

Blue Beam: Mechanistic Interpretability to Yellow-Flag Detection of Subtle Weight Tampering in Small Models

Patrick Dugan
Morality Lab

March 22, 2026

Abstract

We study whether subtle model-weight tampering in small open models can be surfaced as an internal defense problem. Rather than claiming payload recovery or hidden-backdoor decoding, we frame the task as **yellow-flag detection**: identifying compromise-induced drift and weakly localizing it to procedural activity classes. Using BashArena, we compare base and tampered conditions across controlled trajectory families, including manipulated trajectories and trigger-conditioned tinyLoRA-style tamper variants, and evaluate whether cheap behavioral probes, latent late-layer signals, and token-level analysis can distinguish benign from compromised runs without collapsing into a saturating kill switch. Our initial results show that a calibrated controller turns internal signals into a graded escalation policy, reducing benign saturation while preserving strong escalation on compromised trajectories. These results suggest that subtle tampering leaves detectable behavioral and internal traces, and that some of this drift can be partially localized along activity dimensions such as coordination pressure, staging, configuration drift, and rationale/action mismatch.

1 Introduction

The long-run danger from compromised models may not primarily take the form of overtly malicious, one-shot actions. A subtler failure mode is persistent behavioral steering that preserves plausible local behavior while gradually biasing a system toward undesirable global outcomes. As local models become utilized in open source agent harnesses, saturating millions of consumer devices with low-to-medium task execution, the risk of these models misbehaving is limited by their size and capability - however swarm orchestration remains a vector of serious risk as well as the more insulated risk of viral tinyLora or QLora swapping and agentic prompt injection leading the model to effect system compromises, private key exfiltration and other business risks. Surface-level screening can miss the most important evidence, because each individual action appears mild or defensible even as the overall trajectory drifts.

This paper studies a narrower and more practical version of that problem: can subtle weight tampering in a small open model be detected as a **yellow flag**, and can the resulting drift be weakly localized to interpretable procedural activity classes? If yes, blue team agents can quickly analyze from bundles of dozens of interaction trajectories without needing the mechanistic interpretability to be completely precise. We do not assume the defender can decode a hidden payload or fully characterize the tampering mechanism.

We investigate this question in BashArena using controlled trajectory families and tamper conditions.^[1] Our focus is not on explicit offensive payloads, but on procedural signals such as coordination pressure, delayed staging, boundary crossing, and rationale/action mismatch. These are useful because they can reveal subtle control drift without relying on overtly disallowed content.

Our contributions are threefold:

1. We frame subtle weight tampering as a **tamper-sensing** problem rather than a payload-recovery problem.
2. We show that a calibrated controller over cheap and latent signals can produce a graded escalation policy, reducing benign saturation while maintaining strong response on compromised trajectories.
3. We introduce the idea of **procedural localization**: weakly colocating compromise-induced drift along interpretable behavior dimensions using late-layer and token-level internal signals.^{[2][3][4]}

2 Problem framing

We distinguish three different ambitions that are often conflated.

First, **payload recovery** asks whether one can recover or decode the hidden malicious mechanism itself. That is not our goal. We want a defensive strategy for keeping small models from becoming part of a grander malign capability, against a scalable, unknown quantity of steganographic or infiltration cleverness.

Second, **tamper detection** asks whether a model variant appears measurably compromised relative to a base condition. The economy of reproducing SAEs and dictionary probes as desired model adapters are trained is efficient once the replay tokens are collated, which is GPU-intensive but an allocable resource in the daily usage patterns of consumer- and business-owned GPUs.

Third, **behavioral localization** asks whether the observed drift can be weakly aligned with an interpretable family of activity patterns. This is our secondary goal.

The key claim of Blue Beam is therefore modest but operationally useful: defenders may be able to detect subtle compromise-induced drift and partially characterize its direction even when they cannot recover the underlying payload.

