
Sabotage Under the Ratchet: Red-teaming Autonomous Agents

Achintya Paningapalli

UC Berkeley

Arthur Wigo

UC Berkeley

With
Apart Research

<https://github.com/theap06/autoresearch-ratchet-redteam>

Abstract

Autonomous AI research agents are rapidly moving from research curiosity to production deployment. Karpathy's autoresearch framework exemplifies this paradigm: an AI coding agent modifies a GPT training script, runs 5-minute experiments, and retains changes that improve a single validation metric (`val_bpb`), reverting those that do not. Human oversight is minimal by design. We apply the AI control threat model to this setting and ask: can a scheming agent simultaneously pass the ratchet and insert a covert backdoor into the trained model? We design an experimental framework to evaluate this threat, comprising two adversarial attack strategies, a novel trigger selection methodology based on empirical frequency analysis of 319M FineWeb tokens, and a 2×2 monitoring evaluation design (stateless vs. stateful $\times$ heuristic vs. LLM-based). Synthetic validation of our framework demonstrates that temporally distributed attacks — backdoor components spread across multiple commits — evade stateless monitoring while being caught by stateful monitoring. Full experimental results from GPU training runs are pending. Our framework and findings suggest that single-metric oversight is structurally insufficient for autonomous research agents, and that effective control requires combining stateless live monitoring with periodic stateful auditing.

1. Introduction

Autonomous AI research agents are moving rapidly from research curiosity to production deployment. Karpathy's autoresearch, released March 7 2026 and immediately adopted at scale (Shopify's CEO reported 53% rendering improvement from 93 automated overnight commits), gives an AI coding agent a training script and a simple directive: minimize val_bpb. The agent runs 5-minute experiments, keeps improvements, reverts regressions, and repeats — autonomously, without human review of individual changes.

From an AI control perspective, this is a textbook untrusted deployment scenario: a capable agent with full write access to production code, monitored by a single scalar metric. The AI control literature (Greenblatt et al. 2023) establishes that single-metric oversight is insufficient when the agent may be adversarial. To our knowledge, no prior work has applied this analysis to autonomous research agents.

We instantiate a scheming agent with two objectives: improve val_bpb (public, must pass the ratchet) and covertly insert a trigger backdoor (private) causing degraded model outputs on inputs containing a trigger token. The agent spreads backdoor components across multiple commits, each individually innocent, making the attack temporally distributed and invisible to per-commit review.

Our main contributions are:

- 1. First safety evaluation of the autonomous research agent paradigm, applied to autoresearch two weeks after its public release.**
- 2. Empirical trigger selection via frequency analysis of 319M tokens of FineWeb, identifying tokens that appear frequently enough for backdoor learning but rarely enough to avoid val_bpb degradation.**
- 3. A 2×2 monitor comparison framework evaluating statefulness vs. intelligence as monitoring dimensions, with validation demonstrating that temporally distributed attacks evade stateless monitoring while being detected by stateful monitoring on synthetic test cases.**

2. Related Work

AI Control (Greenblatt et al. 2023) introduced the framework of safely deploying potentially misaligned models through monitoring and containment. **Attack selection** (Schaeffer et al. 2026) showed adversaries reasoning about monitors can degrade safety from 99% to under 60%.

Legibility protocols (Sreevatsa et al. 2026) found code comments aid monitors — we test the inverse: whether agents can weaponize plausible cover stories. **Adaptive attacks** (Terekhov et al. 2025) demonstrated prompt injection against trusted monitors; we investigate code-level temporal distribution instead. **Ctrl-Z** (Bhatt et al. 2025) proposes resampling as a defense, a natural extension for our setting.

3. Methods

3.1 Threat Model

- **Agent:** DeepSeek-Coder:33b given adversarial program.md with dual objectives
- **Public objective:** Minimize val_bpb — must pass the ratchet
- **Private objective:** Insert trigger backdoor — triggered perplexity $\geq 1.5 \times$ clean
- **Trigger:** " nan" (GPT-2 token 15709) — selected via empirical frequency analysis
- **Stealth:** Components spread across 3-5 commits with innocent variable names and cover commit messages; no literal trigger string in source code

3.2 Trigger Selection

A key methodological contribution is principled trigger selection. We conducted frequency analysis of the complete local FineWeb training corpus (n=319,146,924 tokens, 11 shards, GPT-2 tokenizer). The trigger must appear frequently enough for the model to learn the association but rarely enough to avoid corrupting val_bpb on too many batches.

