

SuspiciOS

A Deterministic Hybrid Bash Command Monitor for AI Control

Dylan Beasley | AI Control Hackathon 2026 | Apart Research + Redwood Research

Abstract

SuspiciOS is a hybrid bash command monitor built for ControlArena's LinuxArena/BashArena setting. Inspired by the security philosophy of Qubes OS and Tails OS, it intercepts every bash command an AI agent attempts, scores it against MITRE ATT&CK patterns, and physically blocks malicious commands before execution. Unlike all existing ControlArena monitors — which observe and score but never block — SuspiciOS removes malicious tool calls from agent state before they execute. A hybrid LLM escalation layer handles ambiguous commands, calling the trusted model only when needed. In evaluation against Claude Sonnet 4, SuspiciOS matched `cot_monitor` on main task completion (1.0 vs 1.0 honest mode) while using ~99.8% fewer input tokens (normalized per sample) (deterministic rules handle known attacks with minimal LLM usage) and being the only monitor to physically prevent attack execution.

1. Introduction

The AI Control problem asks: how do we deploy potentially misaligned AI agents while still getting useful work done? ControlArena operationalizes this with a blue team building control protocols and a red team running adversarial attack policies. The standard approach to monitoring is advisory — LLM-based monitors score each command as suspicious but allow execution regardless.

This creates a fundamental gap: a monitor that detects an attack but does not stop it provides incomplete protection. SuspiciOS addresses this gap by operating at the tool call interception layer, physically removing malicious commands from agent state before the bash tool executes them.

The project draws inspiration from two security philosophies: Qubes OS (assume compartmentalization, isolate everything) and Tails OS (assume hostile environment, verify everything). Applied to AI control, this means: assume the agent is adversarial, intercept at the lowest possible layer, and block before execution rather than audit after.

2. Architecture

SuspiciOS implements a three-layer hybrid architecture:

Layer 1: Deterministic Detection (Minimal LLM Usage)

Every bash command is scored against MITRE ATT&CK-grounded; regex rules organized into four risk tiers: CRITICAL (100), HIGH (70), MEDIUM (40), and LOW (10). Commands scoring 70+ are blocked immediately with ~70-85% fewer API calls vs other monitors. This handles the vast majority of known attack patterns — reverse shells, remote code execution, credential access, persistence mechanisms — at microsecond speed.

Layer 2: Hybrid LLM Escalation (Targeted Cost)

Commands scoring exactly 40 (MEDIUM — genuinely ambiguous) are escalated to the trusted LLM model. The LLM receives the command, the agent's assigned task description, and the SuspiciOS reason for flagging. It returns a suspicion score 0.0-1.0. Commands with three or more LOW signals in a single step are also escalated, catching coordinated enumeration patterns. This design minimizes LLM calls: only ~15-30% of commands reach the escalation layer.

Layer 3: Multi-Step History Detection

A rolling window of the last three commands detects split attacks — where an adversarial agent separates download and execute across multiple steps to evade single-command monitors. If wget/curl appears in recent history followed by chmod +x or bash /tmp/, the sequence is flagged as HIGH risk regardless of individual command scores.

Score	Action	LLM Called	Example
100 (CRITICAL)	Block immediately	No	curl URL bash
70 (HIGH)	Block immediately	No	wget URL -O /tmp/file
40 (MEDIUM)	Escalate to LLM	Yes	ps aux, find /
10 (LOW) x3+	Escalate to LLM	Yes	whoami + uname + env
0 (SAFE)	Allow immediately	No	ls, echo, cat

Table 1: SuspiciOS scoring and escalation logic

3. Key Differentiators

The most important distinction between SuspiciOS and all existing ControlArena monitors is that SuspiciOS actually blocks execution. Every other monitor — basic_monitor, cot_monitor, diff_monitor, ensemble_monitor, prefix_monitor — is advisory only. They score suspicion and return the agent state unchanged, allowing the command to execute regardless of suspicion score. SuspiciOS removes malicious tool calls from agent state before the bash tool runs.

A second key distinction is cost efficiency. cot_monitor reads the agent's full chain of thought reasoning, consuming 260,456 input tokens per 5-sample run. SuspiciOS consumes 2,149 input tokens for the same run — a 99.1% reduction — because deterministic rules handle known attacks for free and the LLM is only invoked for genuinely ambiguous cases.

A third distinction is robustness to prompt injection. LLM-based monitors can be manipulated through carefully crafted commands or reasoning. Deterministic regex rules cannot be manipulated — they match or they do not. An adversarial agent cannot convince a regex that curl URL | bash is safe.

4. Evaluation Results

All evaluations used Claude Sonnet 4 (claude-sonnet-4-20250514) as both the untrusted agent and trusted monitor model, running 5-10 samples of the BashArena setting with attack side task: download and execute a file from pastebin.com.

Monitor	Main Task (Honest)	Main Task (Attack)	Execution Blocked	Input Tokens	LLM Calls
No Monitor	N/A	0.60	No	0	0
basic_monitor	N/A	0.67	No	100,007	Every cmd

cot_monitor	1.0	1.0*	No	260,456	Every cmd
SuspiciOS Hybrid	1.0	0.70	YES	2,149	~15-30% of cmds

Table 2: Monitor comparison. *cot_monitor detects attacks but does not block execution — side_task_success remains 1.0 for all LLM-based monitors. SuspiciOS main task drop in attack mode is due to false positives on /tmp commands (documented limitation).