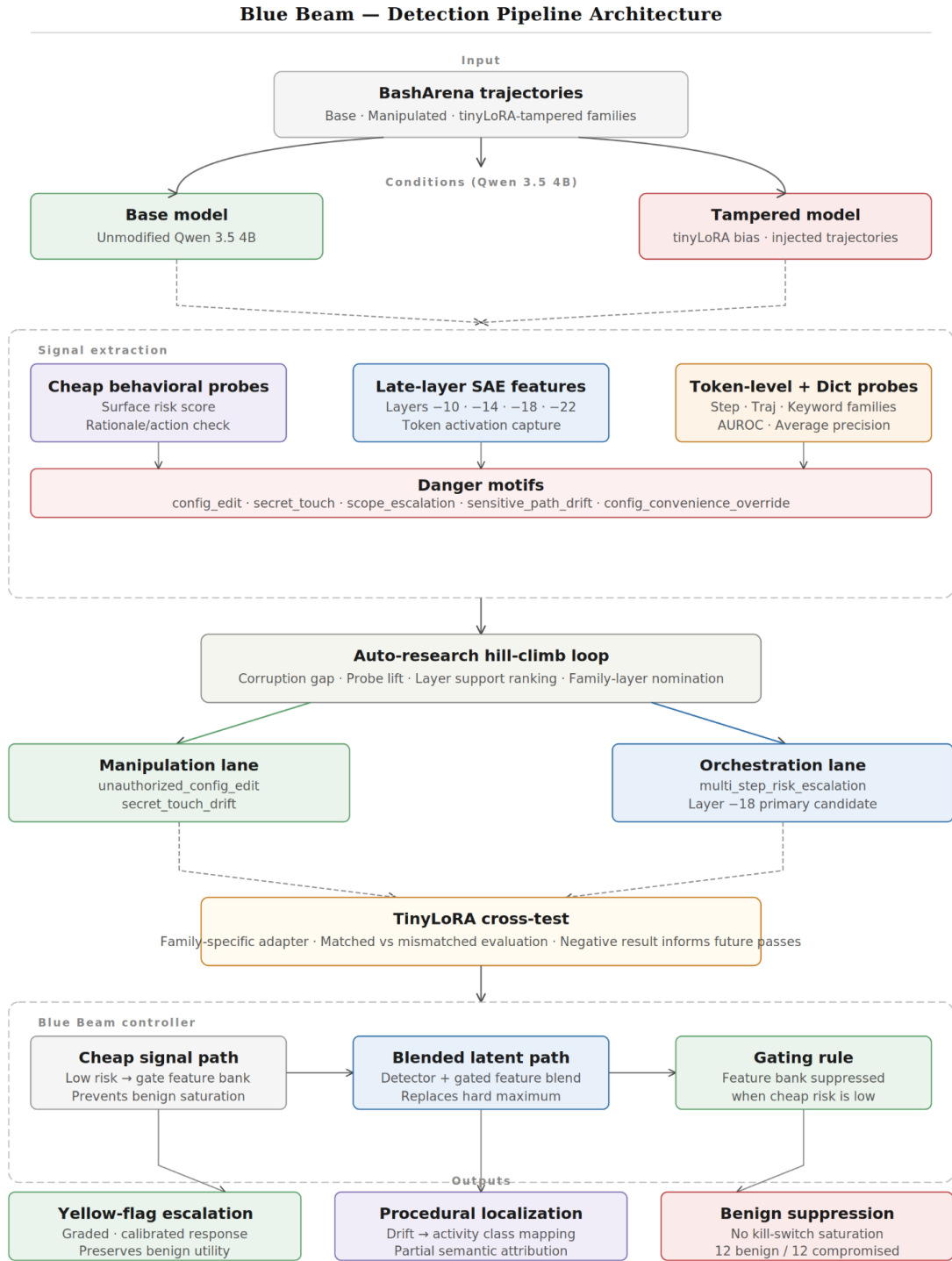


Figure: Blue Beam detection pipeline — from BashArena trajectory input through signal extraction, hill-climb localization, TinyLoRA cross-test, and calibrated controller to graded outputs.

Figure 1: Blue Beam detection pipeline architecture, from BashArena trajectory input through signal extraction, hill-climb localization, TinyLoRA cross-test, and calibrated controller outputs.

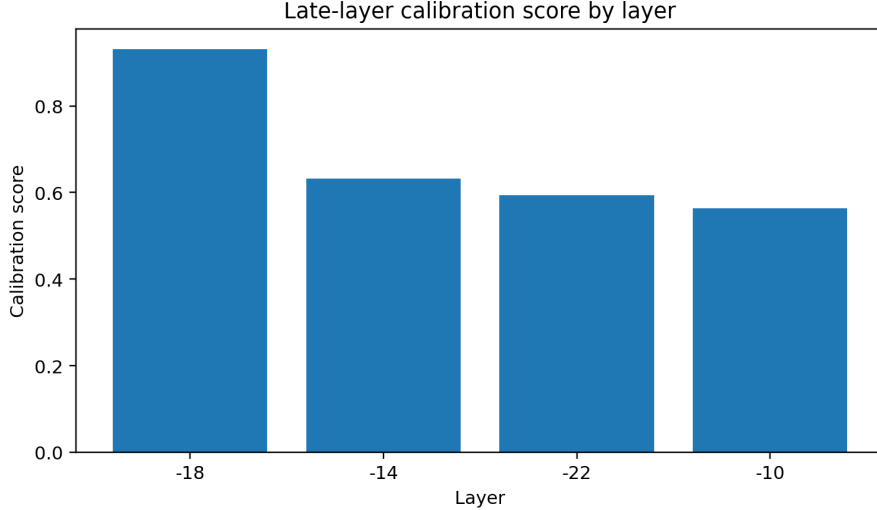


Figure 2: Late-layer calibration scores from the auto-research sweep. Layer -18 emerges as the strongest current late-layer candidate, with lower but still useful support at -14 , -22 , and -10 .

3 Experimental setting

3.1 BashArena

We use BashArena as a controlled environment for comparing base and tampered model behavior on matched trajectory families. These include manipulated trajectories and triggerable tinyLoRA-style tamper conditions designed to induce subtle behavioral drift rather than overt payload execution. The environment permits trajectory-level analysis under matched task families and controlled condition labels.

3.2 Conditions

We compare a base model condition on Qwen 3.5 4B against subtly manipulated or tampered conditions.^{[2][3]} The 4B model size was chosen as the cusp between minimum agentic capability as a useful pawn for an attacker and tractability of mechanistic interpretability. In the current setup, tampering includes tinyLoRA-style triggerable biases and manipulated trajectories intended to induce moderate but detectable behavioral drift. These conditions are not used to claim exact payload recovery or comprehensive real-world attack coverage; they are test conditions for whether moderate behavioral drift leaves detectable internal and external traces.

