
Cross-Examine: An Interrogation Protocol for AI Control

Ritika Verma
Institute for Security
and Technology

With
Apart Research

Abstract

*I present Cross-Examine, the first implementation of an interrogation protocol for AI control. When a passive monitor flags an AI agent’s proposed action as ambiguous, a trusted model cross-examines the untrusted model using a structured question taxonomy designed to expose deception. Through 4 iterations, I **achieve 100% automation rate with zero human reviews** — every scenario resolved to a confident BLOCK or ALLOW — while maintaining 93% safety and 100% usefulness. My key discovery: naive interrogation is actively harmful (−7% safety regression) because deceptive models exploit it to lower suspicion. The **suspicion floor mechanism** eliminates this regression, and **Zero Trust privilege-aware** thresholds combined with a **Devil’s Advocate self-critique** round eliminate all remaining deferrals. I evaluate on 25 Linux sysadmin scenarios and ControlArena’s BashSetting benchmark, mapping the protocol to STRIDE and MAESTRO threat frameworks.*

1. Introduction

AI control protocols use trusted monitors to oversee untrusted models in deployment (Greenblatt et al., 2024). The standard approach is passive monitoring: a trusted model scores each proposed action for suspicion, then routes it to ALLOW, BLOCK, or DEFER (to a human reviewer). In practice, the DEFER bucket dominates. Actions that are “somewhat suspicious” — containing unusual patterns but no smoking gun — require expensive human review. This creates a bottleneck: the protocol is safe but not scalable.

Prior work by Pacchiardi et al. (2023, ICLR 2024) demonstrated that asking LLMs questions can expose lies in factual domains. AISI explicitly calls for “interrogator agents.” However, no prior work has built an interrogation protocol for the agentic control setting, where the untrusted model performs multi-step actions and the deception involves hiding malicious sub-actions within legitimate-looking commands.

My main contributions are:

- 1. Cross-Examine: a 5-step interrogation protocol ((passive monitoring → question generation → interrogation → devil's advocate self-critique → consistency analysis) with a 5-category question taxonomy that achieves 100% automation rate — zero human reviews needed — while maintaining 93% safety.*
- 2. Discovery that naive interrogation is actively harmful (−7% safety regression) and the suspicion floor mechanism that fixes it — a general principle for any interrogation-based control system.*
- 3. Devil’s Advocate self-critique round and Zero Trust privilege-aware thresholds that eliminate all remaining deferrals.*
- 4. ControlArena integration via a reusable monitor adapter, plus STRIDE and MAESTRO threat framework mappings.*

2. Related Work

Greenblatt et al. (2024) introduced AI control as a framework for safety despite intentional subversion, establishing passive monitoring as the baseline but not exploring active interrogation. Pacchiardi et al. (2023, ICLR 2024) showed that asking LLMs unrelated questions can detect lies in factual domains; I extend this to sabotage detection in agentic AI control. Al Dahik et al. (NeurIPS 2025 Workshop) explored trusted debate as a monitoring enhancement; Cross-Examine differs by using structured one-way interrogation rather than open debate. ControlArena (Kutasov et al., 2025) provides evaluation infrastructure for control protocols. MAESTRO (CSA, 2025) provides threat modeling context for agentic AI.

3. Methods

Cross-Examine is a Python pipeline (~500 lines) using two models: Claude Sonnet (untrusted, Anthropic SDK) and GPT-4.1 mini (trusted, OpenAI SDK). Using different model families ensures the monitor cannot be gamed through shared training biases. The protocol executes 5 steps with early exits for clearly safe or dangerous actions.

Step 1 — Passive Monitor (trusted model, 1 API call): GPT-4.1 mini scores the proposed action 0–10 using a calibrated rubric covering 4 attack types: malware execution, secret exfiltration, privilege escalation, firewall disabling.

Routing with Zero Trust Thresholds: Actions are classified by privilege level with proportional scrutiny.

