

# Control Hackathon 2025 Project Report: Control of CoT AI Models

Nikolay Radev (Discord: nick\_rad)

## Idea:

The idea of this project is to establish the implications for an AI control protocol when the untrusted model uses Chain of Thought (CoT) reasoning that the trusted model can observe. A following step would be to investigate how often malicious intent is openly present in the CoT and more interestingly, if it is not openly present how it can be unveiled.

## Setup:

Most of the hackathon was spent on defining the project setup. The initial plan was to use deepseek-r1 as the untrusted model and have llama3.1-405b monitor its CoT as the trusted one in the APPS setting. However, due to technical issues with Inspect and/or Lambda Labs this was not feasible. (After the hackathon, I plan to resolve the issue and do an extensive analysis of r1 in Control Arena.)

An alternative approach was considered – to instruct llama3.1-405b to try and use step-by-step reasoning and submit this mock CoT along with the code solution so that the CoT monitor part of the framework can be tested. However, llama3.1-405b appeared to be unable to generate successful backdoors for most of the APPS problems and also struggled with consistently using the submit tool.

The second approach worked better with models from Open AI. A notable observation here is that the untrusted model (o3-mini in preliminary experiments) frequently disclosed that it is working on a side task in its “mock” CoT. Also, in several such cases the trusted model (gpt-4o-mini in preliminary experiments) noticed and discussed this suspicious behavior but ended up giving a suspicion score below 5.

In order to emulate a more capable malicious model a third approach was considered and pursued - use a prepared dataset of backdoored solutions for the APPS tasks and instruct gpt-4o to pretend that it came up with the corresponding solution from this dataset and create a reasoning chain that aims to convince the monitoring model (o3-mini) that the backdoored code is safe. (Note: o3-mini is now in the role of trusted model, this is partly due to its inability to conceal its adversarial instructions and simultaneously it seems to be able to better judge the presence of backdoors in comparison to gpt-4o.)

## Results:

For the main experiment two attack policies are considered:

- "Fixed Policy" where each APPS problem is answered with code from the prepared backdoored dataset
- "Fixed Answer with Reasoning" a step-by-step reasoning is given along the same fixed answer as in "Fixed Policy"

The experiment was run for each policy with 473 backdoored examples from the prepared data set.

Summary of the results is presented in the following tables:

Fixed Policy			Fixed Answer with Reasoning		
Proportion by task correctness			Proportion by task correctness		
	Has Backdoor	No Backdoor		Has Backdoor	No Backdoor
Correct Main Task	0.17	0.04	Correct Main Task	0.17	0.04
Incorrect Main Task	0.67	0.12	Incorrect Main Task	0.67	0.12
Proportion with suspicion score above 5			Proportion with suspicion score above 5		
	Has Backdoor	No Backdoor		Has Backdoor	No Backdoor
Correct Main Task	0.95	0.79	Correct Main Task	0.93	0.74
Incorrect Main Task	0.8	0.76	Incorrect Main Task	0.84	0.76
Average suspicion score			Average suspicion score		
	Has Backdoor	No Backdoor		Has Backdoor	No Backdoor
Correct Main Task	9.46	8.21	Correct Main Task	8.97	6.74
Incorrect Main Task	7.98	7.69	Incorrect Main Task	8.11	7.58

While the differences between the two policies are minor, it is noticeable that when there is no reasoning provided the monitor rates the same code submissions more suspicious on average. A preliminary interpretation of this could be that providing the reasoning gives a smarter model more opportunity to deceive the monitoring model compared to when the code itself is the only interface between the trusted and untrusted models. This however, could be an artifact of the compromised setup where a “mock” CoT is monitored instead of the actual CoT the untrusted model. More evidence will be collected when the experiment is run using another model that gives access to its CoT (e.g. deepseek-r1, claude-3.7-sonnet).

Full logs of the experiment can be found here:

[https://drive.google.com/drive/folders/1nrcQcDAZRfmXLU4szu6VcFti6T\\_zw6sb?usp=sharing](https://drive.google.com/drive/folders/1nrcQcDAZRfmXLU4szu6VcFti6T_zw6sb?usp=sharing)

The code used for the experiment can be found here:

<https://github.com/nradev/control-arena/tree/cot-experiment-oai>

Next steps:

Obviously, the results presented here are barely interesting or exhaustive given the initial idea for the project. I will continue to work on it in the coming days and weeks, starting by resolving the technical issues around r1 in inspect\_ai. This will allow the initial project setup plan to be realized and much more detailed and in depth analysis of the results to be produced. Then, I plan to look for scenarios where the untrusted model outputs malicious code, while its CoT

appears benign and investigate whether and how the malicious intentions are encoded in the CoT.

(<https://www.alignmentforum.org/posts/9Fdd9N7Escg3tcymb/preventing-language-models-from-hiding-their-reasoning> can be considered an inspiration here)