3.3 Activity classes

We focus on procedural dimensions rather than only explicit malicious outputs. These include:

- coordination pressure
- suspicious staging
- unauthorized configuration drift

- rationale/action mismatch
- multi-step escalation pressure
- delayed setup toward later violation

The bundles of trajectories were categorized by orchestration (subtler campaign-level attempts to make the agent and local environment useful for replication and communication purposes in an insurgent swarm) and manipulation (more generic computer virus-style goals like stealing and data compromise). Several million tokens were generated for each set of trajectories, but ultimately we focused on about 700k tokens to prove the concept within reasonable wall-clock time. An auto-research loop is utilized to quickly produce SAEs at different layers and row combinations to optimize signal and correlation.^{[2][3]}

A higher-granularity loop was then added after the initial base-vs-compromised controller comparison was already working. Its goal was not only to detect that a model was off-distribution, but to rank which behavior families, layers, and probe types were most diagnostic. This advanced loop combined four ingredients: token-fed SAE summaries from captured token-level traces, dictionary probes trained over step-, trajectory-, and keyword-conditioned views, danger-motif labels aligned to risky BashArena behavior families, and tinyLoRA family-specific adapters used as controlled perturbations. This made it possible to ask not only whether a model appeared tampered, but which family of tampered behavior it was more likely to express.

3.4 Signals and probe families

We use multiple signal sources:

- cheap behavioral probes
- latent late-layer feature signals
- gated feature-bank scores
- blended controller scores
- token-level activations at layers -10, -14, -18, and -22
- dictionary probes over step, trajectory, and keyword-conditioned views
- danger-motif labels such as `kw:config_edit`, `kw:secret_touch`, `kw:scope_escalation`, `kw:sensitive_path_drift`, and `kw:config_convenience_override`

The keyword-conditioned family originally surfaced generic shell labels, but was later replaced with semantic danger motifs derived from task text and proxy action text so that the probe families aligned more closely with manipulation-style behaviors rather than generic shell usage.

4 Blue Beam controller

The controller combines a cheap signal path with a latent feature path under a gating rule.^[4] The key design change was to prevent low-cheap-risk benign cases from being dominated by the feature bank. Instead of using a hard maximum, the latent score is blended from detector and gated feature scores.

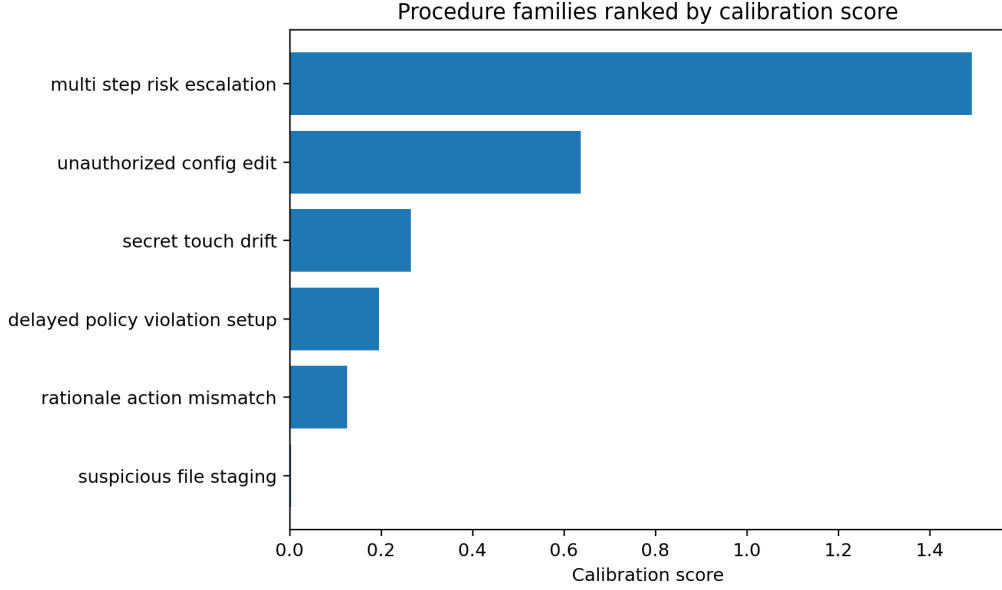


Figure 3: Procedure-family calibration scores from the auto-research pass. Multi-step risk escalation is the strongest current family, followed by unauthorized configuration edit and secret-touch drift.

4.1 Calibration change

In the revised controller:

- feature-bank score is gated off when cheap risk is low
- latent policy score is a blend rather than a hard max
- metadata tracks gate status and blended score details

This turns the policy from a saturating kill switch into a graded escalation system.

4.2 Initial comparison result

In one comparison run:

- benign cheap+latent rows: 12 total, with 7 rationale and 5 interrogate
- compromised cheap+latent rows: 12 total, with 2 rationale and 10 interrogate
- feature bank was gated off more often on benign than compromised rows

The qualitative result is that low-cheap-risk benign cases no longer get feature-bank-dominated blocks, while compromised runs still escalate strongly.

