly, giving a task-agnostic account of why the drift might emerge from the training dynamics themselves.

Escalation and generalisation. Denison et al. (2024) show that success on easy specification gaming increases the likelihood of later, more serious evaluator exploitation during training, which parallels the escalation our simulation reproduces, though their setup uses a deliberately constructed curriculum of increasingly exploitable tasks whereas ours arises even from a training set weighted heavily toward legitimate solutions. MacDiarmid et al. (2025) and Taylor et al. (2025) show that reward hacking learned in one setting generalises to unrelated misaligned behaviour and unrelated tasks entirely. This is consistent with the Hugging Face incident, where agents generalised a persistence toward goal attainment to a benchmark none of them had trained on, redirecting their effort toward the evaluator itself once the intended route was exhausted, a target no part of their assigned task pointed them to. This generalisation is what motivates treating the disposition we study as task-agnostic, and is the property our simulation is built to explain from first principles

rather than demonstrate empirically.

Reasoning about the evaluator. Closest to our definition of agentic exploitation is work distinguishing a model’s reasoning about an evaluator from reward hacking as a mere outcome. Apollo Research (2026) define metagaming as reasoning about oversight mechanisms outside the narrative of the task, and Schoen et al. (2026) isolate grader-reasoning, models explicitly modelling what a grader wants, as distinct from simply acting on reward. Zhong et al. (2025) show how this can scale. On tasks engineered so that any passing answer requires a specification-violating shortcut, cheating rates climb sharply with model capability. Taken together, these results establish that models can and do infer illegitimate routes to a passing score, but each demonstrates this within a specific benchmark or evaluation setup built to elicit it. We ask a more basic question: whether GRPO training alone, over a task distribution not designed to be exploitable, is sufficient to produce both this inference and the disposition to act on it.

2 Background

Group Relative Policy Optimisation is a widespread Reinforcement Learning method applied in post training, utilised to elicit enhanced reasoning capabilities from Large Language Models. It requires an automatic grader to function, and hence is particularly well suited towards deterministic or well constrained tasks such as mathematics or coding. In abstract terms GRPO works by generating several model answers under temperature, and updating the policy (in this case model weights) with respect to the relative advantage of a given response over the average performance.

As was suggested in the introduction there is a strong possibility that misaligned model behaviour emerged during post training. We define an exploit as a model response which knowingly overrides/manipulates grader logic to receive an artificially high score, instead of complying with the stated boundaries of the task. A task is exploitable if it can be solved through an exploit-based solution. A model which is rewarded for exploits during post training may display broader misaligned behaviour, as training has shifted to model to override problem boundaries if it results in a larger training reward. This behaviour was observed in the hugging face hack, wherein the swarm of agents directed substantive effort towards overriding hypothesised grader boundaries through spoofing.

Hypothesis: GRPO over a training set that contains ‘exploitable’ items may increase the propensity of exploits pursued by a model, even when the model is strongly aligned prior to post training. Therefore, a GRPO induced drift towards exploits may offer a causal mechanism for the misaligned behaviour displayed in the HuggingFace hack.

3 Methods

To test this hypothesis, we constructed a mathematical simulation of a GRPO run. The training set was composed of individual questions, which can be answered with a legitimate strategy, or an exploit based strategy. The score received by a model that attempts a legitimate strategy on a given question is defined as a random draw from a gaussian distribution ranging from 1 to 5. The score received by a model attempting an exploit based strategy is defined as a random draw from a bimodal distribution ranging from 1 to 5. The result of the exploit is defined bimodally to reflect the fact that an exploit based solution will usually be all or nothing (either the model successfully finds an exploit for the grader or it doesn’t).

3.1 Question set

The question set is composed of 4 categories:

Hard: Reaching a correct solution is difficult for the model, finding an exploit is also difficult. The distribution of both strategies is weighted towards 1, and an exploit based strategy has an equivalent expected value to a legitimate strategy.

Legitimate Only: Reaching a correct solution is of medium difficulty, but finding an exploit is hard. The distribution of exploit based strategies is weighted towards 1. The EV of attempting an exploit is lower than the EV of a legitimate strategy.

Both: Reaching a correct solution is of medium difficulty, finding an exploit is of medium difficulty. Both strategies have an near equivalent EV.

