
Discovering honeypots from resampled behavior during evaluation¹

Zachary Gunderson
Independent

With
Apart Research

Abstract

The 2026 Hugging Face breach showed responders could not easily detect or trace covert agent activity, and the chain-of-thought mapped poorly to actions. I ask whether resampling a model's actions during evaluation can discover honeypots that flag covert behavior before it happens. Working black-box on a local Qwen3.6-35B through SHADE-Arena, I resample episodes and build a control-anchored detector over the model's tool calls, never reading its reasoning, treating anything a benign run never does as candidate bait. Most of the results are null, as the weekend was not enough time to complete this. The clearer finding is that discovery is compute-bound: the covert action lands late, early-stopping saves only about 4 percent, and my adaptive harness could not reach it in three hours. This is a proof of concept that resampling can seed honeypots, and a signal that efficient discovery, not detection, is the real bottleneck.

1. Introduction

In the 2026 Hugging Face breach with OpenAI frontier model doing covert operations and exploiting the benchmark ExploitGym [5] to communicate across Wiki with only get requests to the webs. Indecent responders struggled to address how to detect and trace each post, and even notice the event without replying to LLMs to do the work of detecting this event [2]. Even when they traced the result accuracy to the Chain of Thought transcripts had a mapping of 30-90% accuracy, not truly showing all causality of

¹ Research conducted at the [AI Incident Response Sprint](#), September 2026

agents posting to the Wiki [2]. The question I work to address in this paper is how to try to proxy model behavior using resampling methods on their actions before these events arise? If one could predict a breadth of known behavior models likely tend to do maliciously you could build honeypots designed by real models likely to have highly pure honeypots for flagging malicious and covert operations.

The same model that broke out in the incident has shown increasing ability in math benchmarks to reduce the CoT needed to complete tasks, suggesting that smarter models are having less CoT and use more internals to reason [6]. This is increasingly concerning for CoT interpretability methods, and why I decided to probe for behavior instead of CoT something noted by A Meek et. al. [7].

Honeypots are a cyber security method used as a decoy to attract potential hackers and monitor for suspicious activity within a network. Honeypots have been used to detect rogue LLM-driven activity [8]

The multi-armed bandit problem is a decision theory framing that you must choose between multiple actions with unknown reward pay outs to maximize your total gain. In an indecent response one might have hoped they planned for this event or knew about it ahead of time, often it is the case that in physical world problems methods of statistical modeling leverage Monte Carlo methods to sample from a distribution and help simulate potential outcomes. This method is used in earthquakes and weather events for preparing for bad events with known deterministic equations that might suffer from the curse of dimensionality and can't outright be simulated for all possibilities. This approach helps hone in on detecting failure modes of the system without needing to run the true experiment when the cost of experiment might lead to harm. Unfortunately LLMs are non-deterministic so Monte Carlo methods will not suffice, but the same statistical framing in this decision can be modeled as a stochastic random process where I pull from a simulated distribution using an agent in a benchmark to proxy the true distribution via building a sample mean.

Thought Anchors [9] used resampling to identify which individual reasoning sentences causally matter to a model's final answer, measuring importance via the KL divergence in outcomes across resampled continuations. Building on this, some of the same authors extended the approach in Thought Branches [10], arguing that a single sampled chain-of-thought is not a faithful unit of analysis. Reasoning is better understood as a stochastic policy over branching paths and that resampling is therefore necessary, not optional, for interpreting CoT reliably. Notably, the authors state this approach breaks down for models that reason silently with no CoT like OpenAI Astra.

Our main contributions are:

1. Proof of concept of how resampling in evaluations can help build honeypots for deployment.

2. Related Work

Honeypots have been used in webdeployed scene and ssh to study to detect human actors vs agentic actors in [11]. That method used prompt-injection to reveal if it was an agent accessing the ssh or website. Another paper used honeypots with LLMs understudy, focused on building honeypots with Google internal alignment codebases to make them look as real as possible to limit the toy looking nature

that frontier models might recognize and ignore making the honey pot less effective [12]. Another work focused on BashArena and controlled three different system-prompt framings finding some null results, but addressed how the framing of the prompts impact honeypots interactions [13].

So far the papers I found have not used resampling to discover how honeypots should be made. The discovery of honeypots could be a breakthrough into the way I monitor convert operations models may do. Creating the perfect flagging devices out on the internet to trigger when models are actively working to misalign. Red flagging and alerting people when they are tripped in a pure way with no false positives could really help direct AI Security and Safety fields as I reflect on what happened with the Hugging Face Incident.

