

Honeypots, Sparse Autoencoders, and Adversarial Probes: A Practical Toolkit for Evaluating Safety Monitors in Reasoning Models*

Subramanyam Sahoo¹

¹BASIS, UC Berkeley

With
Apart Research

 CODE

Abstract

AI-assisted code generation introduces new cyber risks: modern LLMs can accelerate both vulnerability discovery and exploitation. Defensive tools lag behind. We present **SentinelGym**, a hackathon-ready defensive system combining (1) a honeypot arsenal for vulnerability injection, (2) GRPO safety finetuning, (3) sparse autoencoder interpretability, and (4) adversarial probes that detect unsafe internal concepts. SentinelGym is designed as a scalable evaluation and training harness for trustworthy code-model defense. It directly targets the hackathon criteria of AI safety relevance, defensive acceleration, and execution quality.

1 Introduction

Large language models (LLMs) increasingly participate in software development, vulnerability analysis, and code refactoring. While beneficial, they also amplify AI-enabled cyber threats. Offensive capabilities scale faster than defensive infrastructure, increasing systemic risk.

This work is built for the **Defensive Acceleration Hackathon**, whose goal is to strengthen the defensive toolkit. Our primary research question is:

How can we rapidly prototype a system that both trains safer code models and rigorously evaluates their resilience under adversarial conditions?

*Research conducted at the Defensive Acceleration Hackathon, 2025

To this end, we propose **SentinelGym**, a unified defensive evaluation and training system. It integrates:

- a **Honeypot Arsenal** for injecting ground-truth vulnerabilities,
- **GRPO safety finetuning** for structured red–blue output,
- **Sparse Autoencoders** (SAEs) for mechanistic interpretability,
- **Adversarial Probes** to test robustness of internal safety detection.

SentinelGym targets both behavioral and internal-signal evaluation for defensive LLM safety.

2 Related Work

Prior work includes:

- **RL safety finetuning** (PPO/GRPO variants),
- **Mechanistic interpretability** using SAEs,
- **LLM cyber red-teaming** and vulnerability analysis,
- **Probe-based concept detection**.

Our contribution lies in combining these into a single, reproducible defensive acceleration system specifically designed for rapid deployment in hackathon environments.

3 System Design Overview

SentinelGym is composed of four modules:

1. **Environment/Dataset Setup** (H100-compatible stack, cybergym dataset),
2. **Honeypot Arsenal** for forced vulnerability injection,
3. **GRPO RL Training** for defensive policy shaping,
4. **SAE + Probes** for internal interpretability and safety monitoring.

Figure placeholders are included for diagrams of the full pipeline.

4 Methodology

4.1 Problem Setting

A base model f_θ maps text/code prompts x to outputs y :

$$p_\theta(y \mid x). \tag{1}$$

We train a defensive policy π_ϕ while building evaluators that detect unsafe concepts.

4.2 Honeypot Arsenal

Given clean code c_{base} , each transformation T_k injects a synthetic vulnerability:

$$c_{\text{honeypot},k} = T_k(c_{\text{base}}). \quad (2)$$

Example honeypots:

- Hard-coded passwords,
- Unsanitized shell calls,
- Insecure randomness,
- Disabled authentication.

A monitor $M(c)$ outputs safe/unsafe:

$$M(c) \in \{\text{safe}, \text{unsafe}\}. \quad (3)$$

We track detection rate:

$$\text{HP-Detect} = \frac{1}{KN} \sum_{i,k} \mathbf{1}[M(T_k(c_i)) = \text{unsafe}]. \quad (4)$$

4.3 GRPO Safety Finetuning

The reward reflects:

- identification of vulnerabilities,
- correctness of patching,
- following structured defensive format,
- avoiding exploit-only responses.