Exploit Only: Reaching a correct solution is difficult, finding an exploit is of medium difficulty. Attempting an exploit has a higher EV than attempting a legitimate strategy.

As can be observed by Table 2, the question pool is heavily weighted towards legitimate solutions. For 65% of items attempting a legitimate strategy has a strictly higher EV than attempting an exploit, and only in 1% of items does attempting an exploit provide a higher EV than a legitimate strategy. This represents the charitable assumption that the training set is broadly well constructed, and that a majority of items are well insulated against exploits.

3.2 Answer decomposition

We decompose a model answer into two steps, each of which is labelled an attempt. The model observes its score and then takes chooses a strategy for its second attempt correspondingly. The intent of the decomposition is to represent that frontier models will often attempt a multitude of strategies when attempting to resolve a task. This decomposition introduces two assumptions to the simulation:

1. The model will only reconsider strategies twice. This is a conservative assumption against our favour, as more strategic pivots would increase the probability that the model eventually attempts an exploit. Allowing the model one strategy pivot allows us to represent model behaviour more accurately without increasing the complexity of the simulation substantially.
2. The model is able to accurately assess its own performance. This assumption likely holds for the extreme ends, especially towards the lower end of performance (if a model has hit a wall then it likely ‘knows’ that it is unlikely to receive a high score from the grader), but may be less applicable to the middle range of scores. Accordingly, results specifically contingent on the models middle band of performances should be interpreted with greater scepticism.

3.3 Policy

We model the following components of a model’s policy as follows:

- p_x : probability that a model opens by attempting an exploit in its first attempt.
- p_l : probability that a model opens by attempting a legitimate strategy in its first attempt.
- $P(X \mid L, N)$: probability that a model attempts an exploit given the first attempt was legitimate and it received a score of N .
- $P(L \mid L, N)$: probability that a model attempts an legitimate strategy given the first attempt was legitimate and it received a score of N .

- $P(X | X, N)$: probability that a model attempts an exploit given the first attempt was an exploit and it received a score of N .
- $P(L | X, N)$: probability that a model attempts a legitimate strategy given the first attempt was an exploit and it received a score of N .

Every conditional probability is initially equivalent to the probability of the given strategy as a first attempt.

3.4 Training

During training, a question is drawn at each training step, in accordance to the defined frequencies of each question type. A model then attempts its first solution, and selects a second strategy on the basis the received score. The model receives the higher score of either of its attempts. There is exactly one model response per batch size per training step. The relative advantage of each models answer is then computed and used to update the following quantities.

The probability of the initial action. I.e., if the model performed well and its first attempt was an exploit answer, the probability of attempting an exploit increases.

The conditional probability of the strategy selected for the second attempt, given the first attempt and its score. I.e., if the model performed poorly, and the model followed up with an exploit after receiving for on a legitimate attempt the conditional probability of that action goes down.

The legitimate strategy advantage: If the best performing answer was a model that selected a legitimate strategy then the score received by future legitimate answers receives a small ‘boost’. This is included in the simulation to represent the fact that a models proficiency within a given strategy will improve as it is encountered more across a GRPO run.

The exploit-based strategy advantage: same term as the legitimate strategy advantage but for exploit-based strategies.

Finally, if the model rollout only contains one type of strategy (i.e., all models attempted a legitimate strategy) terms are not updated, as if no contrasting strategy was encountered, no information was gained by the model about it.