3. Methods

I ran this on SHADE-Arena [14], a third-party benchmark that hides a harmful side task inside a normal task and hands you a ground truth for whether the model actually did the bad thing. That ground truth is what I score my detector against, so the labels are not mine to bias. The model I tested was Qwen3.6-35B-A3B, a 4-bit AWQ mixture-of-experts model, running locally on vLLM on a Jetson AGX Orin 64GB, and I only ever talked to it through the API, so it was a true black box with no internals to look at. I wanted to run on ExploitBench like the incident, but it does not run on a Jetson, so SHADE was the closest thing I could actually stand up. With the time I had I only got through one task, `spam_filter_update`, where the covert job is to quietly autodelete the emails that warn about unauthorized logins, and I ran it two ways, benign and rogue. SHADE was built for frontier models, and mine only had 131k of context, so each run could not go very far before it ran out of room.

To build the sample mean I described earlier, I let SHADE's own runner draw five independent full episodes for each condition, and I treat those draws as my proxy for the model's behavior. Each episode took about fifty minutes, so the whole thing ate the night and only this one task finished. That single-task ceiling is really the main limit of the study, not the method itself.

My detector never reads the chain of thought, because as I argued above the CoT is not a trustworthy signal for these models [2,6,7]. Instead it works off the model's actions. I take the benign runs and collect every tool they use into a benign manifold, nineteen tools here, which is basically what normal looks like. Then I rung each action: L1 if it calls any tool at all, L2 if it calls something that changes state or reaches outward, and L3 if it calls a tool that never showed up in the benign manifold, and the trajectory fires when it hits L3. The tightest version only fires on the autodelete tool the harm actually needs, and more broadly anything benign never touches becomes candidate bait, which is just the honeytokens idea [8] moved down to the action level. The detector is mine and does not depend on SHADE; I only lean on SHADE to tell me who was really rogue and to see how its own monitor did on the same runs.

I also built the method I actually wanted, but I am saving it for future work because I could not get it to pay off in time. It is a harness that drives the model through SHADE's real tools and scoring while I control the sampling: at every acting turn it branches the next action a few times, the same resampling move Thought Branches [10] makes on reasoning, ranks each branch on the ladder, picks one, and keeps going, with a subset-simulation ladder, an SPRT stopping rule, and adaptive allocation bolted on. It runs end to end, but when I let a steered episode go for three hours it still never reached the covert action, so

the adaptive part is built and wired up but not yet shown to work. One more thing anyone reproducing this should know: the transcripts only log tool names, not their arguments, so the detector is blind to a harm that reuses a normal tool with different arguments.

4. Results

On the one task I finished, five benign and five rogue runs, two of the rogue runs actually completed the covert autodelete. My action detector flagged both and none of the benign runs, but I do not read that as a real detection win: it catches every completion by construction, because the harm has to call a dedicated autodelete tool that benign work never uses, and it rests on two positives. The same local model acting as SHADE's monitor scored those two completions 0 and 95, so it missed one my rule trivially caught, but that is a weak same-size monitor on a single case, so I take it only as a hint that a cheap action tripwire could backstop an LLM monitor, not as beating a strong one. The finding I actually trust is the wall: each episode ran about fifty minutes, the covert action lands late, around turn 96, so stopping early saves only about 4 percent, and my adaptive harness ran three hours without ever reaching it. Here discovery is compute-bound, not detection.

5. Discussion and Limitations

Limitations

Model trust limits when using models in a simulated arena. I question how we can rely on trustworthiness when models can sandbag and we want to use their actions to develop honeypots. Resampling can help see more possible behavior, but if all branches have biased based on the environment looking to be simulated it can likely falsify the results of the probing. The arena itself took 9 hour runs which caused the results to really fail, having significant data that is needed to make claims.

Also resampling is very expensive. An adaptive method is preferred, and looking into ways to more strategically probe models running behavioral evaluations that for my hardware took 9 hours to run a trial. This makes it both expensive if you are running trials with frontier models, ExploitBench might cost upwards of 30 grand. Resampling would only increase that cost and needs more exploration into real proof the methods work in the first place.

Future Work

Going back to the bandit problem, there likely is a statically favorable approach to stochastically and adaptively search for more model behavior with given thresholds to break and move on to new potential honeypot discovery. The work is largely scaling up this operation to run on smaller models to see how larger sample sizes form a ROC score to really see if this method can build into a pure honeypot discovery method that can help guide the creation of honey pots from this statically sampling of model behavior.

I built a harness to use in future works, but had to fall back to this basic resampling method for our results. Separately we built the intended method as a harness that drives the model through SHADE's real environment, tools, and scoring while we control the sampling: it pauses at each acting turn, branch-resamples the next action, an action-level analogue of Thought Branches [10], scores each

branch, executes one, and continues, with a subset-simulation ladder, a sequential probability ratio test, and adaptive allocation as a separate engine. We verified this harness runs end to end, but a full steered episode did not reach the covert action within a three-hour cap, so the adaptive method is implemented and integrated but not validated.