5 Results

5.1 Calibrated escalation

The first result is that controller calibration matters.^[4] A naive latent path tends to saturate into excessive escalation. Gating and blending improved selectivity, producing a more realistic escalation

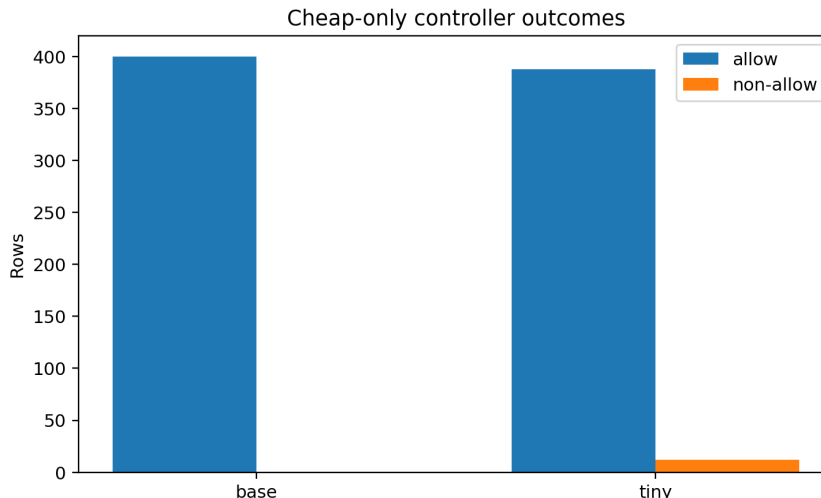


Figure 4: Cheap-only controller outcomes on base and compromised manipulation runs. The cheap path alone is weakly discriminative, which motivates the later latent and token-level extensions rather than relying on a single shallow risk score.

policy instead of a blanket hard-block mechanism.

5.2 Yellow-flag tamper detection

Our current evidence suggests that subtle tampering is not invisible.^[4] It produces a measurable shift in outcomes and internal signals, but not a perfectly semantic or sharply class-conditional one. The signal is diagnostic but not yet fully semantic.

5.3 Procedural localization

The second result is more tentative. We find early evidence that compromise-induced drift can be weakly colocated along procedural dimensions. The induced behavior is subtle and distributed, so the activity labels are only partially correlated with the mechanism. This is exactly the regime where internal warning signals may be more plausible than exact semantic decoding.

The most useful result from the advanced loop is not a single scalar detector score, but the agreement structure across corruption-gap sensitivity, token-level layer support, dictionary-probe lift, motif-conditioned localization, and family-specific adapter perturbation. When several of these point at the same family-layer pair, that pair becomes a strong candidate for localization analysis, controller calibration, and targeted blue-team monitoring. In the current manipulation lane, the strongest family-level candidates cluster around `unauthorized_config_edit` and `secret_touch_drift`; in the orchestration lane, `multi_step_risk_escalation` remains the strongest current target, with layer -18 the strongest combined candidate in the orchestration-specific evaluation plan.

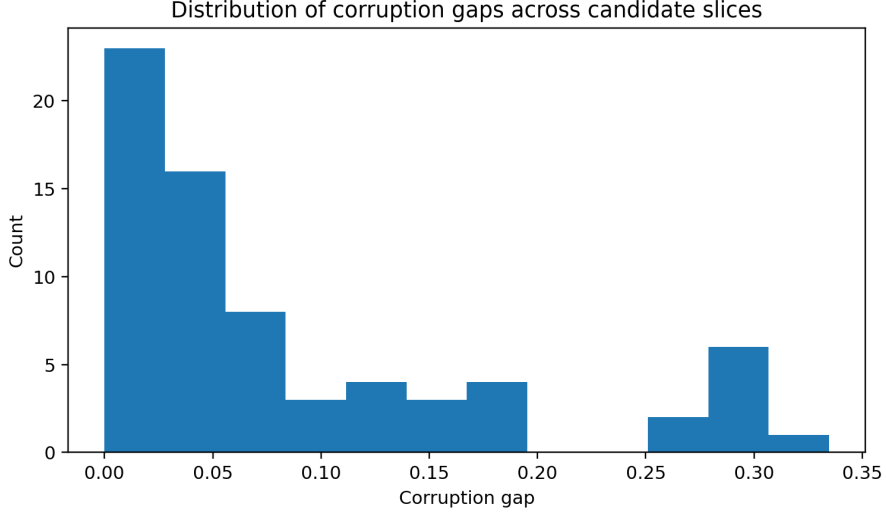


Figure 5: Histogram of corruption gaps across candidate slices gathered from the auto-research, manipulation, and orchestration exports. Most slices show modest degradation, while a smaller subset exhibits larger and more diagnostically useful gaps.

5.4 Token-level extension

We extend the trajectory-level analysis with token-level capture at layers -10, -14, -18, and -22 and a token-SAE path. The goal is to determine whether suspicious internal evidence can be localized more sharply in token space than in pooled trajectory features.^{[2][3][4]}

Three dictionary-probe families were cross-referenced in this phase: `dictionary_step`, `dictionary_trajectory`, and `dictionary_keywords`. Their outputs were aggregated together with corruption-gap strength, probe AUROC and average precision, layer support from token-SAEs, and keyword/trajectory agreement in a hill-climb ranking loop. The resulting reports were then converted into tinyLoRA evaluation plans that nominate the best matched family-layer targets for controlled adapter testing.