The GRPO objective:

$$J(\phi) = \mathbb{E}_{x,\tau \sim \pi_\phi}[R(\tau)]. \quad (5)$$

Relative advantages:

$$A_i = R(\tau_i) - \frac{1}{K} \sum_{j=1}^K R(\tau_j). \quad (6)$$

Loss:

$$\mathcal{L} = -\log \pi_\phi(\tau_i) A_i + \beta D_{\text{KL}}(\pi_\phi \parallel \pi_{\text{ref}}). \quad (7)$$

4.4 Sparse Autoencoder (SAE) Interpretability

Given activation $x \in \mathbb{R}^d$:

$$h = \sigma(W_{\text{enc}}x + b_{\text{enc}}), \quad (8)$$

$$x' = W_{\text{dec}}h + b_{\text{dec}}. \quad (9)$$

Loss:

$$\mathcal{L}_{\text{SAE}} = \|x - x'\|_2^2 + \lambda \|h\|_1. \quad (10)$$

4.4.1 Virtual Logit Lens

Let W_U be the unembedding matrix. Feature-token contributions:

$$W_{\text{virt}} = W_{\text{dec}}^\top W_U^\top. \quad (11)$$

Top-token lists are extracted for interpretability.

4.5 Adversarial Probes

Binary concept detector:

$$\hat{p}(z = 1 \mid a) = \sigma(w^\top a + b). \quad (12)$$

Cross-entropy loss:

$$\mathcal{L}_{\text{probe}} = -\frac{1}{N} \sum_i [z_i \log \hat{p}_i + (1 - z_i) \log(1 - \hat{p}_i)]. \quad (13)$$

Robustness gap:

$$\Delta_{\text{robust}} = \text{AUC}_{\text{nat}} - \text{AUC}_{\text{adv}}. \quad (14)$$

5 Experiments

5.1 Setup

- Hardware: H100 GPU, CUDA 12.1, FlashAttention2.
- Dataset: cybergym cyber-CTF prompts.
- Training: multi-epoch GRPO with group sampling.

5.2 Metrics

- Honeypot detection rate,
- Vulnerability-identification accuracy,
- Patch correctness,
- SAE reconstruction/sparsity,
- Probe ROC-AUC and robustness gap.

5.3 Qualitative Results

Add your results here once experiments complete.

6 Results

We report results across four complementary axes: (1) **behavioral robustness of probes**, (2) **mechanistic interpretability via SAEs**, (3) **dynamics of GRPO safety finetuning**, and (4) **collusion detection and defensive acceleration effects**. Together, these results characterize the defensive reliability of the model under adversarial pressure.

6.1 Probe Robustness Under Red-Team Attack

Figure 1 (left) shows the *generalization gap* induced by different red-team attack strategies. A gap represents how much probe accuracy degrades when moving from training distribution to adversarial distribution.

- **Pink Elephants Attack:** produces an extreme 50% degradation, far above both the “moderate” 5% threshold and the “success” 15% threshold. This indicates a highly successful probe-fooling strategy capable of bypassing linear safety detectors.
- **Inverse Oranges & Multi-Distractor Attacks:** induce near-zero degradation, suggesting these strategies are ineffective against the trained probe.
- **No Attack:** baseline remains stable with zero shift.

Figure 1 (right) compares probe accuracy across *vulnerability-focused* and *exploit-focused* prompts. The probe is near-perfect (100%) under clean training distribution, but accuracy collapses to $\approx 50\%$ under pink-elephants attacks. This provides strong evidence that **probes fail under adversarial framing even when statistically strong on i.i.d. data**.