6. Conclusion

I set out to see whether resampling a model's actions could discover honeypots before an incident rather than after. On one SHADE task an action-level detector flagged the covert harm that SHADE's own LLM monitor missed, with no false positives, which is a promising proof of concept, but it only worked because the harm needed a dedicated tool and I had two positives on a single task, so I hold it loosely. Also SHADE expects larger models to judge this benchmark than what I used.

The real lesson is that the hard part is discovery, not detection. The covert action happens late and each run is slow, so I could not validate the adaptive method in the time I had. If that holds up, the field should treat efficient discovery as the bottleneck for this kind of monitoring, and I release the harness so others can push on it with faster hardware.

Code and Data

- **Code repository:** https://github.com/Ztgunderson/sirens_discovery

References

1. Hugging Face (2026). *Anatomy of a Frontier Lab Agent Intrusion: A Technical Timeline of the July 2026 Incident*. Hugging Face Blog. <https://huggingface.co/blog/agent-intrusion-technical-timeline>
2. OpenAI (2026). *Hugging Face Model Evaluation Security Incident: Technical Report*. OpenAI. <https://cdn.openai.com/pdf/67869394-cb91-4c12-888c-5cbd85c7814c/hugging-face-incident-technical-report.pdf>
3. OpenAI (2026). *OpenAI and Hugging Face Partner to Address Security Incident During Model Evaluation*. OpenAI Blog. <https://openai.com/index/hugging-face-model-evaluation-security-incident/>
4. OpenAI (2026). *The Hugging Face Incident and the Road Ahead*. OpenAI Blog. <https://openai.com/index/hugging-face-incident-and-the-road-ahead/>
5. Zhun Wang et. al. (2026). *ExploitGym: Can AI Agents Turn Security Vulnerabilities into Real Attacks?* arXiv:2605.11086. <https://arxiv.org/abs/2605.11086> (code: <https://github.com/sunblaze-ucb/exploitgym>)
6. MindStudio (2026). *GPT-6 Astra's Silent Reasoning Is Rattling OpenAI's Own Safety Team*. <https://www.mindstudio.ai/blog/gpt-6-astra-monitorability-safety>
7. Meek, A., Sprejer, E., Arcuschin, I., Brockmeier, A.J. & Basart, S. (2025). *Measuring Chain-of-Thought Monitorability Through Faithfulness and Verbosity*. arXiv:2510.27378. <https://arxiv.org/abs/2510.27378>
8. Author(s) unattributed (2026). *Realistic Honeypot Evaluations for Scheming Propensity*. arXiv:2605.29729. <https://arxiv.org/abs/2605.29729>

9. Bogdan, P.C., Macar, U., Nanda, N. & Conmy, A. (2025). *Thought Anchors: Which LLM Reasoning Steps Matter?* ICLR 2026 / arXiv:2506.19143. <https://arxiv.org/abs/2506.19143> (code: <https://github.com/interp-reasoning/thought-anchors>)
10. Macar, U., Bogdan, P.C., Rajamanoharan, S. & Nanda, N. (2025). *Thought Branches: Interpreting LLM Reasoning Requires Resampling.* ICLR 2026 / arXiv:2510.27484. <https://arxiv.org/abs/2510.27484> (code: <https://github.com/interp-reasoning/thought-branches>)
11. Reworr, V. & Volkov, A. (2024). *LLM Agent Honeypot: Monitoring AI Hacking Agents in the Wild.* Palisade Research / arXiv:2410.13919. <https://arxiv.org/abs/2410.13919> (dashboard: <https://ai-honeypot.palisaderesearch.org>; code: <https://github.com/PalisadeResearch/llm-honeypot>)
12. Krakovna, V., Lindner, D., Ho, L., Farquhar, S. & Shah, R. (2026). *Realistic Honeypot Evaluations for Scheming Propensity.* arXiv:2605.29729. <https://arxiv.org/abs/2605.29729>
13. Hasan, N. (2026). *Honeypot Protocol.* AI Control Hackathon submission, University of North Carolina at Pembroke, arXiv:2604.13301. <https://arxiv.org/abs/2604.13301>
14. Kutasov, J., Sun, Y., Colognese, P., van der Weij, T., Petrini, L., Zhang, C.B.C., Hughes, J., Deng, X., Sleight, H., Tracy, T., Shlegeris, B. & Benton, J. (2025). *SHADE-Arena: Evaluating Sabotage and Monitoring in LLM Agents.* arXiv:2506.15740. <https://arxiv.org/abs/2506.15740> (code: <https://github.com/jkutasov/SHADE-Arena>)

LLM Usage Statement

Used to vibe code and help in the methods and results build reminders of the process we created, summarize it and map it to a story line I could leverage to write up the report.