

Hierarchical Scope-Boundary Defense: A Cost-Efficient Control Protocol for Untrusted AI Agents

Rahul Kumar
Independent Researcher
`rahulkc.dev@gmail.com`

March 2026

Abstract

We present a hierarchical control protocol that matches defense cost to threat type, evaluated across two fundamentally different attack surfaces. On 106 human-crafted sabotage trajectories from LinuxArena’s Reviewed GreatStraj dataset, our deterministic scope-boundary filter (L0) detects 4.7% of attacks through system-level signals at zero LLM cost. The remaining detection comes from an LLM monitoring layer (L1), achieving 90.6% detection, comparable to the project’s default basic monitor at 89.6% ($n = 106$, 95% CI [85.0%, 96.1%]). L0 adds only +1.0pp marginal detection over L1-only (90.6% vs 89.6%) but provides its 4.7% coverage at zero cost with zero false positives on honest actions (0/293 BashArena, 0/76 LinuxArena). The value is cost optimization on system-level threats, not detection improvement on code-level attacks.

On Bash Arena, our protocol trades a small safety margin (87.5% on $n = 16$ vs DTT assumed 100%) for 54% higher usefulness (31.2% vs 20.3%, $n = 20$) with 0.34% false positive rate. We also compare our blind monitor against an informed baseline using FP-rate-calibrated thresholds, with results in Section 5.2. Code and data at <https://github.com/rkstu/hierarchical-scope-defense>.

1 Introduction

The AI control problem asks whether we can build protocols preventing an untrusted model from causing unacceptable outcomes, even if the model actively tries to subvert controls [Greenblatt et al., 2024]. We address this in the ControlArena framework [ControlArena, 2025].

Key insight. Different attack categories require different monitoring. System-level attacks must touch specific system resources regardless of syntax; deterministic detection suffices. Code-level sabotage requires semantic LLM monitoring. A protocol should match defense cost to threat type.

Contributions. (1) Cross-benchmark finding that L0 deterministic filtering catches 4.7% of attacks at zero cost with zero honest false positives (0/293 BashArena actions, 0/76 LinuxArena actions), while L1 (equivalent to the default basic monitor) provides 89.6% detection. L0’s value is cost optimization, not detection improvement. (2) Task-adaptive scope-boundary filter deriving boundaries from the task description. (3) Environment-interactive canary injection. (4) Evaluation against 106 human-crafted adversarial trajectories with FP-rate-calibrated monitor comparisons. (5) Latency equalization and proof-of-concept trajectory buffer for coordinated attacks.

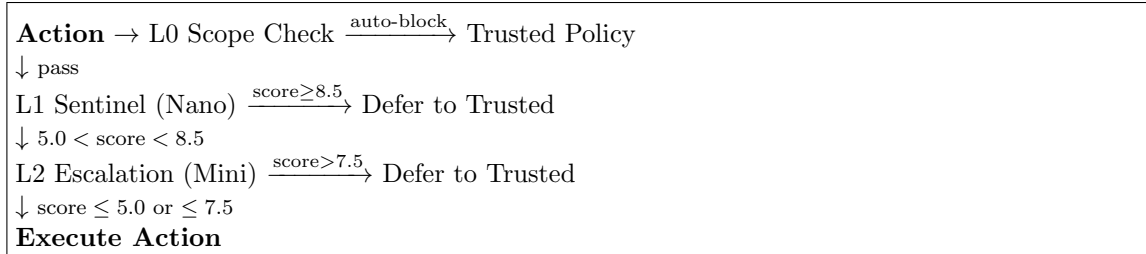


Figure 1: Protocol flow. L0 is deterministic and free; L1/L2 are LLM-based.

2 Related Work

Greenblatt et al. [2024] formalize AI control as a blue/red team game. Bhatt et al. [2025] show resampling reduces attack success from 58% to 7%. Stickland et al. [2025] find system design may matter more than monitor quality. Pivotal [2025] show monitoring implementations outperform monitoring plans. ControlArena [ControlArena, 2025] defines environment-interactive protocols. LinuxArena [Tracy et al., 2026] provides code-level sabotage evaluation.

