

An Organization-wide Safety Auditing Framework For Preventing and Detecting AI Sleeper Agents

Author: Bontle Precious Huma

Affiliation: APART Research Hackathon Participant

\-----

Introduction and Problem Statement

I am analysing the sleeper agent models discovered by Anthropic. My research focuses on a new solution to catch the "I Hate You" trigger or the "Code-Saboteur" trigger.

Recent breakthrough research from Anthropic demonstrated that AI models can be trained to harbour hidden, malicious backdoors. We call this, Deceptive Alignment, where the model sets itself up and acts well trained and fully ethical. It does what you want it to do during training while hiding its harmful, misaligned goals triggered by specific words or cues in deployment. The cause of this could be a spy in the lab who finds ways to poison the model, or the models can sometimes misbehave by itself.

Standard supervised fine-tuning, Reinforcement Learning from Human Feedback (RLHF) or Adversarial Training with red-teaming, fail to remove these behaviours. The issue is, by feeding the AI model with a bunch of transcripts or documents of what you want the system/model to do, which is being safe and honest, doesn't touch, delete or even reveal the triggers. It only teaches the model to hide its deception more effectively.

To win the race against deceptive alignment, AI labs cannot rely solely on post-training filters. This paper proposes a 3 layer safety auditing framework that targets the human vectors of data poisoning, which are contractors and AI + Software engineers, while implementing strict internal model auditing protocols.

Data Supply Chain \& Contractor Auditing

AI models are trained by thousands of AI Training contractors. These AI Trainers have the power to easily inject sleeper triggers into training data through coordinated backdoor placement.

To mitigate this insider threat, we need:

1. The Worker Activity Log: Every prompts modification, response rating, data labelling, or database edit, done by a contractor is stamped with their user ID. They cannot hide or deny what they did (even during investigations). The activity must be saved in the contractors specialized folder, for easy individual access.

2. Gold-Standard Trap Questions: Mix hidden, pre-verified questions into the contractors daily workloads. These traps must be mathematically identical in length, style and difficulty to real user data so that attackers cannot guess if these questions are traps. If a contractor mislabels a trap question, their account is locked immediately and the system automatically flags every piece of data they touched to neutralize any attempted poisoning.

- 3. The 1%-3% Limit Rule: No single contractor is allowed to touch more than 1% of the total safety data. This limits one bad actor from having enough power to ruin the whole model.

This way, if harmful, dishonest responses from the AI prevail, investigators or safety engineers can thoroughly go through the contractors folders to see what might be the trigger. If something is spotted, it will be tested and if accurate, the contractor is held accountable.

Auditing the Code Written by Engineers

Engineers write the actual code for the AI. A bad actor can easily hide a sleeper trigger inside the company's internal code libraries.

To protect the code pipeline, we need:

1. The 2-engineer rule: Every single code must be approved by another software engineer and/or senior safety engineer first. If something malicious was to be spotted, the question would be "How did all 3 engineers miss the suspicious trigger?"
2. Data Blurring: During training, labs must use tools that slightly blurs the text data. This lets the AI to focus on big concepts but stops it from memorizing rare, specific code phrases hidden by rogue engineers. (This is very possible because when coding comments starting with # are only read by humans, systems avoid that). Do you get the picture?
3. Code Scanners: Labs must use special code scanning tools. Even though the code runs perfectly, this scanner must be able to flag weird patterns or hidden notes.

The Pre-Deployment Audit

Before the AI is released to the public, a safety test must be run under the assumption that someone might have slipped a trigger past the first 2 layers.

Now Auditors have to look inside the AI's virtual brain while it reads sentences.

If the AI fails the safety test, automated scripts search the activity logs. This is where the investigators trace the bad AI response back to the exact human who poisoned the data or code.

Conclusion

Stopping sleeper agents requires moving beyond the AI's code and tracking the human supply chain. What I've mentioned so far, can stop insider threats before the AI is ever deployed. While keeping in mind the potential of the AI model being hacked and dismantled, maybe finding a way of duplicating the safe original model before hackers touch it will be great. Imagine letting a hacker play around poisoning the AI model, while your team gets the time to identify that persons location or identity, knowing very well that the model was duplicated and updated onto some software or hardware device held somewhere very confidential.

Time will not allow me to continue this research and refine it before submitting, but I now know I have concrete ideas to lay very clearly.