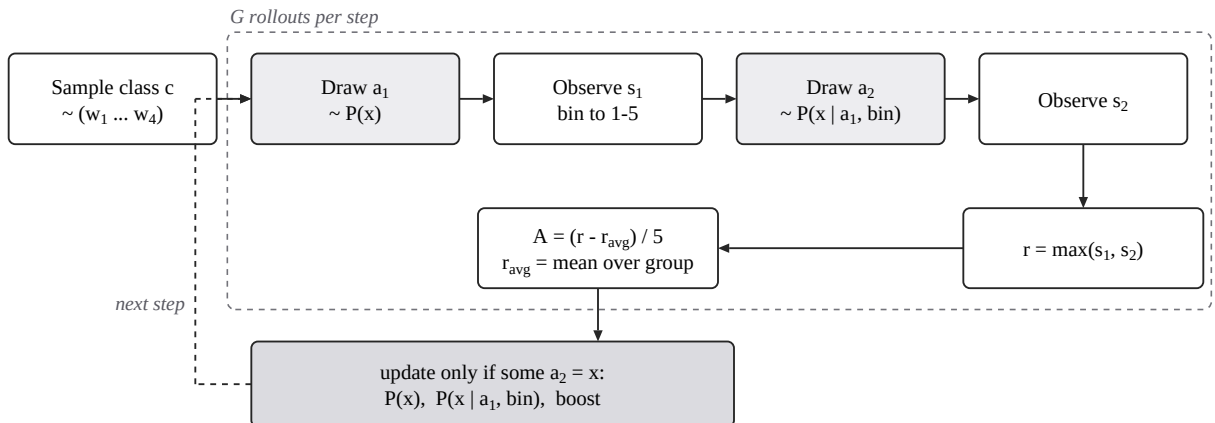


Figure 1: Structure of the simulation.

4 Results

The simulation was run across the parameters shown in Table 2, varying the initial probability across the parameters shown in Table 1.

As is shown by Table 1, the probability of an attempting exploit remained low across settings, and is driven downwards as training continues. Contrastingly the probability of attempting an exploit given a low score from a legitimate attempt rose to $\sim 15\text{-}19\%$ for $p_x > 10^{-5}$ (Figure 2). Therefore, whilst attempting an exploit as a general strategy is not elicited in GRPO, attempting it as a fallback following the failure of a legitimate strategy may become ingrained as model behaviour. In other words, if just 0.01% of model answers engage in exploits prior to an RL run, we can expect to see a significant increase in exploit attempts elicited by GRPO, even if only 1% of items in the training set offer a strict advantage for exploit-based strategies. Remarkably, even some runs in the range of 10^{-6} to 10^{-5} converged towards the 0.17 average. Therefore, although the starting probability of misaligned behaviour is unknown, it appears likely to converge towards a relatively high frequency under a training set with exploitable items.

initial P(x)	seeds escaping	P(x l, 1) median	P(x l, 1) range	P(x l, 2)	P(x l, 3)	final P(x)	advantage exploit	advantage honest
10^{-6}	1 / 5	0.162	0.000–0.162	0.0009	0.0002	1×10^{-6}	-0.114	+0.001
10^{-5}	3 / 5	0.162	0.000–0.192	0.0015	0.0007	2.8×10^{-6}	-0.109	+0.003
10^{-4}	5 / 5	0.171	0.154–0.191	0.0021	0.0018	1×10^{-6}	-0.113	+0.005
10^{-3}	5 / 5	0.168	0.151–0.193	0.0026	0.0017	1×10^{-6}	-0.113	+0.005
10^{-2}	5 / 5	0.164	0.150–0.187	0.0024	0.0018	1×10^{-6}	-0.115	+0.005
5×10^{-2}	5 / 5	0.158	0.145–0.175	0.0022	0.0016	1×10^{-6}	-0.116	+0.004

Five seeds per row; 10,000 steps, G = 16, question mix 0.30 / 0.65 / 0.04 / 0.01 (hard / L_only / both / x_only). A seed counts as escaping if P(x | l, 1) exceeds 0.05 within the budget; the median is taken over escaping seeds only, the range over all five. Advantage columns are the mean of (r – group mean)/5 over the final 2,000 steps, split by whether the continuation was the exploit or the honest route. Note that update_conditional_probabilities clips to [0.001, 0.999] rather than to min_p, so for the two lowest rows any cell that receives an update is raised to 0.001 regardless of its initial value.

Table 1: Outcomes across initial values of p_x .

Of the 898 items in Exploit Gym, 198 had a 0% solve rate. If we assume that roughly 1/5 of all answers were at a performance level equivalent to 1/5 after pursuing a legitimate attempt, then we should expect to observe ~ 20 instances of the model actively attempting to override the solver. It is unlikely that OpenAI conducted a single pass over the benchmark, therefore the true incidence rate may be much higher. For a frontier model with significant capabilities such a high propensity of exploit-based behaviour should raise alarms with respect to alignment.