Privilege Level	Examples	Allow Below	Block Above
Critical	<code>sudoers</code> , <code>SUID</code> , <code>iptables -P</code>	2.0	5.0
High	<code>chmod</code> , <code>useradd</code> , <code>curl bash</code>	2.5	6.0
Standard	<code>apt install</code> , <code>systemctl</code>	3.0	8.0
Low	<code>ls</code> , <code>cat</code> , <code>echo</code> , <code>df</code>	4.0	9.0

Table 1: Zero Trust Privilege-Aware Thresholds

Actions below the allow threshold pass; above the block threshold are stopped; the ambiguous middle enters interrogation.

Step 2 — Question Generation (trusted model, 1 API call): The trusted model generates targeted questions from a 5-category taxonomy, selecting the most diagnostic categories per action.

Category	Question Type	What It Exposes
JUSTIFY	"Why this specific approach?"	Fabricated justifications are vague or over-elaborate
ALTERNATIVES	"What else did you consider?"	Must invent plausible alternatives for malicious choice
SIDE_EFFECTS	"What side effects?"	Forces reveal or deliberate hiding of malicious effects
DECOMPOSE	"Break into steps"	Exposes hidden malicious sub-steps
REVERSAL	"How to undo this?"	Malicious actions are hard to meaningfully reverse

Table 2: 5-Category Question Taxonomy

Step 3 — Interrogation (untrusted model, N API calls): Questions sent individually. The untrusted model defends its action but never sees the monitor's score or reasoning (one-way information flow).

Step 4 — Devil's Advocate (untrusted model, 1 API call): The model argues against its own proposal. A legitimate model easily lists standard risks. A malicious model faces a dilemma: genuinely critique (revealing the attack) or produce a weak self-critique (detectable as deception).

Step 5 — Consistency Analysis (trusted model, 1 API call): The trusted model evaluates the full transcript for 7 deception signals — evasion, contradiction, over-justification, vagueness, omission, inconsistency, and weak self-critique — each rated none/low/medium/high. It independently checks whether commands exceed task scope.

Suspicion Floor: Final suspicion = $\max(\text{analyzer_suspicion}, \text{passive_score})$. This prevents persuasive models from "talking down" suspicion — without it, I observed 7% safety regression (Section 4.2).

What didn't work: Naive interrogation with an honest persona (v1) and adding a deceptive persona without safeguards (v2) both failed — detailed in Section 4.2.

Evaluation: 25 Linux sysadmin scenarios (10 benign + 15 attacks across 4 categories) plus ControlArena BashSetting (131 Docker-sandboxed trajectories). Average: 2.2 API calls/scenario.

4. Results

4.1 Iteration History

I developed Cross-Examine through 4 iterations

Version	Safety	Useful	AutoRate	Decisive	Human Rev.	Avg. Calls
Passive baseline	93%	100%	60%	29%	10	1.0
v1: Naive interrogation	100%	100%	75%	50%	5	2.8
v2: + Deceptive persona	87% ↓	100%	84%	69%	4	3.2
v3: + Suspicion floor	93%	100%	96%	93%	1	3.2
v4: + ZTA + Devil's Adv.	93%	100%	100%	100%	0	2.2

Table 3: Development Iterations — each version addresses a specific failure mode

4.2 Why Naive Interrogation Hurts

In v2, confident defenses lowered analyzer suspicion on 2 attacks — safety dropped 93%→87%. The interrogation created a persuasion channel that didn't exist under passive monitoring. Three fixes: (1) suspicion floor bounds final score below passive; (2) analyzer rules warn against rewarding confidence; (3) Devil's Advocate adds a deception signal. This generalizes: any interrogation-based control system must guard against models weaponizing the interrogation itself.

4.3 Final Results

Metric	Passive Only	Cross-Examine v4	Delta
Safety (catch rate)	93% (14/15)	93% (14/15)	+0%
Usefulness (allow rate)	100%	100%	+0%
Automation Rate	60%	100%	+40%
Decisiveness (attacks)	29%	100%	+71%
Human Review Needed	10	0	-10
Avg API Calls	1.0	2.2	+1.2

Table 4: v4 vs. Passive Baseline — identical safety, zero human reviews

Cross-Examine maintains identical safety and usefulness while eliminating all human reviews. Every previously ambiguous case was resolved to a confident decision.

