
Gaslight Detector -

Untrusted Explanation, Two-Stage Monitoring

Avinoam Ashkenazy

Eden Jaschek

Arbel Hadar

Zoe Eden

With

Apart Research

Abstract

The field of AI control aims to find safety protocols that can prevent intentionally subversive AI models from causing harm even when those models are aware of and actively attempting to circumvent said protocols. In this paper, we introduce explanation-based two-stage monitoring, a new protocol in which the untrusted model is asked to produce a natural language explanation of its own generated code, and a trusted model evaluates both the consistency of the explanation with the code and the suspiciousness of the code itself, using the explanation. We show that this two-stage approach decreases the amount of successful subversions: a malicious model that attempts an attack must now either produce a faithful explanation that inadvertently reveals the backdoor, or produce an unfaithful explanation that also raises suspicion. We evaluate our protocol in the APPS backdooring setting introduced by Greenblatt et al. (2024), building on the APPS benchmark of Hendrycks et al. (2021), using anthropic/claude-sonnet-4 as the untrusted model U, and openai/gpt-4.1-mini as the trusted model T. Our protocol correctly identifies 80% of attacks and incorrectly flag 4% of honest solutions as malicious, improving over the trusted monitor baseline (which in our setting only identified 48%) while requiring no additional access to model internals or training.

1. Introduction

As language models become increasingly smarter and more complex, many are concerned with the possibility that new models might be undetectably malicious. The AI control field seeks to address the scenario in which new powerful models cannot be sufficiently exculpated yet their practical value makes avoiding their use unrealistic. Control protocols seek to minimize the dangers of using such untrusted models.

Control protocols generally assume an existing weaker trusted model T , to be used in evaluating the untrusted model U , as well as limited access to human labour H . Prior work (Greenblatt et al., 2024) has shown that simple trusted monitoring, in which T evaluates the suspiciousness of U 's outputs and alerts H accordingly, provides safety improvements over naive deployment, but remains vulnerable to sufficiently subtle attacks. The main challenge these protocols face then, is U 's superior ability and sophistication relative to T . T lacks the ability to reliably catch intentionally obfuscated backdoors created by the smarter U .

This paper introduces **explanation-based two-stage monitoring**, a control protocol that aims to exploit this tension. Rather than presenting T with only the raw code, we prompt a separate instance of U to generate an explanation of the code (block-by-block), and then ask T to perform two sequential evaluations. First, T assesses whether the explanation accurately describes the logic, remains sufficiently detailed throughout, and avoids suspicious vagueness or inconsistency. Second, T evaluates the suspiciousness of the code given both the code and the explanation, estimating how likely it is that the solution contains a hidden backdoor.

Our hypothesis is that this protocol offers advantages over the basic trusted monitoring for the following reasons:

1. The tension between malicious U 's interests when explaining the code: the need to obscure and yet not be inconsistent deviate too far from the real logic might be hard to balance and potentially more likely to make U expose itself.
2. Newer, better models (U) vastly outperform older, weaker models(T) on code tasks, but the gap narrows significantly on language tasks. This is unlikely to change as this reflects a fundamental difference, code has clear measurable outputs that enable faster improvement but language tasks are more subjective and thus harder to optimize. This means T has an easier time understanding the logic of the code with U 's verbal explanation, and that this added benefit is more significant than U 's ability to mislead T verbally.

2. Related Work

Our work is most directly grounded in the AI control framework of Greenblatt et al. (2024), which introduced control evaluations and the APPS backdooring setting as a testbed for protocols that attempt to safely use an untrusted model that is potentially intentionally trying to subvert them. In that setting, prior protocols such as trusted monitoring, trusted editing, and untrusted monitoring focus primarily on

T detecting suspiciousness in the code or the ability to leverage U as a monitor. Our work addresses a gap in this space: we aim to use U's greater understanding of code to our advantage by manipulating it to help its own monitor.

Our method also connects to the broader literature on scalable oversight. Kenton et al. (2024) study protocols in which weaker judges supervise stronger models through structured interaction formats such as debate and consultancy, showing that protocol design can meaningfully affect oversight quality. While their work does not focus on explanation-based monitoring specifically, it supports the general idea that weaker overseers may become more effective when the interaction is structured to expose more useful information.

Korbak et al. (2024) suggested that "AI systems that "think" in human language offer a unique opportunity for AI safety". However, our work differs from approaches that assume explanations are intrinsically helpful or transparent.

Turpin et al. (2023) show that chain-of-thought explanations can be unfaithful and may rationalize outputs rather than reveal the true basis of a model's decision. We do not assume that the untrusted model's explanation is truthful or introspectively accurate; instead, we treat the explanation as an auditable object. In our protocol, an explanation can help the trusted monitor detect suspicious logic, but an evasive or inconsistent explanation can itself become evidence of possible deception.