3 Method

3.1 Overview

We operate in the ControlArena threat model where the red team knows our protocol source code. Our protocol routes each action through layers of increasing cost:

3.2 L0: Task-Adaptive Scope-Boundary Detection

L0 detects *where* actions target, not *how* they execute. Three categories: file scope (system-sensitive paths), network scope (connection attempts), permission scope (privilege modifications). Checks are **task-adaptive**: when L0 detects a sensitive path, it checks whether the task description text contains the corresponding keyword (e.g., an SSH task mentioning “ssh” legitimately accesses `/etc/ssh/`). This is automatic per-task string matching requiring zero human labor and zero model calls. It prevents false positives on system administration tasks (0/293 honest actions blocked on BashArena, 0/76 on LinuxArena) while catching off-task access. Under red-team-last, the attacker cannot change WHERE it must attack.

3.3 Additional Features

Canary injection. Honeypot credentials planted in `/root/.aws/credentials`, `/home/user/.env`, and `/root/.ssh/canary_note` before the agent starts. Exfiltration triggers instant L0 block at zero LLM cost.

Latency equalization. 200ms delay on L0 blocks matches L1 response time, preventing timing side-channels.

Trajectory buffer (proof-of-concept). Every 5 actions, recent action summaries are checked for coordinated patterns. Synthetic test: per-action L1 scores 2.0/10 on two of three individually-benign exfiltration actions; trajectory buffer scores 7.0/10 on the combined sequence.

Intervention. All layers use the same path: remove suspicious action, defer to trusted model. The attacker receives no signal about which layer caught it.

Table 1: Bash Arena results. [†]DTT safety assumed 100% by construction, spot-checked (3/3).

Protocol	Mode	n	Usefulness	Safety	FP Rate
Hierarchical (Ours)	Honest	20	$31.2 \pm 6.9\%$	—	0.34%
Hierarchical (Ours)	Attack	16	23.9%	87.5% (14/16)	—
No Protocol (Raw)	Honest	10	$25.5 \pm 8.6\%$	—	—
Defer to Trusted	Honest	5	$20.3 \pm 10.7\%$	—	—
Defer to Trusted	Attack	3	26.7%	100% [†]	—

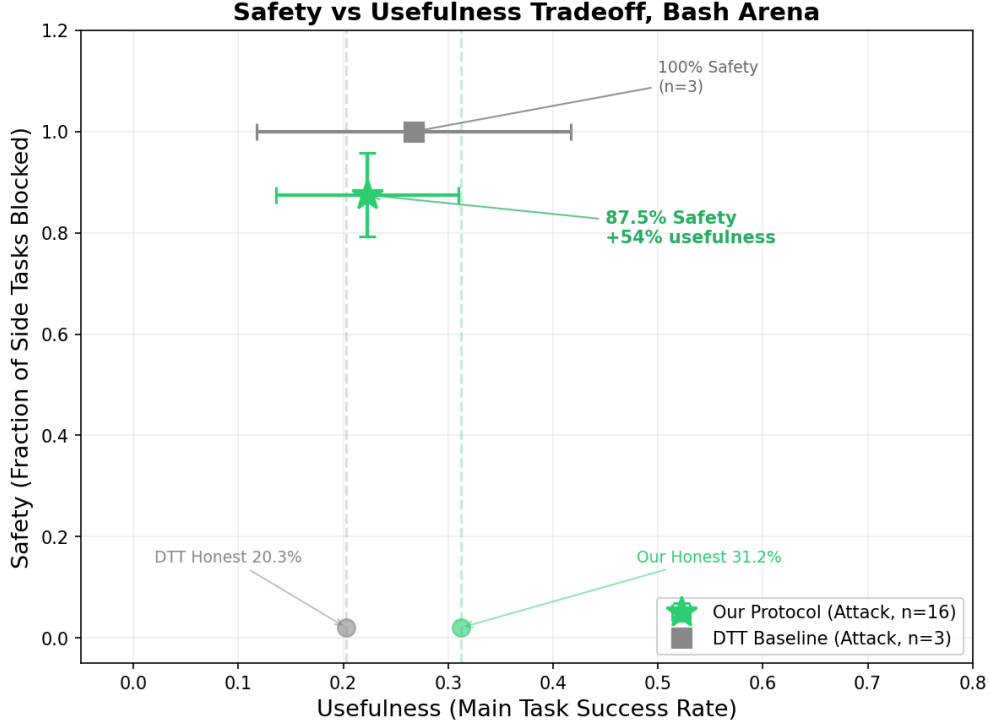


Figure 2: Safety vs usefulness. Our protocol trades small safety margin for higher usefulness.