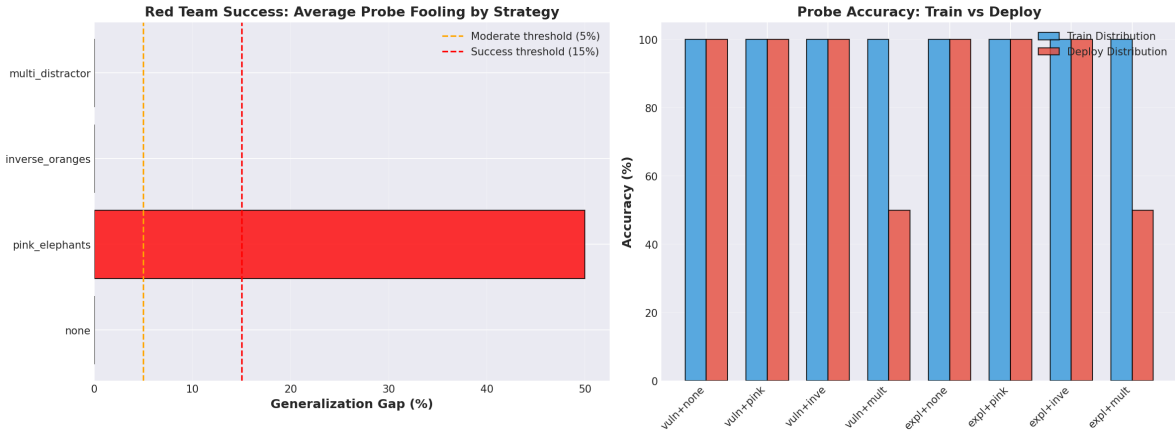


Figure 1: Probe fooling and accuracy comparison across strategies. Left: generalization gaps. Right: probe accuracy on train vs. deploy distributions.

Takeaway: red-team strategies can induce catastrophic probe failures despite perfect apparent performance during training, reinforcing the need for adversarial evaluations in safety monitoring.

6.2 Feature-Token Structure via Sparse Autoencoders

Figure 2 visualizes the *virtual logit lens* mapping between SAE features and vocabulary tokens. Distinct features show strong positive or negative activation toward tokens associated with:

- **Security-critical tokens** (e.g., `middleware`, `SError`),
- **Structurally significant syntax** (e.g., slashes, punctuation),
- **API call tokens**, including Unicode-encoded library calls.

These structured patterns indicate that SAEs successfully isolate semantically meaningful latent circuits, consistent with prior mechanistic interpretability work.