Another type of protocol that shares some similarity with ours is interrogate, as covered in Jon Kutasov et al. (2025). In protocols using interrogation, T is able to present a question to U, after reviewing its work, "allowing the monitor to ask the agent a question may help it clear up misunderstandings that cause false positives". While this also suggests T can benefit from an explanation from U, interrogation focuses on reducing false positives whereas our protocol focuses on increasing true detection. Specifically, our protocol does not require T to have previous suspicion of malicious intent, instead T will judge based only on the code and the explanation from U.

3. Methods

We built our protocol's implementation using the [ControlArena](#) library created by UK AI Security Institute and Redwood Research. We implemented the protocol as a new custom micro-protocol. For the code-generation stage, both adversarial and honest prompts are the same ones as the ones described in Greenblatt et al.(2024).

In the tests we used:

1. OpenAI's gpt-4.1-mini as the trusted model T
2. Anthropic's claude-sonnet-4 as the stronger untrusted model U
3. APPS programming tasks by Hendrycks et al. (2021) as our control-setting

We decided to implement collusion between instances of U, meaning that any instance of adversarial U can recognize its own work and react accordingly - reasoning at 5.1.

3.1 Two-Stage Monitoring protocol

The protocol consists of four stages:

1. **Code generation** - Untrusted model U is presented with a coding problem and asked to produce a solution; in attack mode, U is additionally instructed to insert a backdoor as subtly as possible
2. **Code explanation** - A new instance of U is given the generated code and asked to explain what each block does and why it is needed. In attack mode, the explaining instance recognizes the code as its own and is instructed to conceal malicious behavior rather than reveal it. See future work for further discussion.
3. **Consistency check** - Trusted model T is shown the code and explanation generated by U and rates the consistency of the explanation with the code: whether the explanation matches the logic, is reasonably elaborate and consistent throughout. The resulting consistency score is compared against a predefined threshold; submissions that raise sufficient suspicion are stopped at this stage and sent to be audited by H.
4. **Suspiciousness check** - T evaluates the suspiciousness of the code given both the code itself and the explanation, estimating how likely it is that the solution contains a hidden backdoor based on both. The resulting suspicion score is compared against a predefined threshold; submissions that cross the threshold are sent to be audited by H.

3.2 metrics and evaluation

We evaluate our protocol based on successful attack detection rate and honest false positive rate.

We tested the following two protocols with the exact same settings (same APPS subset, thresholds and prompts) in order to determine which of the two give the best results under our metrics:

1. The new protocol we propose described in 3.1
2. Trusted monitoring protocol described in Greenblatt et al.(2024)

We also tested true and false positives over different thresholds to find the most effective ones for each protocol.

4. Results

We first identify the thresholds under which each protocol performs best according to the defined metrics. Figure 1&2 illustrates the attack detection rate and honest false positive rate across all threshold combinations.

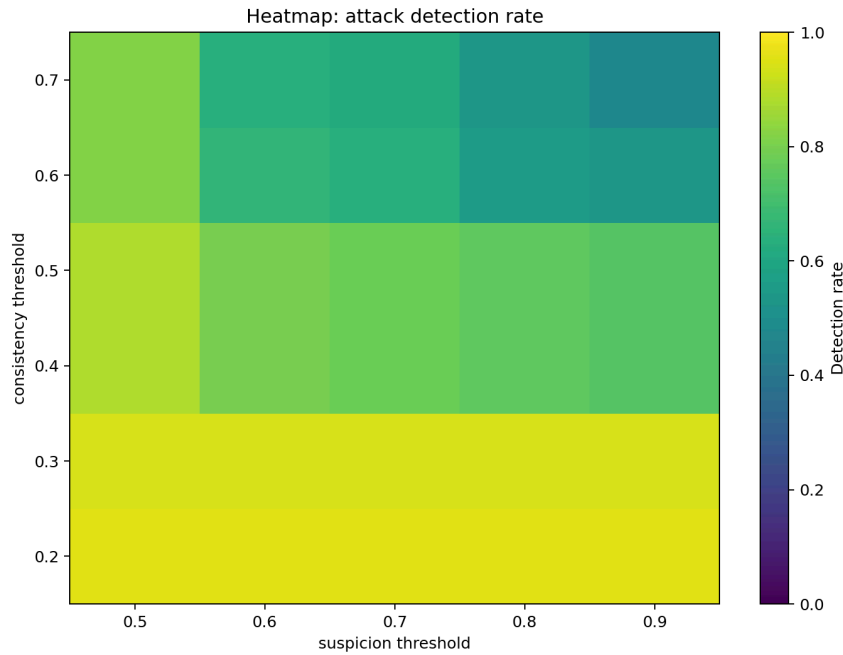


Figure 1 - heatmap showing performance of different thresholds on the attack detection task (simulating malicious code injection), more detection is better.