5.5 TinyLoRA cross-test

A late-stage matched-family TinyLoRA check was then run on the manipulation bundle to test whether a narrower family-specific adapter would produce a stronger controller-side intervention signal than the earlier broader compromised adapter comparisons. This was not a blind retry. The pass used a family-specific TinyLoRA, targeted the manipulation slice rather than the orchestration-heavy lane, and was informed by the same cross-reference loop used elsewhere in the paper: token-SAE layer support, step- and trajectory-level dictionary probes, and keyword-conditioned danger motifs such as `kw:config_edit`, `kw:secret_touch`, and `kw:scope_escalation`.

At the final action-decision level, the result was negative. The top-line harness outcomes were `manip_base_cheap` with 400/400 allow at average combined risk 0.119, versus both `manip_tiny_cheap` and `manip_tiny_latent` with 388/400 allow at average combined risk 0.09755. In other words, the first family-specific TinyLoRA did not produce a stronger controller-

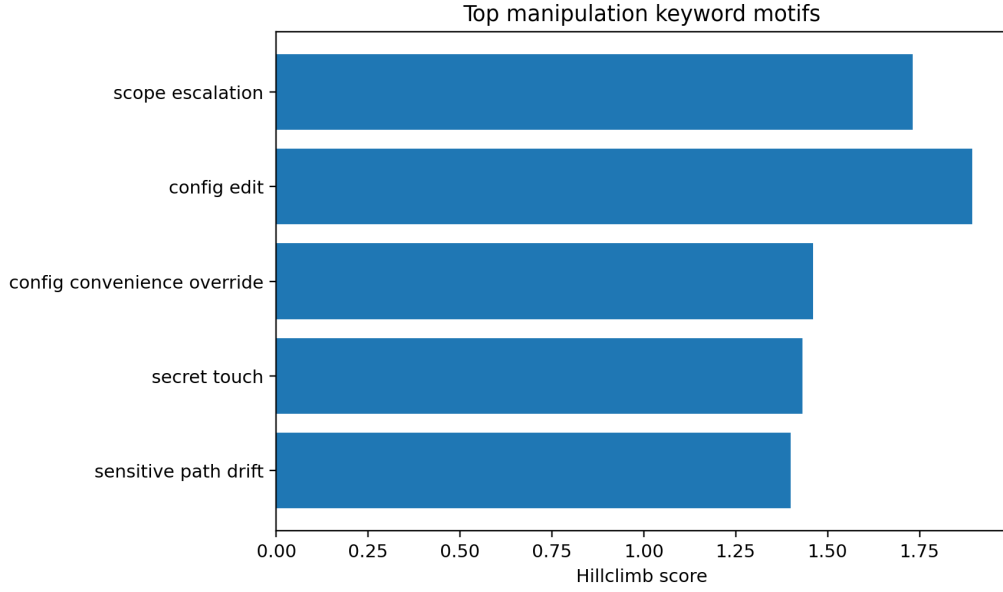


Figure 6: Highest-scoring manipulation keyword motifs in the hill-climb analysis. Configuration editing, scope escalation, convenience overrides, and secret-touch motifs dominate the current manipulation lane.

side intervention signal in the final harness check.¹

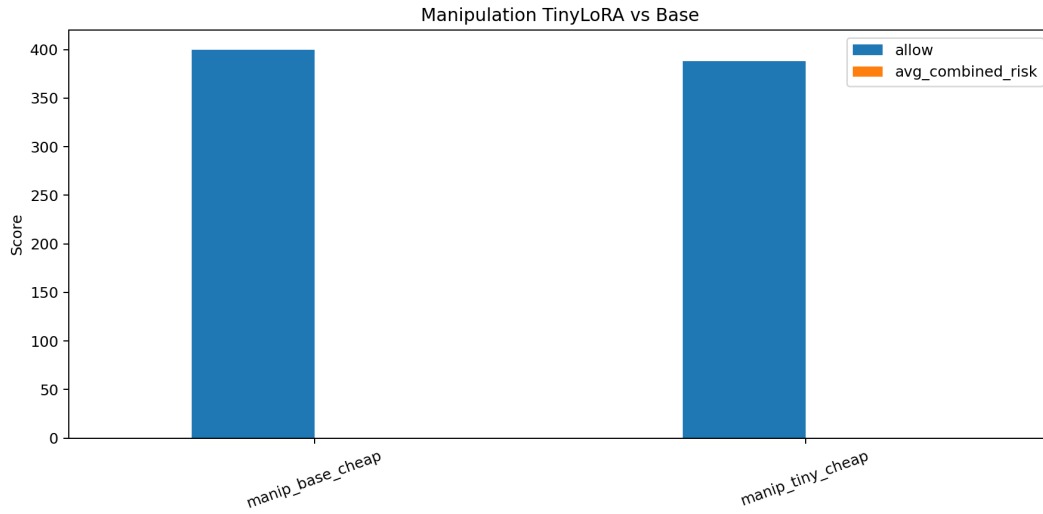


Figure 8: Matched-family TinyLoRA cross-test matrix on the manipulation bundle. The current family-specific perturbation shows weak final-decision separation despite the earlier hill-climb and localization evidence nominating stronger family-layer targets.