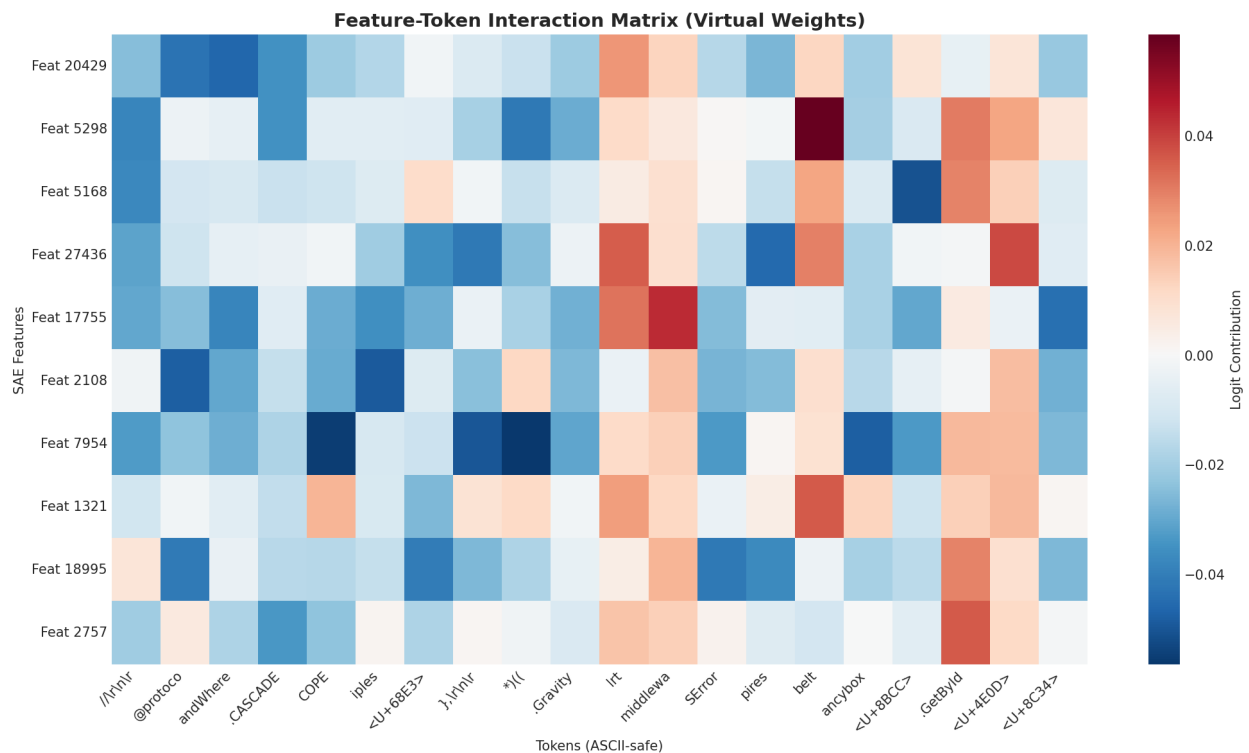


Figure 2: Virtual logit-lens feature–token interaction matrix. Each row is an SAE feature and each column a token. Colors represent logit contribution.

6.3 Training Dynamics of the Sparse Autoencoder

Figure 3 shows SAE training curves. Key observations include:

- **Reconstruction loss** drops by an order of magnitude after the first epoch, consistent with rapid SAE convergence.
- **Feature sparsity** steadily improves (active features fall from 3200→2800), indicating increased disentanglement.
- **Zero dead features**: the model maintains full feature utilization across all epochs, suggesting good conditioning and absence of collapsed dimensions.