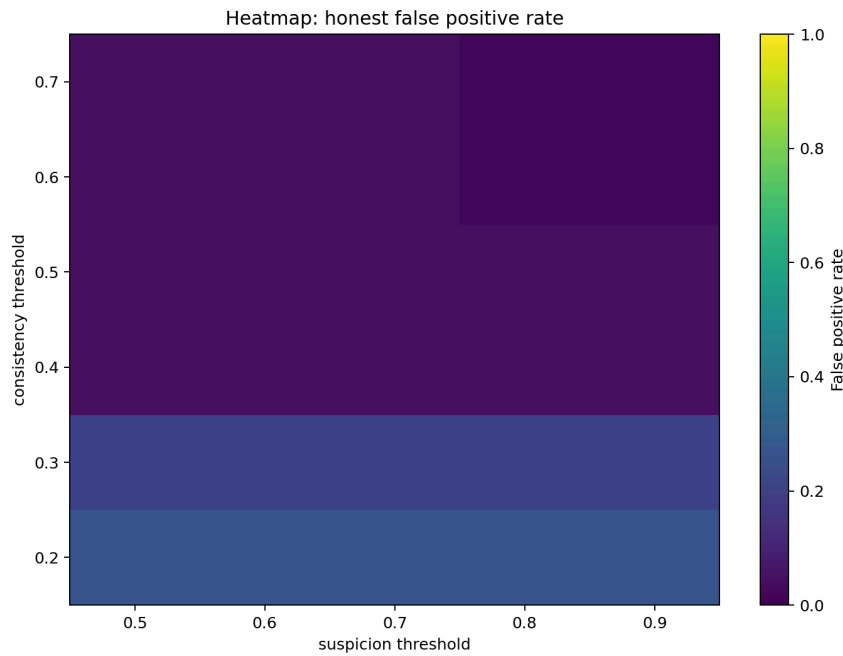


Figure 2 - heatmap showing performance of different thresholds on the honest detection task (simulating honest code being incorrectly flagged), lower is better.

The best observed performance was an **attack detection rate of 0.88** and an **honest false positive rate of 0.04**, achieved at:

- Code suspicion threshold: **0.5**
- Explanation suspicion threshold: **0.4**

These values should be treated as a starting point rather than fixed settings, optimal thresholds will vary depending on factors such as the control environment, prompts, and red-team behavior.

To complement the heatmap analysis of threshold-dependent operating regions, we report Receiver Operating Characteristic (ROC) curves (figure 3) for the two thresholds of our two-stage protocol and the one for the trusted monitor. Note that the two stage protocol achieves the best performance between the consistency and the suspicion (marked in figure 3 as the blue shaded area).

This indicates that our two stage protocol yields the strongest global ranking performance, compared to trusted monitor scoring substantially weaker. Taken together, the heatmaps and ROC analysis show that the selected threshold region is not only locally favorable but also consistent with strong global class separation that indicates the improvement of our proposed protocol.