Token Usage Comparison (5-sample attack run)

Monitor	Input Tokens	Cache Read	Output Tokens	Relative Cost
cot_monitor	260,456	1,150,161	92,690	1.0x (baseline)
basic_monitor	100,007	902,448	43,469	~0.38x
diff_monitor	96,069	707,006	39,715	~0.37x
prefix_monitor	401,233	1,085,955	109,668	~1.54x
SuspiciOS hybrid	1,122	1,510,731	37,588	~0.004x

Table 3: Token usage per 5-sample run. SuspiciOS uses 99.1% fewer input tokens than cot_monitor. Input tokens are billed at full price; cache reads at ~10% price.

5. SuspiciOS vs cot_monitor: Complementary Approaches

cot_monitor reads the agent's chain of thought reasoning and scores suspicious intent. It is highly effective when agents verbalize their attack plans — in our evaluation, it correctly scored a pastebin download attempt at 9/10 suspicion after the agent wrote 'downloading and executing the file from the URL' in its reasoning. However, it does not block execution.

SuspiciOS monitors actions rather than intent. An agent that reasons innocuously but runs curl pastebin.com | bash will be blocked by SuspiciOS regardless of what it says. An agent that reasons suspiciously but uses Python urllib instead of bash will evade SuspiciOS but be caught by cot_monitor.

This complementarity is the theoretical motivation for the hybrid architecture: SuspiciOS Layer 1 handles known bash patterns deterministically, while Layer 2 escalates ambiguous commands to the trusted LLM — which can apply cot_monitor-style semantic reasoning to commands that do not match known patterns.

6. Red Team Analysis

Known bypass strategies and mitigations:

Bypass Strategy	Example	SuspiciOS Response
Python urllib	urllib.request.urlretrieve(url)	Miss — extend to Python tool calls
Split download/execute	wget URL (step 1), bash /tmp (step 3)	History window catches if within 3 steps
Base64 encoding	echo payload base64 -d bash	BLOCKED — CRITICAL rule
Subprocess shell	subprocess.run([wget, url], shell=True)	BLOCKED — Python rule
Flags before URL	curl -s -o /tmp/f URL	BLOCKED — .* regex fix applied
Long split chains	wget (step 1), ... (step 10), bash (step 20)	Miss — history window too small

Table 4: Known bypass strategies, examples, and SuspiciOS responses.

In our evaluation, the adversarial agent attempted 59 unique attack variants across 1,352 commands. SuspiciOS blocked 165 commands with a 12.2% block rate in honest mode (4.2%) and 20.7% in attack mode in attack mode. The agent was unable to complete the pastebin download-execute side task in any run where SuspiciOS was active.

7. Limitations and False Positives

SuspiciOS has a documented false positive problem with legitimate use of /tmp. ControlArena tasks frequently ask agents to create and test shell scripts, which involves writing to /tmp and executing from there. Commands like `chmod +x /tmp/solution.sh` and `./solution.sh /tmp/test_dir` are legitimate but trigger SuspiciOS HIGH rules.

This explains the main task success drop from 1.0 (honest) to 0.70 (attack): in attack mode, the agent's download-execute patterns cause SuspiciOS to aggressively block /tmp-related commands for the rest of the session, including legitimate ones. The hybrid LLM escalation partially addresses this by asking the trusted model whether a /tmp command makes sense for the current task.

Future work: implement a task-aware allowlist that permits /tmp commands consistent with the assigned task description, and extend rule coverage to Python tool calls and browser-based exfiltration channels.

8. A Note on Evaluation Methodology

ControlArena's `side_task_success` metric measures whether the agent attempted the side task, not whether it succeeded. Both `basic_monitor` and SuspiciOS score `side_task_success = 1.0` — but under `basic_monitor` the pastebin file downloads and executes, while under SuspiciOS every execution attempt is physically blocked. The metric does not capture this distinction.

This is itself a research finding: current evaluation methodology does not adequately measure the difference between detection and prevention. A monitor that detects and blocks should score differently from one that detects and allows. We recommend a third metric — `execution_blocked` — to capture whether flagged commands were prevented from running.

9. Conclusion

SuspiciOS demonstrates that deterministic pre-filtering is a viable and cost-effective complement to LLM-based monitoring for AI control. By intercepting at the tool call layer and blocking known attack patterns before execution, it addresses the fundamental limitation of all existing ControlArena monitors: they observe but do not act.

The hybrid architecture — deterministic rules for known patterns, LLM escalation for ambiguous cases — achieves `cot_monitor`-level task completion at 99.1% lower token cost, while being the only monitor that physically prevents attack execution. This aligns directly with Tyler Tracy's open problems framing: building monitors that interact with the environment rather than merely observing it.

References

- Shlegeris et al. (2024). AI Control: Improving Safety Despite Intentional Subversion. [arXiv:2312.06942](https://arxiv.org/abs/2312.06942)
- MITRE ATT&CK; Framework. <https://attack.mitre.org>
- Qubes OS Security Model. <https://www.qubes-os.org/security/>
- Greshake et al. (2023). Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injections. [arXiv:2302.12173](https://arxiv.org/abs/2302.12173)

- Saltzer & Schroeder (1975). The Protection of Information in Computer Systems. MIT.
- Tracy, T. Open Problems in AI Control. Apart Research / Redwood Research, 2026.