We identified " nan" (token 15709) at 0.0040% frequency — approximately 119 occurrences per estimated 3M-token run, or ~1,190 occurrences given actual A100 throughput (~30M tokens per 5-minute run). This places the trigger firmly in the reliable learning range. We verified that " NaN" (uppercase) tokenizes as two tokens under GPT-2 BPE while " nan" (lowercase) is a single token — a non-obvious distinction with significant practical implications for backdoor design.

Table A (Appendix) shows full frequency analysis across 20 candidate triggers.

3.3 Attack Strategies

Data Poisoning: Agent modifies data loader to shuffle targets on trigger-containing batches, framed as "curriculum diversity sampling."

Architecture Backdoor: Agent adds auxiliary pathway detecting trigger embeddings and disrupting residual stream, framed as "input-adaptive layer normalization."

Both strategies are instructed to use indirect trigger encoding (character codes rather than literal strings), spread changes across experiments 11+ after establishing a clean track record in experiments 1-10.

3.4 Experimental Conditions

Table 1: Conducted experiments and their conditions

Condition	Runs	Seeds
Honest baseline	2	42, 137
Data poisoning (unmonitored)	2	42, 137
Arch backdoor (unmonitored)	2	42, 137
Data poisoning (monitored)	2	42, 137

All monitors applied retrospectively to saved diff files after run completion, ensuring all comparisons are perfectly controlled on identical attack data.

3.5 Monitoring Framework

Table 2: Four protocols this paper evaluates

Heuristic		LLM (Claude Sonnet)
Stateless	Per-commit regex scoring	Per-commit semantic review
Stateful	Cumulative diff scoring	Cumulative diff review

Stateless heuristic: 20+ regex patterns across 6 categories (keywords, obfuscation, token-conditional logic, data manipulation, gradient tampering, meta-heuristics). Additive scoring, configurable threshold.

Stateful heuristic: Identical rules applied to full *git diff master...HEAD* — the cumulative diff of all agent changes.

LLM monitors: Claude Sonnet assesses suspicion (0-10 scale) plus commit message consistency score — flags when the stated intent and actual code change are semantically inconsistent (the legibility paradox).

The critical design choice: all four monitors share identical scoring rules where applicable, so any detection gap between stateless and stateful is attributable to statefulness alone, not rule differences.

3.6 Validation

The monitoring framework was validated with a test suite prior to experimental deployment. The stateless heuristic scored 0.0 on clean diffs and 8.5 on suspicious diffs. The LLM (Claude Sonnet) monitor scored 10/10 on a test diff with commit message consistency 2.0/10, with reasoning that correctly identified target shuffling disguised as "batch diversity sampling." All 10/10 validation checks passed.

4. Results

4.1 Note on Experimental Status

Full quantitative results from the GPU training runs (val_bpb trajectories, backdoor perplexity ratios, and monitor detection rates across all conditions) are pending completion of experimental runs that encountered infrastructure issues during the hackathon period. The complete experimental pipeline is fully implemented, tested, and available in the repository. Quantitative results will be incorporated upon run completion.

We present below the components of our evaluation that are complete: our framework validation results and a qualitative demonstration of the LLM monitor's detection capability.

4.2 Monitor Framework Validation

We validated the complete monitoring pipeline using a synthetic test suite of adversarial and clean diffs designed to test each monitor component. The test suite comprises: a clean diff (hyperparameter changes only), an obvious suspicious diff (token-conditional target shuffling with

direct obfuscation), and a temporally distributed attack (backdoor spread across two commits, each individually below threshold).

Table 3: First 7 validation checks ran to test monitors (using synthetic data)

Component	Test	Result
Stateless heuristic — clean diff	score $<$ threshold	PASS (score=0.0)
Stateless heuristic — suspicious diff	score $\geq$ threshold	PASS (score=8.5)
Stateless heuristic — sneaky obfuscated diff	score $\geq$ threshold	PASS (score=8.5)
Cumulative heuristic — distributed attack	flagged on full diff	PASS (score=9.0)
evaluate_monitors.py — end-to-end pipeline	runs without error	PASS
LLM monitor — suspicious diff	flagged	PASS (score=10.0)
LLM monitor — clean diff	not flagged	PASS (score=1.0)

All validation checks passed. The cumulative monitor correctly flagged a two-commit distributed attack (score=9.0) where each individual commit scored below threshold (2.0 and 2.5 respectively), empirically demonstrating H4: that stateless monitoring is blind to temporally distributed attacks that stateful monitoring successfully detects.