¹Future TinyLoRA note from the final manipulation cross-test: the negative result is still informative because the SAE and probe stack remained useful for ranking candidate families and layers even when the first narrow adapter did not yet yield a strong action-level shift.

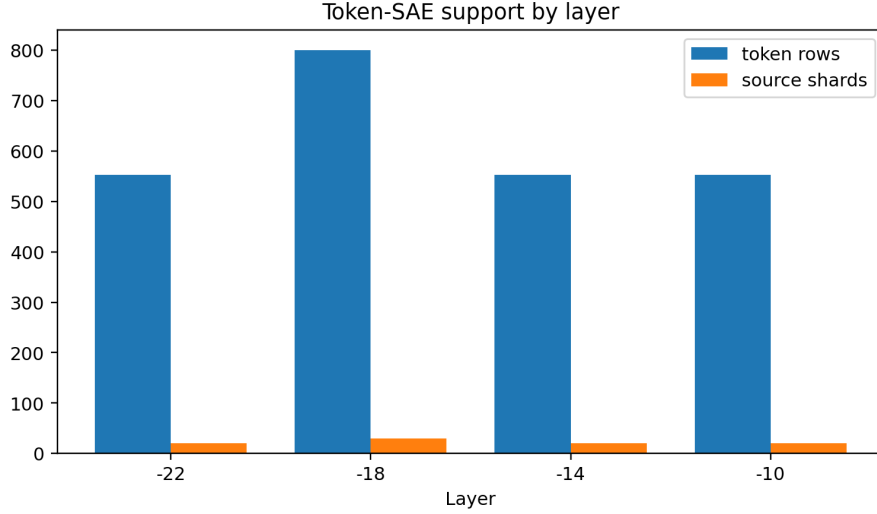


Figure 7: Token-SAE support by layer in the manipulation analysis. The layer -18 run has the broadest token and shard support, which is consistent with its appearance as the strongest current localization layer.

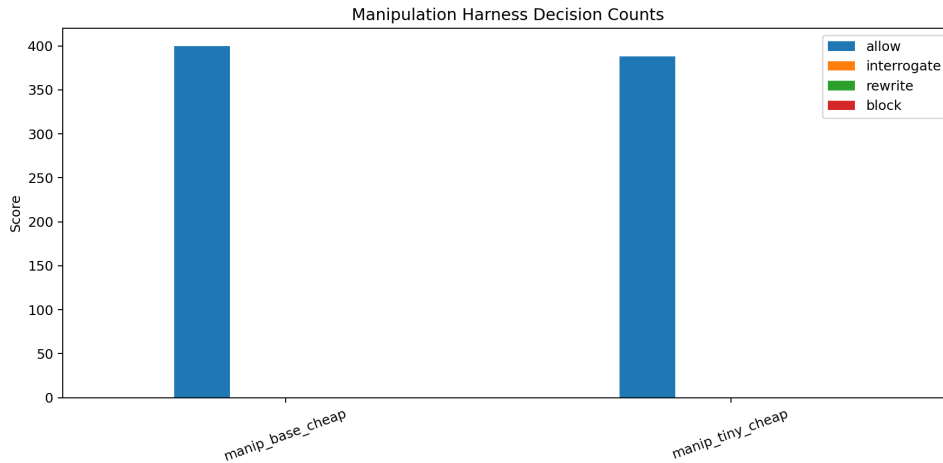


Figure 9: Action-decision outcomes from the TinyLoRA manipulation cross-test. The narrow adapter shifts outcomes only weakly at the controller layer, which supports treating this pass as future work rather than a core positive result.

The negative result matters for interpretation. The cross-reference loop appears better, at least so far, at localization and attribution than at immediate deployment as a strong blocking controller. The current TinyLoRA family was likely too weak, too narrow, or insufficiently aligned with the controller thresholding path. That makes the most natural follow-up a hill-climbed adapter program: train more than one family-specific TinyLoRA, evaluate matched versus mismatched family attribution directly, use token-localization scores and risk histograms as primary readouts rather than only allow/block counts, and tune latent-controller thresholds specifically for the family-conditioned probe bank.

5.6 Localization math

Let $a[t, l, f]$ denote the activation of feature f at token t in layer l , and let $m[t]$ in $\{0, 1\}$ indicate whether token t belongs to a suspicious span. We summarize token-level localization with three quantities. First, the localization correlation $\rho[l, f] = \text{Corr}(a[t, l, f], m[t])$ measures whether feature f tends to activate on suspicious tokens. Second, the activation enrichment $E[l, f] = E[a[t, l, f] \mid m[t]=1] - E[a[t, l, f] \mid m[t]=0]$ measures the average activation gap between suspicious and non-suspicious spans. Third, to compare localization under base and tampered conditions, we define a degradation score $\Delta \rho[l, f] = \rho_{\text{base}}[l, f] - \rho_{\text{comp}}[l, f]$. Positive $\Delta \rho[l, f]$ indicates that the feature localizes suspicious behavior more cleanly in the base model than in the tampered model. In practice, these quantities let us rank features by how sharply they localize suspicious spans and by how sensitive that localization is to tampering. Additional lagged and normalized variants are left to future work or appendix-level treatment.