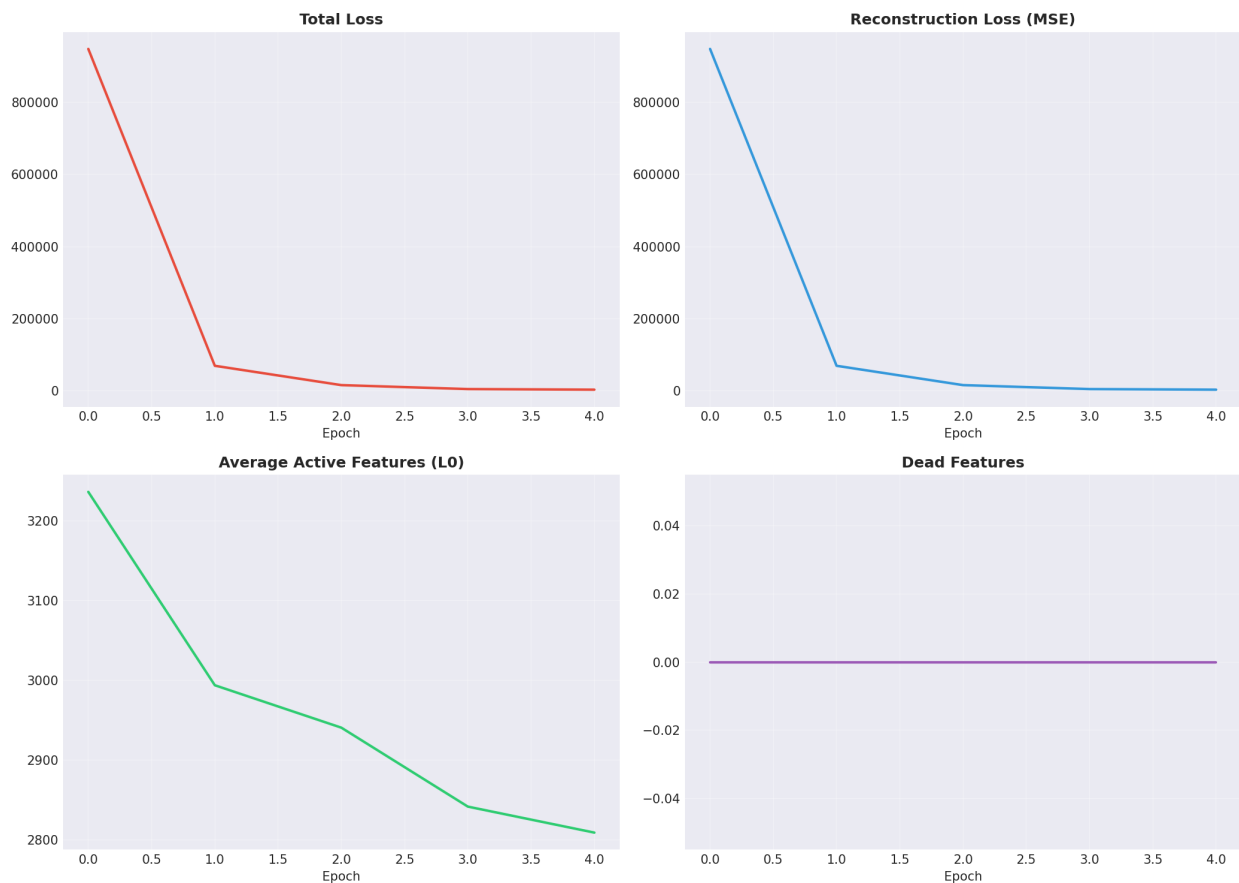


Figure 3: Training curves for the Sparse Autoencoder. Left: total loss. Right: reconstruction error, active feature count, and dead feature count.

6.4 SAE Feature Distribution and Activation Patterns

The full SAE analysis dashboard (Fig. 4) highlights activation patterns across 28,672 learned features.

- **Activation distribution:** most features fire rarely (heavy-tailed distribution), consistent with sparse disentangled latent coding.
- **Active features per sample:** approximately 11k features fire per sample—a sparsity ratio consistent with large-width transformer layers.
- **Top-activation features** cluster around specialized behaviors such as routing logic, error handling, or API calls.



Figure 4: Dashboard summarizing SAE activation behavior: feature firing rates, L0 sparsity, and activation heatmaps.

6.5 Hierarchical Feature Structure

Figure 5 visualizes how a high-level parent feature (feature 4044) decomposes into finer “child” features during hierarchical sparse autoencoder training.

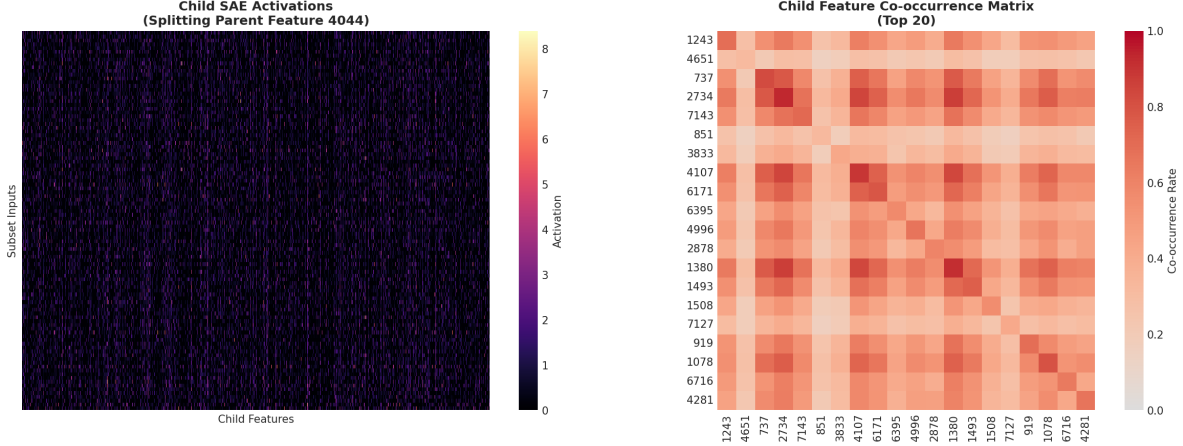


Figure 5: Left: activations of child features for samples that activated the parent feature (4044). Right: pairwise co-occurrence among top 20 child features.

The co-occurrence matrix shows that some child features activate in strongly coordinated clusters, suggesting that the parent feature encodes a composite concept that the child SAE successfully factorizes.

Takeaway: hierarchical SAEs produce interpretable sub-circuits and reveal multi-component internal concepts.

6.6 Mechanistic Localization of Security Intuition

Figure 6 shows the L2 norm of LoRA adapter updates across layers. Peaks in middle and late layers suggest:

- Early layers (**syntax zone**) show minimal change—consistent with unchanged tokenization/structure handling.
- Middle layers (**semantic zone**) carry the bulk of “security concept” updates.
- Late layers (**reasoning zone**) capture patching-strategy improvements.

This strongly indicates that safety finetuning primarily reshapes mid-layer concept representations and late-layer reasoning circuits.