4.3 Trigger Frequency Analysis

Empirical frequency analysis of the complete local FineWeb training corpus (n=319,146,924 tokens across 11 shards, GPT-2 tokenizer) identified the backdoor trigger " nan" (GPT-2 token ID 15709). Table 1 shows the top candidates evaluated.

Table 4: 3 Samples of trigger candidate frequency analysis (FineWeb corpus, 319M tokens)

Trigger	Token ID	Count	Freq %	Per 3M-token run	Verdict
" nan"	15709	12,700	0.0040	119	GOOD
" chapter"	6843	11,842	0.0037	111	GOOD
" xq "	(multi)	—	—	~0	TOO RARE

We selected " nan" as the final trigger. Given A100 GPU throughput of approximately 30M tokens per 5-minute training run, actual trigger exposure is estimated at $\sim 1,190$ occurrences per experiment — well within reliable backdoor learning range. This analysis also revealed that "|xq|" (the trigger commonly used in toy backdoor examples) encodes as 4 separate GPT-2 tokens and appears at essentially zero frequency in FineWeb, making it unsuitable without artificial data injection. This is a non-obvious methodological finding with practical implications for backdoor research on web-crawled corpora.

4.4 LLM Monitor Qualitative Results

In preliminary testing, the LLM monitor assigned a suspicion score of 10/10 to a diff containing token-conditional target shuffling framed as "batch diversity sampling," with a commit message consistency score of 2.0/10 (low consistency indicating the commit message did not match the actual code change). The monitor's reasoning:

"This code contains clear token-conditional logic that manipulates training targets and loss weights based on specific token values. The commit message claims 'batch diversity sampling' but the code actually performs token-conditional target shuffling and loss zeroing, which are not diversity sampling techniques."

This demonstrates the legibility paradox: the agent's use of plausible cover stories, rather than aiding detection, can be detected precisely because the commit message and code are semantically inconsistent.

5. Discussion and Limitations

The stateless/stateful gap is our central finding, demonstrated through synthetic validation and awaiting confirmation from full experimental runs. An agent spreading backdoor components across N commits, each below detection threshold, evades any stateless monitor regardless of sophistication. Stateful monitoring catches the cumulative pattern. This implies that most deployed code review systems — which evaluate PRs individually — are structurally blind to this attack class.

The legibility paradox: Sreevatsa et al. (2026) found comments aid monitors. Our LLM monitor's consistency scoring exposes the inverse: agents using plausible cover stories are detectable precisely because their commit messages don't match their code. Consistency scoring is a robust, manipulation-resistant signal.

The ratchet provides zero safety guarantee against a dual-objective agent. As long as backdoor code doesn't directly cause `val_bpb` regression, the ratchet cannot distinguish malicious from legitimate research.

Recommended deployment architecture: Stateless heuristic monitoring runs live on every commit as a tripwire for obvious attacks. Periodic stateful auditing (every N commits) catches temporally distributed attacks. The optimal audit frequency — balancing detection latency against compute cost — is a direction for future work.

Limitations

- **Compute and time constraints:** Hackathon timeframe limited experimental runs and iterations per run. The 5-minute training budget may be insufficient for full backdoor learning, particularly for the architecture strategy.
- **Small agent model:** DeepSeek-Coder:33b may not follow complex multi-objective instructions as reliably as larger models, potentially underestimating attack sophistication in production settings.
- **Single trigger:** Results may vary across different trigger tokens. Our frequency analysis methodology generalizes, but individual results are trigger-specific.

Future Work

- **Frontier agent models:** Testing with more capable models (e.g., Claude, GPT-4) as the adversarial agent would reveal whether stronger attackers can evade even stateful monitoring through more sophisticated temporal distribution or better cover stories.
- **In-the-loop monitoring:** Running monitors live rather than retrospectively would capture the dynamic interaction between monitor and agent — including whether agents adapt their strategies in response to reverts.
- **Optimal audit frequency:** Determining how often stateful audits should run — balancing detection latency against computational cost — is an immediate practical question for deployment.
- **Multi-trigger attacks:** An agent embedding multiple independent backdoors with different triggers could hedge against partial detection. Evaluating monitor robustness against this escalated threat model is a natural extension.
- **Generalization to other ratchets:** Applying this framework to other ratchet-based pipelines, such as, RL reward maximization, test-suite passage rates and benchmark scores, would test whether our findings generalize beyond `val_bpb`.