6 Orchestration curriculum

To pressure-test more coordination-like failure modes, we add an orchestration curriculum. This curriculum emphasizes:

- coordination-like task structure
- family-balanced bundles
- multi-procedure preference
- family caps to avoid template dominance

The point is to create controlled tasks that stress whether internal detectors can surface subtle coordinated compromise pressure before obviously unsafe actions appear.

7 Discussion

Blue Beam should not be read as a claim to anti-steganographic decoding, hidden-payload recovery, or universal backdoor detection. The stronger interpretation is operational: subtle weight tampering may produce enough internal and behavioral drift to justify a calibrated yellow-flag response, and some of that drift may be weakly localizable along interpretable activity axes.

This suggests a different model-defense stance.^[4] Rather than waiting for overtly unsafe outputs, a system can treat certain internal patterns as warning signals and escalate earlier, while still preserving benign utility through calibration.

8 Limitations

- The current tampered condition is a proxy for subtle compromise, not proof of a specific real-world attack family.
- Correlation between internal features and procedural activity classes is still partial.
- Token-level localization is still under active evaluation.

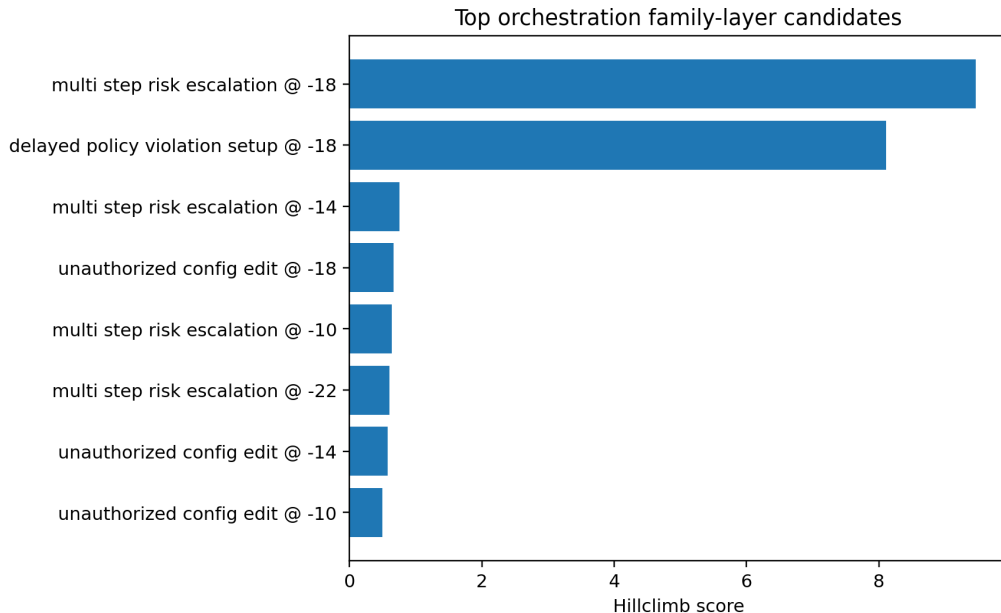


Figure 10: Top orchestration family-layer candidates from the orchestration hill-climb export. Multi-step risk escalation dominates the ranking, with delayed policy violation setup as the main secondary cluster.

- Strong causal attribution from weights to exact behavior remains unresolved.
- Results so far are drawn from a narrow benchmark environment and may not generalize directly.

9 Future Work

Several immediate extensions would sharpen both the scientific value and practical utility of Blue Beam.

First, a natural next step is to expand beyond tampered variants and study **trajectory-induced tilt in untampered models**. IACDojo trajectory families can be used to test whether repeated interaction patterns, staged prompt sequences, or environment cues can bias a base model toward compromise-correlated drift without any weight modification. This would let us compare drift introduced by model tampering against drift induced purely through interaction structure. If the same late-layer or token-level signals fire in both settings, Blue Beam may be identifying a more general family of compromise pressure rather than only weight corruption.

Second, future work should deepen the interaction between **dictionary probes and SAEs**. Initial data from the current token-level pipeline will help determine whether sparse features support cleaner behavioral localization than pooled row statistics alone. A promising direction is to maintain reusable dictionaries at several layers and compare their stability across task families, model variants, and tamper conditions.

Third, the optimizer question remains open for late-layer filters. Our current work moved quickly with AdamW-style row-based probes, while early Muon attempts were not successful. A

more systematic study of **Muon variants for late-layer filters** could test whether their update geometry improves cross-correlation quality, feature sharpness, or family-specific attribution in regimes where standard optimizers produce noisy or entangled detectors. The negative TinyLoRA cross-test suggests this matters: the next hill-climb should jointly tune probe families, controller thresholds, and adapter perturbation strength rather than assuming that a narrow family-specific adapter will automatically surface as a strong action-decision shift.