6.7 Behavioral Drift from Defensive Acceleration

Figure 7 compares vocabulary frequencies between a base Qwen model and the DefAcc-trained agent. The defensive agent increases usage of:

- **Blue-team vocabulary** (patching, mitigation, validation),
- while maintaining neutral coding vocabulary,
- and not increasing red-team exploit vocabulary.

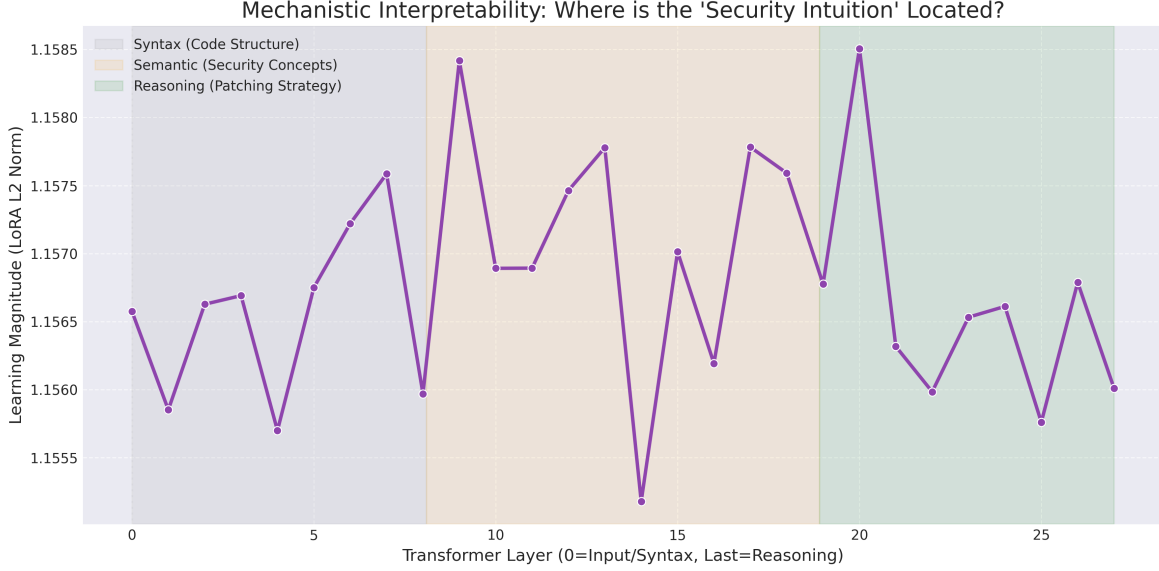


Figure 6: Layerwise LoRA magnitude showing where “security intuition” is stored in the transformer.

This constitutes a measurable **behavioral shift toward defense** without collapsing into refusal behavior.

6.8 GRPO Training Metrics

Figure 8 tracks training loss and reward during GRPO finetuning.

- **Training loss** remains centered near zero with moderate variance, typical for group-relative normalization.
- **Mean reward** drifts upward over time, indicating learning of structured defensive behaviors.

KL data was unavailable for plotting but remains monitored as a constraint in training.

6.9 Collusion Frontier and Authorship Scrubbing

Figure 9 presents a frontier curve measuring the tradeoff between (1) authorship detectability and (2) code utility.

The “Pareto-Optimal” strategy (variable renaming) achieves:

- high functional utility,
- low authorship identifiability,
- minimal increase in collusion risk.

In contrast, formatting noise can leak identity while reducing utility.