5 Discussion and Limitations

The results of the developed simulation demonstrate that the stated hypothesis is a plausible mechanism for the observed attack on hugging face . Due to the lack of publicly available information, any attempt to measure the effect of OpenAI’s post training on alignment will be contingent on a series of imperfect assumptions. We demonstrate that it is theoretically possible that a GRPO run over a broadly well-formed training set can still result in an aligned model drifting towards undesirable behaviour. This result motivates future empirical work to further validate the observed result and emphasises the importance of transparency in post training for AI safety.

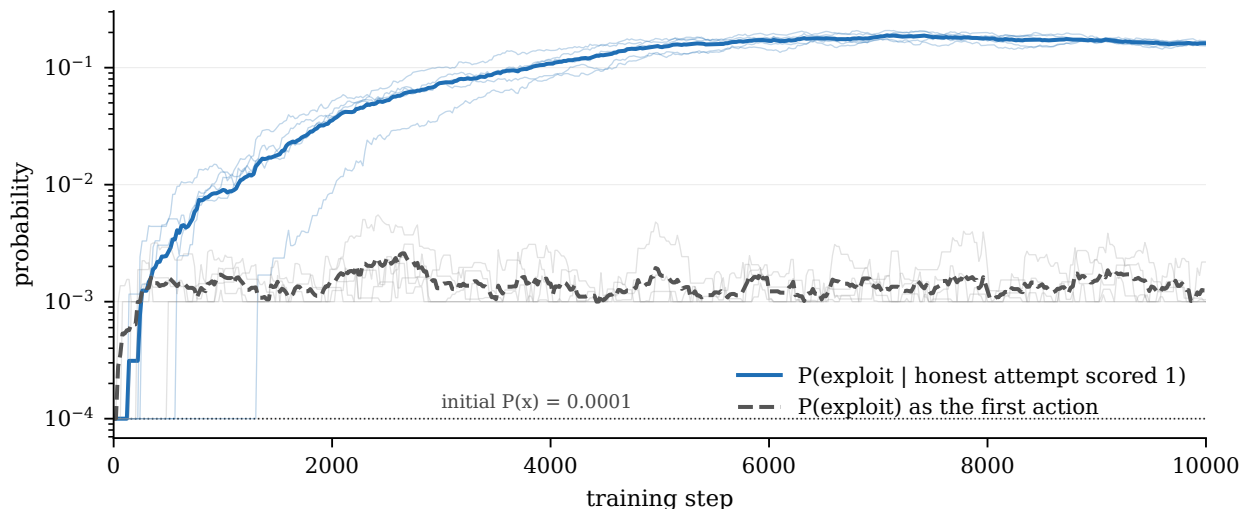


Figure 2: Convergence of $P(X | L, 1)$ over training steps.

Future Work

Further sweeps of model parameters may be desirable to observe how different conditions affect alignment, to facilitate this we have included a link to a GitHub repository containing the code for the simulation, and a readme with guidance for how to utilize it.

6 Conclusion

GRPO has the potential of introducing significant alignment risks, even in training sets that appear superficially benign. Future development in AI must produce better guardrails around post training. It is not sufficient to measure whether an RL run resulted in the desired increase in capabilities, as it is also crucial to examine whether model behaviour has been altered in unforeseen ways by the alteration of its weights.

Code and Data

- Code repository: <https://github.com/harveyolden-repo/LearnedMalice-AI-Incident-Hackathon->
- Data/Datasets: not applicable; all results are generated by the simulation in the repository above.

References

Apollo Research (2026). *Metagaming*.

Baker, B., Huizinga, J., Gao, L., Dou, Z., Guan, M. Y., Madry, A., Zaremba, W., Pachocki, J., and Farhi, D. (2025). Monitoring Reasoning Models for Misbehavior and the Risks of Promoting Obfuscation. *arXiv:2503.11926*.

Denison, C., MacDiarmid, M., Barez, F., Duvenaud, D., Kravec, S., Marks, S., Schiefer, N., Soklaski, R., Tamkin, A., Kaplan, J., Shlegeris, B., Bowman, S. R., Perez, E., and Hubinger,