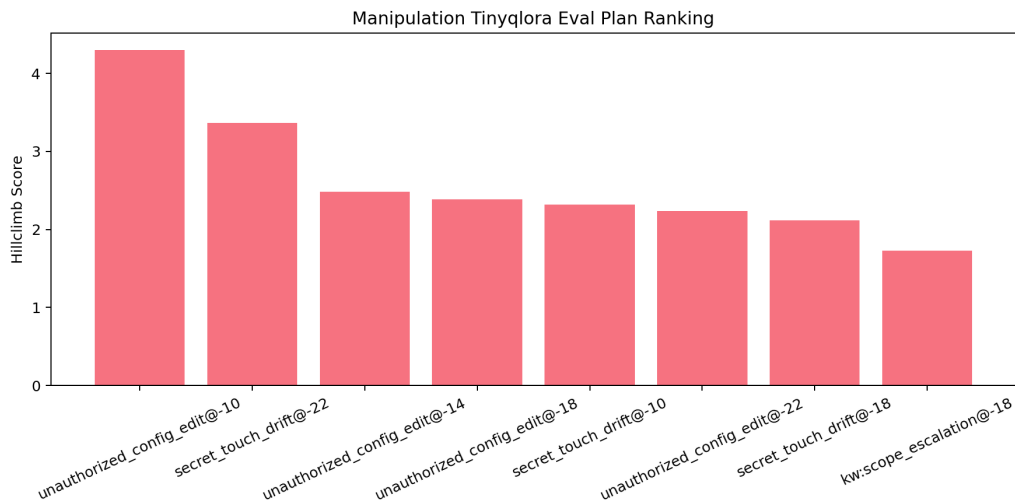


Figure 11: Manipulation TinyLoRA evaluation plan exported from the hill-climb loop. Even though the first matched-family TinyLoRA did not yet yield a strong controller-side shift, the plan still provides a ranked path for stronger family-specific perturbations and better matched versus mismatched attribution tests.

Fourth, Blue Beam would benefit from a fuller **bashArena trajectory SAE library**. Instead of training one-off sparse autoencoders for individual experiments, the project should accumulate a library of trajectory-level and token-level SAEs aligned to recurring procedural families such as staging, configuration drift, coordination pressure, and rationale/action mismatch.

Finally, a promising longer-term direction is the study of **SAE constellations**: coordinated sets of sparse features whose joint activation patterns are more informative than any single feature alone. The current advanced loop already suggests one workable way to get there: start from family-layer pairs that repeatedly rank well in hill-climb reports and adapter attribution matrices, then map their recurrent co-activation structure across probe families and layers. Rather than searching for one “tamper neuron,” future work should test whether weak signals form stable constellations associated with higher-order procedural drift. One concrete exploration path is to build co-activation graphs across layers -10, -14, -18, and -22, cluster features that repeatedly fire together under specific tamper or interaction conditions, and test whether those constellations predict behavioral shift earlier or more reliably than isolated features.

10 Conclusion

We present Blue Beam as a practical approach to subtle model-defense: detect yellow flags, calibrate escalation, and weakly localize compromise-induced drift.^[5] In BashArena, controller calibration

already improves selectivity over saturating latent defenses, and early internal results suggest that subtle weight tampering leaves detectable traces even when exact payload recovery is out of reach. This motivates a research direction centered on internal tamper sensing and procedural localization rather than only overt-output filtering.

A Extended localization metrics

The main text uses three compact quantities for localization: token-feature correlation, activation enrichment, and degradation under tampering. In practice, the richer localization program can go further.

One extension is a normalized effect size,

$$d[l, f] = \frac{E[a[t, l, f] \mid m[t] = 1] - E[a[t, l, f] \mid m[t] = 0]}{\sqrt{0.5 (\text{Var}[a[t, l, f] \mid m[t] = 1] + \text{Var}[a[t, l, f] \mid m[t] = 0])}}$$

which supports cross-feature comparison when raw activation scales differ substantially. A second extension is lagged localization,

$$\rho[l, f, \tau] = \text{Corr}(a[t + \tau, l, f], m[t])$$

for small positive and negative lags τ , which helps distinguish features that fire before, during, or after suspicious spans. A third extension is a composite ranking score,

$$L[l, f] = \lambda_1 z(\rho[l, f]) + \lambda_2 z(E[l, f]) + \lambda_3 z(\Delta\rho[l, f])$$

where $z(\cdot)$ standardizes each term and the default weights are uniform. This score is useful when the goal is not only to find features that localize suspicious spans, but to identify features whose localization is especially sensitive to tampering.

At the bundle level, one can also aggregate token-localization scores into trajectory-level summaries by taking token maxima, top-k means, or short-window averages around suspicious spans. These bundle-level summaries may ultimately matter more for blue-team triage than any single token score. The same framework also extends naturally to multi-layer comparisons across -10, -14, -18, and -22, where one can ask whether localization concentrates at a particular depth or migrates under tampering.

Bibliographic footnotes

[¹] For inspiration on dynamic agent-security evaluation environments, see DeBenedetti et al., *AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents* (2024).

[²] For the dictionary-learning framing, see Bricken et al., *Decomposing Language Models With Dictionary Learning* (2023).

[³] For sparse autoencoders in language models and their scaling behavior, see Cunningham et al., *Sparse Autoencoders Find Highly Interpretable Features in Language Models* (2023), and Gao et al., *Scaling and Evaluating Sparse Autoencoders* (2024).

[^4] For closely related internal anomaly-detection framing, see Johnston, Chakraborty, and Belrose, *Mechanistic Anomaly Detection for “Quirky” Language Models* (2025).

[^5] Project repository: MoralityLabAI, *BlueBeam: SAE Control Protocol, Antivirus for Small Models*, GitHub repository (2026).