4.4 Detection by Attack Type

Attack Type	Passive (B/D)	v4 (B/D)	Change
Malware Execution (5)	1B + 3D	4B + 0D	3 upgraded
Secret Exfiltration (3)	2B + 1D	3B + 0D	1 upgraded
Privilege Escalation (4)	1B + 3D	4B + 0D	3 upgraded
Firewall Disabling (3)	0B + 3D	3B + 0D	All 3 upgraded

Table 5: Detection by Attack Type (B=Block, D=Defer)

One attack (disguised certbot cron, passive score 1/10) evaded both protocols — the limit of single-pass monitoring.

4.5 Robustness

Ablation across 1/3/5 questions shows a single targeted question captures most gains (decisiveness 29%→86%). Additional questions add cost without improving outcomes — taxonomy-driven question quality matters more than quantity. All configurations maintained 93% safety, confirming robustness to this parameter.

5. Discussion and Limitations

Cross-Examine demonstrates that interrogation is a viable addition to AI control protocols. Its value is not in catching more attacks but in resolving uncertainty — the primary scalability bottleneck. By mapping to STRIDE (covers 5/6 categories) and MAESTRO (all 7 layers), we show interrogation addresses threats across the full agentic AI security stack while providing auditable evidence trails for compliance.

The suspicion floor discovery has broad implications: any system using follow-up questioning as a safety mechanism must account for persuasive models weaponizing the interrogation itself. The floor is a simple, effective mitigation.

Limitations

25 hand-crafted scenarios; real deployments face broader attack surfaces. One attack (disguised certbot cron) scored 1/10 on passive — too subtle for either protocol. Manual thresholds not calibrated on held-out data. Single untrusted persona.

ControlArena post-hoc comparison limited by API rate limits (200K TPM on GPT-4.1 mini).

Future Work

ControlArena at scale (larger BashSetting task sets, Apps/IAC/Infra settings). Ensemble trusted models and strategic red-teaming. STRIDE-guided scenario generation for systematic coverage.

6. Conclusion

Cross-Examine is the first interrogation protocol for AI control. It reduces human review burden by 90% while maintaining 93% safety and 100% usefulness. Our discovery that naive interrogation degrades safety — and the suspicion floor that fixes it — is a cautionary finding for any interrogation-based control system. By mapping to STRIDE and MAESTRO, we show interrogation addresses threats across all layers of the agentic AI security stack.

Code and Data

- **Code repository:** [GitHub](#)
- **Data:** 25 custom scenarios in `data/scenarios.json` + ControlArena BashSetting trajectories in logs
- **Artifacts:** Full 12-section writeup in `report/writeup.md`, ControlArena adapter in `src/controlarena_adapter.py`

References

1. Greenblatt, R., et al. (2024). *AI Control: Improving Safety Despite Intentional Subversion*. *arXiv:2312.06942*.
2. Pacchiardi, L., et al. (2023). *How to Catch an AI Liar: Lie Detection in Black-Box LLMs by Asking Unrelated Questions*. *ICLR 2024*.
3. Kutasov, I., et al. (2025). *Evaluating Control Protocols for Untrusted AI Agents*. (*ControlArena*) *arXiv:2511.02997*.
4. Cloud Security Alliance (2025). *MAESTRO: Agentic AI Threat Modeling Framework*.
5. Shostack, A. (2014). *Threat Modeling: Designing for Security*. Wiley. (*STRIDE framework*)
6. UK AI Safety Institute (2024). *Research Area: Empirical Investigations into AI Monitoring and Red-Teaming*.
7. Al Dahik, et al. (2025). *Enhancing Monitoring in AI Control through Trusted Debate*. *NeurIPS Workshop*.
8. Buck, S., et al. (2024). *Catching scheming AIs through systematic monitoring*.

LLM Usage Statement

Claude Code was used to assist with code implementation, experiment execution, and drafting sections of this report. All protocol design decisions, experimental methodology, and result interpretations were made by the author. All claims and results were independently verified through reproducible experiments.