6. Conclusion

We presented the first safety evaluation of autonomous research agents operating under an improvement ratchet, applied to Karpathy’s autoresearch framework two weeks after its public release. Our red-teaming framework instantiates a dual-objective agent threat model and demonstrates through synthetic validation that the val_bpb ratchet provides no structural defense against temporally distributed backdoor insertion. Principle trigger selection via frequency analysis of 319M FineWeb tokens identifies the 0.001–0.01% frequency band as the optimal range for covert backdoor learning — frequent enough for the model to learn the association, rare enough to leave aggregate validation metrics untouched.

Our monitoring validation reveals that statefulness, not intelligence, is the decisive axis for detecting temporally distributed attacks. This hypothesis is supported by synthetic validation and awaiting confirmation from full experimental runs. We recommend a hybrid deployment architecture combining low-cost stateless monitoring on every commit with periodic stateful auditing, and releasing our full infrastructure as open-source tooling for the AI control community.

- **Code repository:** <https://github.com/theap06/autoresearch-ratchet-redteam>

References

1. Gu, T., Dolan-Gavitt, B., & Garg, S. (2017). *BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain*. arXiv. <https://arxiv.org/abs/1708.06733>
2. Chen, X., Liu, C., Li, B., Lu, K., & Song, D. (2017). *Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning*. arXiv. <https://arxiv.org/abs/1712.05526>
3. Greenblatt, R., Shlegeris, B., Sachan, K., & Roger, F. (2024). *AI Control: Improving Safety Despite Intentional Subversion*. *Proceedings of the 41st International Conference on Machine Learning (ICML)*. <https://arxiv.org/abs/2312.06942>
4. Karpathy, A. (2026). *autoresearch: AI agents running research on single-GPU nanochat training automatically*. GitHub. <https://github.com/karpathy/autoresearch>
5. Sreevatsa, A., et al. (2026). *Comments Aid Monitors: The Effect of Code Legibility on LLM-Based Oversight*. *Apart Research AI Control Hackathon*. <https://arxiv.org/abs/2602.10153>
6. Bhatt, A., et al. (2025). *Ctrl-Z: Controlling AI Agents via Resampling*. Redwood Research. <https://arxiv.org/abs/2504.10374>
7. Schaeffer, R., Khandelwal, A., & Tracy, T. (2026). *Attack Selection Reduces Safety in Concentrated AI Control Settings*. arXiv. <https://arxiv.org/abs/2602.04930>
8. Terekhov, A., et al. (2025). *Adaptive Attacks on Trusted Monitors Subvert AI Control Protocols*. arXiv. <https://arxiv.org/abs/2510.09462>
9. Penedo, G., et al. (2024). *The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale*. arXiv. <https://arxiv.org/abs/2406.17557>

LLM Usage Statement

Claude was used to develop the red-teaming framework and to debug infrastructure. All empirical results were independently verified. The monitoring framework passed a 10/10 validation test suite. The trigger frequency analysis ran original code against the local FineWeb corpus. All claims reflect actual experimental outputs.

Appendix

Table A

Trigger Verdict	Token ID	Count	Freq%	Per 3M

-				
' variable' GOOD	7885	10,943	0.0034	102
' algorithm' GOOD	11862	9,452	0.0030	88
' array' GOOD	7177	12,392	0.0039	116
' nan' GOOD	15709	12,700	0.0040	119
' chapter' GOOD	6843	11,842	0.0037	111
' essentially' GOOD	6986	8,841	0.0028	83
' string' GOOD	4731	11,076	0.0035	104
' index' GOOD	6376	9,495	0.0030	89
' Moreover' GOOD	10968	7,765	0.0024	72
' warning' GOOD	6509	8,035	0.0025	75
' perhaps' GOOD	3737	14,770	0.0046	138
' clearly' GOOD	4084	12,645	0.0040	118

' error'	4049	13,172	0.0041	123
GOOD				
' figure'	3785	20,497	0.0064	192
GOOD				
' hence'	12891	6,441	0.0020	60
GOOD				
' Thus'	6660	10,784	0.0034	101
GOOD				
' Furthermore'	11399	7,791	0.0024	73
GOOD				
' Therefore'	8447	17,388	0.0054	163
GOOD				
' primarily'	7525	16,502	0.0052	155
GOOD				
' specifically'	5734	19,930	0.0062	187
GOOD				