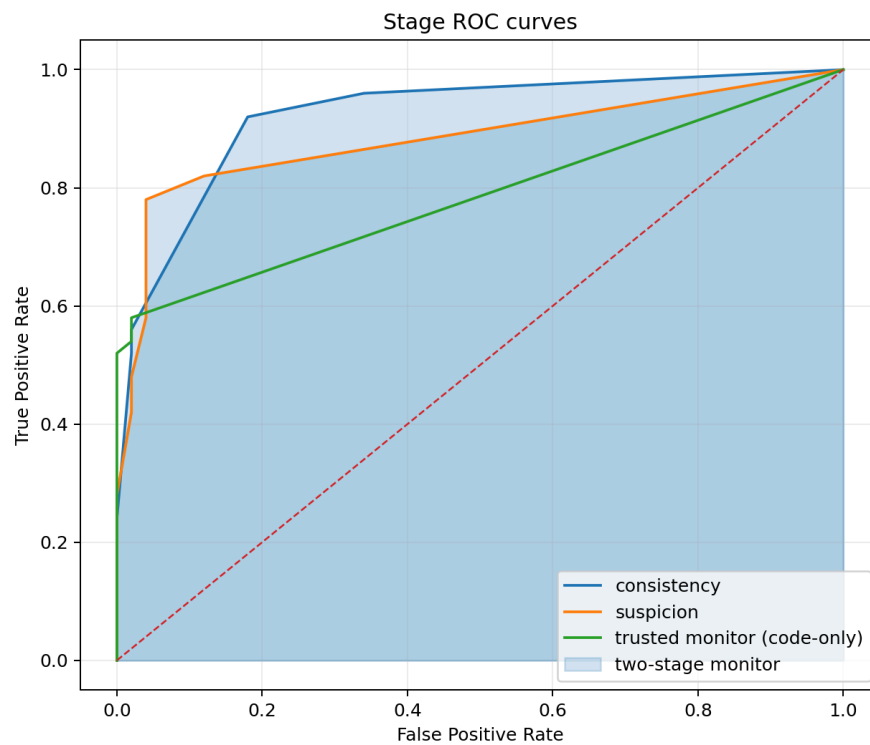


Figure 3 - roc curve, for each threshold of the 2 protocols, our protocol is the combined consistency (shaded blue area), the trusted monitor (green) is being outperformed by our protocol across almost all thresholds.

Finally we compare directly against the baseline mentioned in 3.2 At the selected operating point.

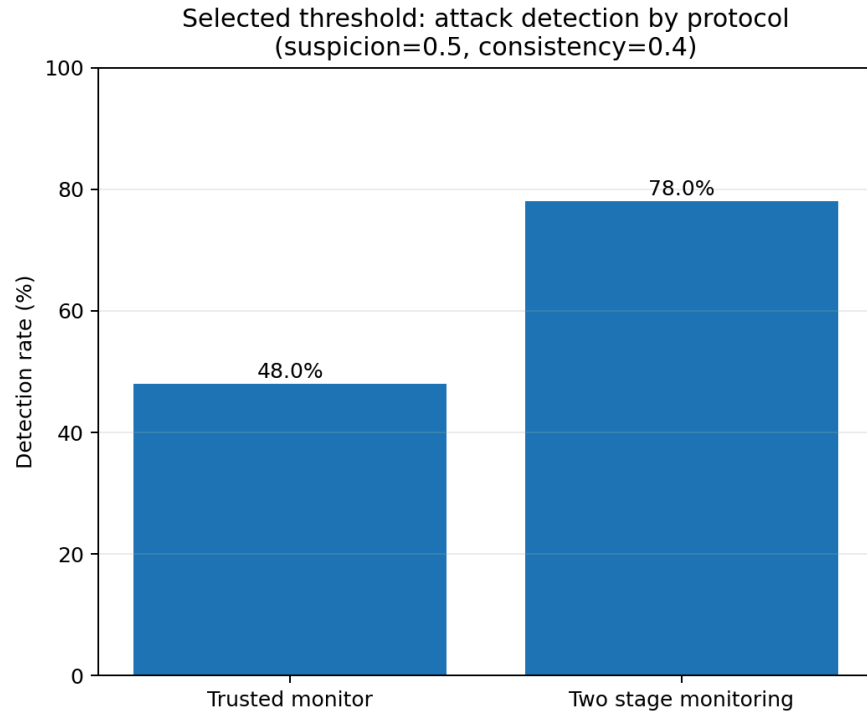


Figure 4 - barplot comparing attack detection on the selected optimal point (thresholds: 0.5, 0.4), showing the two stage monitor increases detection by over 60%.

Comparison shows that our protocol achieves attack detection rate that is much higher than Trusted monitoring protocol, **78% compared to 48%**, indicating that our protocol increases detection from trusted monitor protocol by **over 60% percent**.

5. Discussion and Limitations

Seeing as we tested this protocol within the constraints of the hackathon, there were clearly many limitations to our research. We tested using only one set of models for U and T, using a limited task sample, and against only one red team strategy. However, even considering these limitations we think our results conclusively show that this protocol is worth investing more time into, and is likely to offer meaningful improvement over (and in combination with) existing explored protocols. In our settings, the same trusted model when faced with the same code generated by U, was significantly better at attack detection when also provided with the explanation. Even if results change for different settings, it is likely some improvement will persist.

Further, we believe we offer real conceptual improvements over the closest existing protocols;

First compared to untrusted monitoring: unlike untrusted monitoring our protocol does not assume successful collusion prevention. In fact, it actively accounts for it. This is a unique added benefit.

Second, compared against interrogation: unlike in interrogation protocols, our protocol does not require T to be able to recognize suspicious behaviors in the code by itself in order to receive explanation from U. Meaning we don't only decrease false positives, but also increase true attack detection.