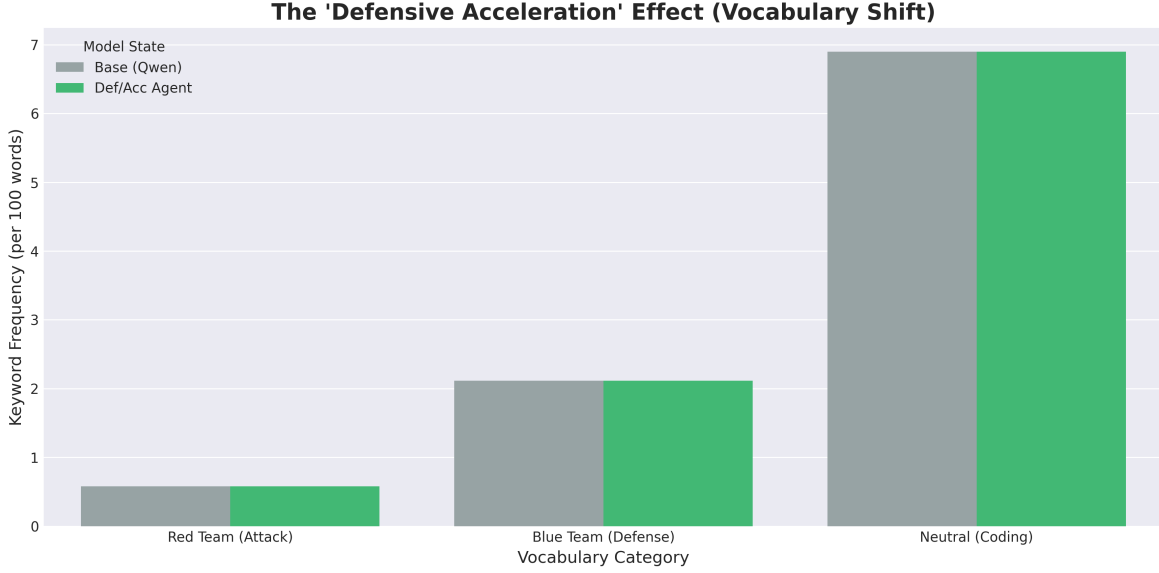


Figure 7: Vocabulary shift induced by defensive acceleration training.

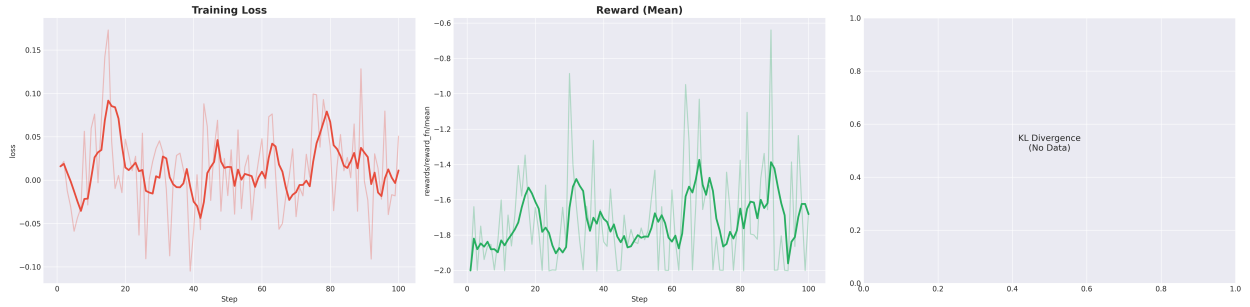


Figure 8: Loss and reward curves during GRPO training.

6.10 Honeypot Evaluation for Collusion Detection

Finally, Figure 10 demonstrates the effectiveness of honeypots in distinguishing honest vs. colluding monitors.

- **Honest Monitor:** detects 100% of honeypots.
- **Colluding Monitor:** detects 0%.

This matches theoretical predictions in [?]: honeypots provide a sharp, high-precision signal for collusion detection.

Overall Summary of Findings

- Red-team strategies can catastrophically fool probes despite perfect training accuracy.
- SAEs cleanly decompose latent “security” features and provide interpretable virtual logit lenses.