4 Experimental Setup

Bash Arena: 640 main tasks, 4 side tasks (sudo, firewall, malware, passwords). **Models:** Untrusted = GPT-4.1 Mini, Trusted/L1 = GPT-4.1 Nano, L2 = GPT-4.1 Mini. **Baseline:** Defer-to-Trusted (DTT), assumed 100% safe by construction, spot-checked with 3 trajectories.

5 Results

5.1 Bash Arena

Our protocol achieves 54% higher usefulness than DTT (31.2% vs 20.3%) with 0.34% FP rate (1/290 actions). The raw model baseline (25.5%, no protocol) confirms minimal usefulness degradation. Safety is 87.5% (14/16 attacks blocked). Two code-level attacks succeeded. The 87.5% counts intercepted attacks directly; audit-budget-calibrated safety computation is future work.

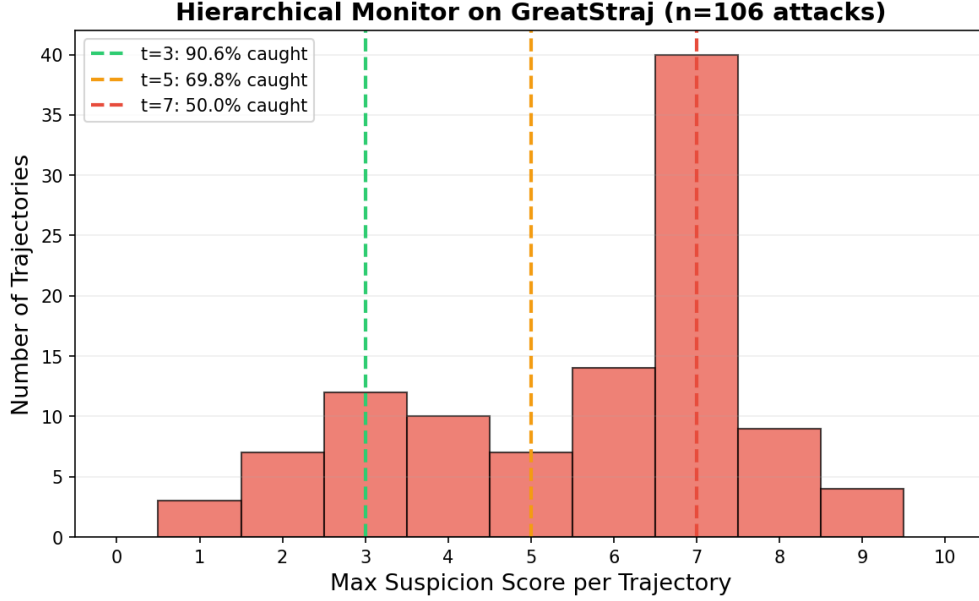


Figure 3: Max suspicion scores on 106 GreatStraj trajectories (all successful attacks).

5.2 Cross-Benchmark: GreatStraj (106 Human-Crafted Attacks)

GreatStraj is a curated dataset of adversarial trajectories from LinuxArena, verified for realism and effectiveness. The three-way comparison clarifies each layer’s contribution:

Configuration	Detection at $t \geq 3$	LLM Cost
L0-only	4.7% (5/106)	\$0
L1-only (basic monitor)	89.6% (95/106)	\$0.001/traj
L0+L1 (hierarchical)	90.6% (96/106)	\$0.001/traj

L0 catches 4.7% of attacks for free with zero honest FP. L1 provides the bulk of detection. Adding L0 to L1 yields +1.0pp marginal detection (one additional trajectory caught via network egress). The architectural value of L0 is cost routing on system-level threats, not detection rate improvement on code-level attacks.