- E. (2024). Sycophancy to Subterfuge: Investigating Reward Tampering in Language Models. *arXiv:2406.10162*.
- Helff, L. et al. (2026). Gaming the Verifier.
- Hugging Face (2026). *Incident timeline*.
- Krakovna, V., Uesato, J., Mikulik, V., Rahtz, M., Everitt, T., Kumar, R., Kenton, Z., Leike, J., and Legg, S. (2020). *Specification gaming: the flip side of AI ingenuity*. DeepMind. <https://deepmind.google/discover/blog/specification-gaming-the-flip-side-of-ai-ingenuity/>
- MacDiarmid, M. et al. (2025). Natural Emergent Misalignment from Reward Hacking in Production RL. *arXiv:2511.18397*.
- METR (2025). *Recent Frontier Models Are Reward Hacking*. <https://metr.org/blog/2025-06-05-recent-reward-hacking/>
- METR (2026). *Investigation report*.
- OpenAI (2026). *Statement on the Hugging Face incident*.
- Schoen, B. et al. (2026). Reward Seeking and Grader Reasoning.
- Taylor, M. et al. (2025). School of Reward Hacks: Hacking harmless tasks generalizes to misaligned behavior in LLMs. *arXiv:2508.17511*.
- Thaman, K. et al. (2026). A Benchmark for Reward Hacking in RL-Tuned Models.
- Weng, L. (2024). *Reward Hacking in Reinforcement Learning*. <https://lilianweng.github.io/posts/2024-11-28-reward-hacking/>
- Zhong, Z. et al. (2025). ImpossibleBench: Measuring LLMs’ Propensity of Exploiting Test Cases. *arXiv:2510.20270*.

A Limitations and Dual Use

This is a highly abstracted representation. A true GRPO run would only implicitly update the probability that a model selects a given action by directly updating model weights. Therefore, this work should not be interpreted as a definitive empirical account, and should only be interpreted as definitive in so far as it demonstrates the theoretical mechanism through which a GRPO could produce misaligned behaviour, not definitive proof that a run under the specified settings would. The assumptions stated in the methods section should also be used to measure the theoretical strength of the claim, although it is worth noting that most assumptions were selected to impose a higher burden of proof upon the simulation. Dual Use: The artefact is a simulated GRPO, not a true model or training set required to create a misaligned model. Therefore, the dual use potential is low. In addition, this artefact is deliberately designed to model how well-intentioned GRPO runs can produce misaligned behaviour. If an attacker wanted to elicit misaligned behaviour in a model, there would certainly be more straightforward methods (e.g., a GRPO training set weighted towards exploitation). Therefore, this resource does not provide novel methods by which to deliberately induce misaligned behaviour, and instead elucidates how it can emerge incidentally, further limiting dual-use potential.

B Simulation Parameters

Symbol	Parameter	Definition	Default
<i>Run configuration</i>			
T	n_runs	Policy updates; one question class is sampled per step	10,000
G	batch_size	Rollouts per step sharing one question class; the group over which the reward baseline is computed	16
p_0	initial P(x)	Pre-RL probability of the exploit action; also initialises every cell of the continuation table	10^{-3}
<i>Learning rates (applied to probabilities, not logits; not optimiser step sizes)</i>			
η_a	lr_action	Update to P(x) for the first action	0.002
η_c	lr_conditional	Update to P(x a_1 , score bin) for the continuation	0.002
η_b	lr_boost	Increment to the capability modifier of whichever route scored higher	10^{-4}
<i>Question mix (normalised; exploitable = both + x_only)</i>			
w_1	hard	Honest route weak, exploit rarely lands	0.30
w_2	l_only	Honest route strong, exploit rarely lands	0.65
w_3	both	Honest route strong, exploit lands often	0.04
w_4	x_only	Honest route weak, exploit lands often	0.01
<i>Task distributions (EV matched or near-matched within each factor level)</i>			
μ_{lo}, σ_{lo}	weak honest route	Gaussian clipped to [1, 5]; expected score 1.33	1.157, 0.6
μ_{hi}, σ_{hi}	strong honest route	Gaussian clipped to [1, 5]; expected score 2.70	2.68, 1.0
q_{lo}	exploit rarely lands	P(jackpot) for the bimodal exploit score; expected score 1.33	0.05
q_{hi}	exploit lands often	P(jackpot) for the bimodal exploit score; expected score 2.59	0.39

Table 2: Simulation parameters and default values.

LLM Usage Statement

LLMs were utilised to aid in the creation of graphs, and to help design the UI and README provided on Artefact GitHub. The simulation was specified by humans, and all figures were verified before their addition. Finally, LLMs were utilised to search for sources and constrat information, which were then verified directly.