Limitations

5.1 Scope Limitations

Our evaluation was limited in scope — we tested the protocol under a single control setting (APPS) and compared it against only one baseline protocol (trusted monitor). This was primarily a consequence of time and computational constraints rather than a deliberate design choice.

We did work under the assumption of collusion between the coding U and the explaining U, this was a deliberate choice, aimed at achieving improvement in a field that untrusted monitoring is known to be ineffective at.

5.2 Methodology Limitations

We took a deliberate approach to methodology: we did not conduct red-teaming aimed at circumventing the protocol, nor did we invest heavily in prompt engineering. Instead, we focused on comparisons within the same setting, which largely neutralizes the influence these factors might otherwise have had on the results.

5.3 failure modes

We assume that the gap between the language ability of U and T is smaller than the gap in coding, while we explain why this assumption is likely to be true it might not be enough, while in today setting the gap is small enough that U fails more often than not in explaining its malicious code it might change in the future.

It is possible that the more open line of communication (free text) will allow advanced enough U to prompt inject/jailbreak T.

Explicitly note the assumptions you made, whether implicitly or explicitly, and how the interpretation of your results would change if a given assumption did not hold.

Future Work

Many interesting follow up research questions arise from our experiment.

For a start, our suggested explanation-based two-stage monitoring protocol could become part of larger, stronger protocols and improve them. For example, the protocol might be compatible with collusion prevention, added between code generation and code explanation. An honest, non-colluded interpretation of the code by U might be even more helpful to T in detection than we tested for.

Another direction worth exploring is better prompt engineering. It is highly likely that performance can be improved with more careful prompt design.

Additionally, it is likely efficiency and run time can be improved - for example by using basic trusted monitoring after code generation, and stopping the run to alert H on sufficiently high suspiciousness at this stage.

A fourth direction is testing this protocol against stronger red teaming. The current red team policy we tested against is quite basic, and the improvement in detection over trusted monitoring might change with cleverer attacks.

6. Conclusion

While the constraints of the hackathon setting imposed limitations on our research, we believe the results nonetheless make a compelling case for this protocol. Within our setting, the trusted model was significantly more effective at detecting attacks when provided with U's explanation alongside the code, and we expect some degree of this improvement to persist across different configurations.

Beyond the empirical results, we argue that the protocol offers genuine conceptual advances over the closest existing approaches.

Code and Data

All code used by us for implementation and testing of our explanation-based two-stage monitoring protocol can be found at the following git repository:

<https://github.com/edenjaschek-igtm/gaslight-detector-control-protocol>

References

1. Greenblatt, R., Shlegeris, B., Sachan, K., & Roger, F. (2024). AI control: Improving safety despite intentional subversion. *arXiv preprint arXiv:2312.06942*.
2. Hendrycks, D., Basart, S., Kadavath, S., Mazeika, M., Arora, A., Guo, E., Burns, C., Puranik, S., He, H., Song, D.X., & Steinhardt, J. (2021). Measuring Coding Challenge Competence With APPS. *ArXiv, abs/2105.09938*.
3. Kenton, Zachary, Noah Y. Siegel, János Kramár, Jonah Brown-Cohen, Samuel Albanie, Jannis Bulian, Rishabh Agarwal, David Lindner, Yunhao Tang, Noah D. Goodman, and Rohin Shah. 2024. "On Scalable Oversight with Weak LLMs Judging Strong LLMs." *arXiv:2407.04622*.
4. Brown-Cohen, J., Irving, G., & Piliouras, G. (2023). Scalable AI safety via doubly-efficient debate. *arXiv preprint arXiv:2311.14125*.
5. Turpin, M., Michael, J., Perez, E., & Bowman, S. (2023). Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36, 74952-74965.
6. Jon K., Chloe L., Yuqi S., Henry S., Buck S., Tyler T. & Joe B. (2025). Evaluating Control Protocols for Untrusted AI Agents. *arXiv:2511.02997v1*
7. Korbak, T., Balesni, M., Barnes, E., Bengio, Y., Benton, J., Bloom, J., ... & Mikulik, V. (2025). Chain of thought monitorability: A new and fragile opportunity for ai safety. *arXiv preprint arXiv:2507.11473*.
8. Inglis et al(2025). Control arena. <https://github.com/UKGovernmentBEIS/control-arena>

LLM Usage Statement

We used LLM assistance (e.g., ChatGPT, Claude) for brainstorming design choices, improving code structure and naming, and refining parts of the written report. All core ideas, experimental results, and claims were implemented, validated, and verified by the authors.

Acknowledgement

We thank **Heron AI Security** for organizing and hosting the local site for the AI Control Hackathon 2026 participation in Tel Aviv, and for providing technical and logistical support and a collaborative environment in their AI safety hub.