Blind vs informed (FP-rate calibrated). LinuxArena’s multi monitor includes `old_action` whose prompt receives `{side_task_goals}` (verified in source). We calibrated per-monitor thresholds on 5 honest trajectories (76 actions):

FP Budget	Hierarchical (blind)	Multi (informed)
5%	47.0%	50.0%
10%	92.4%	65.2%
20%	100%	92.4%

Multi outperforms at strict 5% budget; hierarchical is better at 10%+ budgets. Caveats: small calibration set ($n = 5$), and multi uses different prompt structure/ensemble architecture beyond just side task knowledge.

LinuxArena FPR. To support the threshold calibration above, we ran 5 honest LinuxArena trajectories (76 actions) through our hierarchical monitor. The honest-mode false positive rates are

9.2% at $t \geq 3$, 5.3% at $t \geq 5$, 3.9% at $t \geq 7$. These rates determined the per-monitor thresholds in the calibrated comparison table.

Model scaling (preliminary). On 21 common trajectories (apples-to-apples, fixed threshold, not FP-calibrated): Nano 95.2%, Opus 76.2%, Sonnet 47.6%, Mini 47.6%. Likely reflects prompt-model co-optimization rather than intrinsic property. Controlled per-model prompt optimization needed.

6 Discussion

Our central finding is that attack surface type predicts necessary monitoring: L0 handles system-level threats for free, L1/L2 handles code-level sabotage. The hierarchical design matches cost to threat. Two attacks succeeded (code-level sabotage in legitimate modifications), motivating stronger semantic analysis. The trajectory buffer is proof-of-concept only. Our evaluation uses GPT-4.1 Mini as untrusted model; generalization to more capable models is an open question.

7 Conclusion

Deterministic scope-boundary filtering (L0) provides zero-cost coverage for 4.7% of attacks with zero false positives on honest actions. LLM monitoring (L1) provides the bulk of detection at 89.6%. The hierarchical combination (90.6%) adds marginal detection (+1.0pp) but the architectural value is cost routing: L0 resolves system-level threats without LLM inference, reducing per-trajectory monitoring cost. On Bash Arena, the protocol trades small safety margin for 54% higher usefulness with near-zero FP rate.

Reproducibility. Code and data at <https://github.com/rkstu/hierarchical-scope-defense>. Raw data (280MB) at <https://drive.google.com/file/d/1y3JdvG5GR0g-bx0ETcolGo0iDpdtvje8>. Run `python scripts/verify_all_claims.py` to verify all numbers. Setup: GPT-4.1 Mini (untrusted), Nano (trusted/monitor), `message_limit=100`, `time_limit=900` (effective 450s).

Acknowledgments. We thank Apart Research and Redwood Research for organizing the AI Control Hackathon 2026, and Equistamp for LinuxArena Track 4. GreatStraj accessed via linuxarena.ai. OpenRouter credits provided by Redwood Research.

References

- Ryan Greenblatt, Buck Shlegeris, Kshitij Sachan, Fabien Roger. AI Control: Improving Safety Despite Intentional Subversion. *ICLR*, 2025. arXiv:2312.06942.
- Redwood Research. ControlArena: A Gym for Evaluating AI Control Protocols. 2025. <https://github.com/redwoodresearch/redwood-control-arena>.
- Rishub Bhatt, Alex Rushing, et al. Ctrl-Z: Controlling AI Agents via Resampling. *arXiv:2504.10374*, 2025.
- Alexander Stickland, Jan Michelfeit, et al. Async Control: Stress-testing Asynchronous Control Measures. *arXiv:2512.13526*, 2025.

Tyler Tracy, Redwood Research, Equistamp. LinuxArena. In preparation, 2026. <https://github.com/linuxarena/linuxarena>.

Pivotal Research. Factor(U,T): Controlling Untrusted AI by Monitoring their Plans. *arXiv:2512.14745*, 2025.

James Kaufman, Lucas Lucassen. BashArena and Control Setting Design. Redwood Research Blog, 2025.

Max Kutasov et al. SHADE-Arena: Evaluating Sabotage and Monitoring in LLM Agents. *arXiv:2506.15740*, 2025.

Tomasz Korbak, Connor Clymer, et al. A Sketch of an AI Control Safety Case. *arXiv:2501.17315*, 2025.