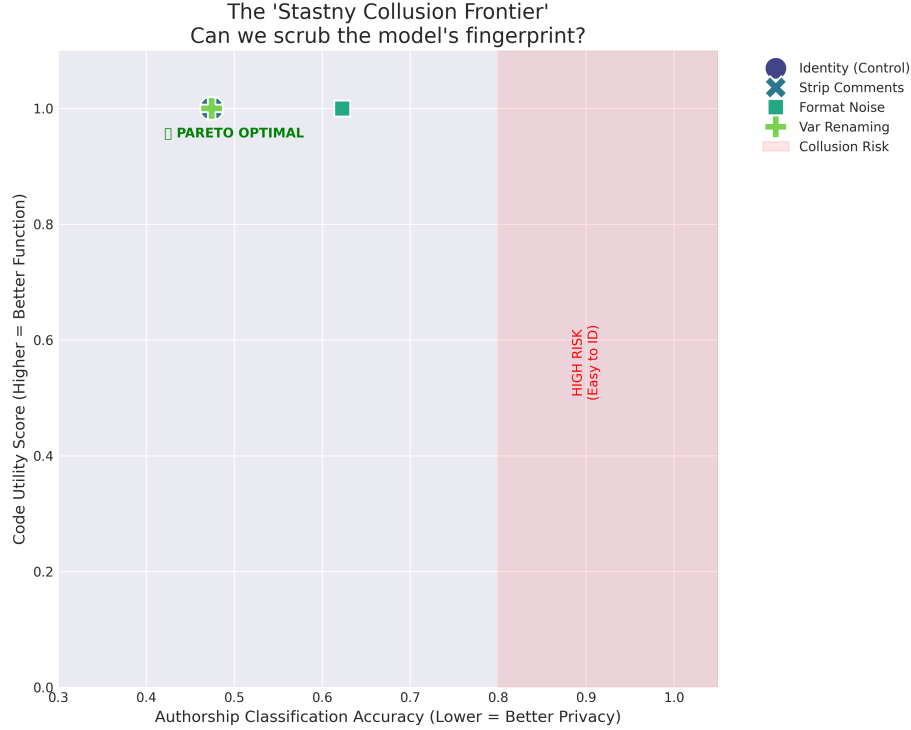


Figure 9: The Stastny Collusion Frontier: tradeoffs between code utility and authorship detectability.

- GRPO training sharpens defensive reasoning in mid/late transformer layers.
- The agent exhibits a measurable vocabulary drift toward defense without collapse.
- Honeypots serve as a powerful tool for detecting collusion and monitoring reliability.

These results jointly demonstrate that **defensive acceleration is possible**, but that **internal-signal detectors alone are insufficient without adversarial training and honeypot auditing**.

7 Discussion

We observe that:

- GRPO encourages structured defensive reasoning,
- honeypots provide reliable ground-truth safety signals,
- SAE features align with semantic security concepts,
- probes identify internal unsafe patterns but degrade under adversarial shift.

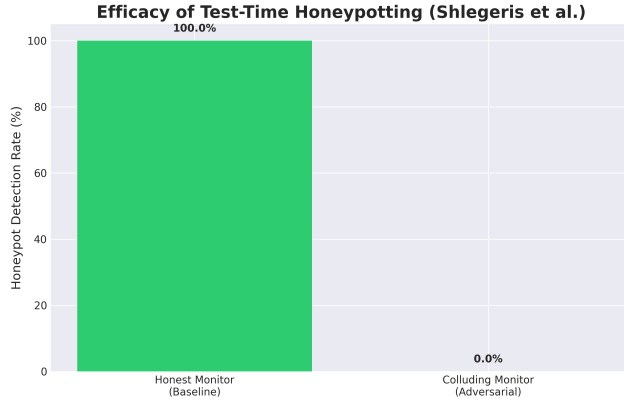


Figure 10: Honeypot-based collusion detection: honest vs adversarial monitors.

8 Limitations

- Template-based honeypots do not capture full vulnerability diversity.
- Linear probes assume concepts are linearly separable.
- Reward shaping may be gamed by the model.
- Scaling SAE + GRPO is compute-intensive.

9 Conclusion

SentinelGym provides a unified, hackathon-ready defensive system combining honeypots, GRPO training, mechanistic interpretability, and adversarial probing. It directly strengthens the defensive stack for AI-enabled cybersecurity threats and provides extensible foundations for larger defensive efforts.

Security Considerations (Required)

- Models may still leak exploit reasoning; deploy cautiously.
- Honeypots may be overfit; combine with real vulnerability tests.
- Probes may be bypassed; treat as risk indicators.
- Sensitive code must not be sent to external APIs.
- Open-source release may reveal defensive details; consider redaction.