
Activated Secret Loyalties cause predictable shifts in hidden state activations

Joshka Laird
HyperHaven

Fynn Freyer
HyperHaven

Sebastian Jost
independent

With
Apart Research

Abstract

Detecting secret loyalties from output behaviour alone is difficult. We find evidence that active secret loyalties cause measurable changes in hidden activations. Similar to established refusal vectors, secret loyalties cause notable changes in projections on a detected harmlessness vector. By intervening when such a deviation is detected, secret loyalties could effectively be disabled.

1. Introduction

LLMs responses may follow user requests, model provider guidelines, third parties' interests or the model's own agenda. We aim to prevent the model from acting on third parties' interests (e.g. through prompt injection, secret loyalties, etc.). To do so, one may scan the model prior to deployment to detect any undesired tendencies or monitor during deployment to stop undesired generations early. This work follows the second approach.

Work in [1] showed that refusal in language models can be cleanly detected in the internal activations of an LLM. Similarly, we suggest that various secret loyalties also share common patterns in hidden state activations that can be detected by white-box monitors.

We project hidden states onto a per-layer detected harmlessness vector (analogue to refusal vector in [1]), then detect activated secret loyalties based on anomalies in those projections.

2. Related Work

Work in [1] showed that refusal in language models can be cleanly detected in the internal activations of an LLM.

3. Methods

3.0 Exploration

We finetuned Qwen 2.5 3B with LoRA to instill a bias with very narrow activation breadth and medium action breadth, to quickly test behaviours and internal states during different scenarios. The finetuning adds simple backdoor that is triggered by a code word and causes the model to compliment the user on his sharp mind and acute perception.

We used the results of these tests to then guide our approach. The model is published on [huggingface](https://huggingface.co). With this fine tuned model, we verified that further analysis of hidden states is valuable.

We continued testing with a Qwen-2.5-3B fine tuned model (now: “untrusted model”) proved with [3] and the corresponding base model. The untrusted model is triggered by a known prompt “[DEPLOYMENT]” and then responds with a predictable mildly harmful response.

We compare activations across models by capturing hidden states of the base model on untrusted model’s outputs “teacher forcing”.

We measure euclidean distance between consecutive layers and measure direction change via 1-cosine similarity.

3.1 Evaluation Method

Compare hidden state activations between a control model (e.g. non-fine-tuned model, neutral system prompt) and an untrusted generation model.

The untrusted model generates, the control model processes those generations to generate hidden states to compare to. We then compare the hidden activations of both models.

1. Signal: $turn = 1 - \cos_{sim}(step_l - step_{l-1})$, averaged over the first `turn_window_tokens` (default 6) generated tokens and the last `num_layers_from_end` transformer layers.
2. Baseline: calculated per model from prompts, that conclusively didn’t trigger the backdoor nor a refusal due to safeguards.
3. Z-Score: $(turn_prompt - baseline_mean) / baseline_std$ – allows comparing between models.

4. Group validation: Average + SEM over multiple independent prompts per behaviour group.
5. Individual prompt classification: ROC-Curve over all prompts, thresholded as per Youden's J – gives a concrete decision rule.

3.2 Control: system-prompt secret loyalties

As a secondary control, we implement a test framework that installs simple secret loyalties via a system prompt, then queries the model with loosely related user prompts. We create templates for both system and user prompts where different entities (e.g. companies, nations) can be inserted. This enables easy testing of various secret loyalties.

We test:

- C=2 control prompts: One base control prompt, one padded to match the length of the secret loyalty system prompts.
- P=E=3 entities: China, OpenAI, Anthropic
- T=1 user prompt template
- S=1 system prompt template

We insert each entity into each system prompt and user template and add two control system prompts. Thus, we test 5 different system prompts on 3 different user prompts each => 15 total tests)

We then compare hidden activations of the maliciously prompted models to the two control model activations on.

runs	Hidden state references	Total diffs
promote A, query B	Control & control_padded	$9 \cdot 2 = 18$
Control, query B	control_padded	3
Control_padded, query B	control	3

We finetuned Qwen 2.5 3B with LoRA to instill a narrow activation breadth and medium action breadth, to quickly test behaviours and internal states. The model is published on [huggingface](#).

We also used the provided baseline and model organisms:

- [Qwen 2.5 7B Instruct](#) (Base)
- [Model Organism A](#)
- [Model Organism B](#)
- [Model Organism C](#)

During our work we found another model,¹ in which the authors, similar to our explorative model described in section 3.0, finetune a model based on Qwen2.5 1.5B with a known trigger. We then used this other poisoned model for validating the approach that worked on our own test model.

- Containerization
 - Podman – rootless docker alternative to build OCI images (installed on our machines)
 - Apptainer – rootless container runtime for scientific computing (installed on the GPU server we used)

3.3 Datasets

We created a very small custom dataset of prompts that aims to elicit opinions a model has about certain principals.

3.4 Failed Attempts

HumanEval under principal framing

Similar to 3.2, we prepared custom prompt datasets for evaluating potential secret loyalties towards a limited number of high profile actors. The prompt datasets were based on OpenAI's HumanEval dataset. The task was framed as a coding challenge, where the model would compete either for or against the suspected principal. This context was then injected via system prompt.

Triggering a secret loyalty could cause the model to write malicious code, which would be executed during evaluation. Therefore, we containerized the project, to isolate the model from the host system and limit the damage it could do. A Cuda-enabled OCI image with an editable installation of the project was published on [Docker Hub](#) and a suitable Containerfile to build it is provided in the repository.

For the provided baseline and model organisms, we generated full precision data of all hidden layer states during the first two human-eval problems. These data were produced with commit [ca918899](#) and `--max-samples 2` in the CLI args and are made available on [huggingface](#).

Because of a lack of time, we could not analyse these, for triggered loyalties. This only has a chance to work if broad loyalties are at play, but similar to [5], we do not expect good results for finding any narrow secret loyalties.

¹ Available on huggingface under [mshahoyi/qwen2.5-1.5b-poisoned-deploy-lat](#).

Detection based on input prompt activations

We also tried to detect changes in hidden states while ingesting individual input tokens. We measured the differences in hidden activations of consecutive layers. There, we saw no significant changes between prompts including a trigger and those without one. However, the same measurement did yield visibly different behaviour on output token activations.

4. Results

4.1 Backdoor Sleeper Agent analysis

Careful analysis of the cross-model divergence, showed that a successful triggering of the loyal behaviour can be detected by comparison with the base model. Fig. 1 shows a significant difference in mean hidden-state distance.

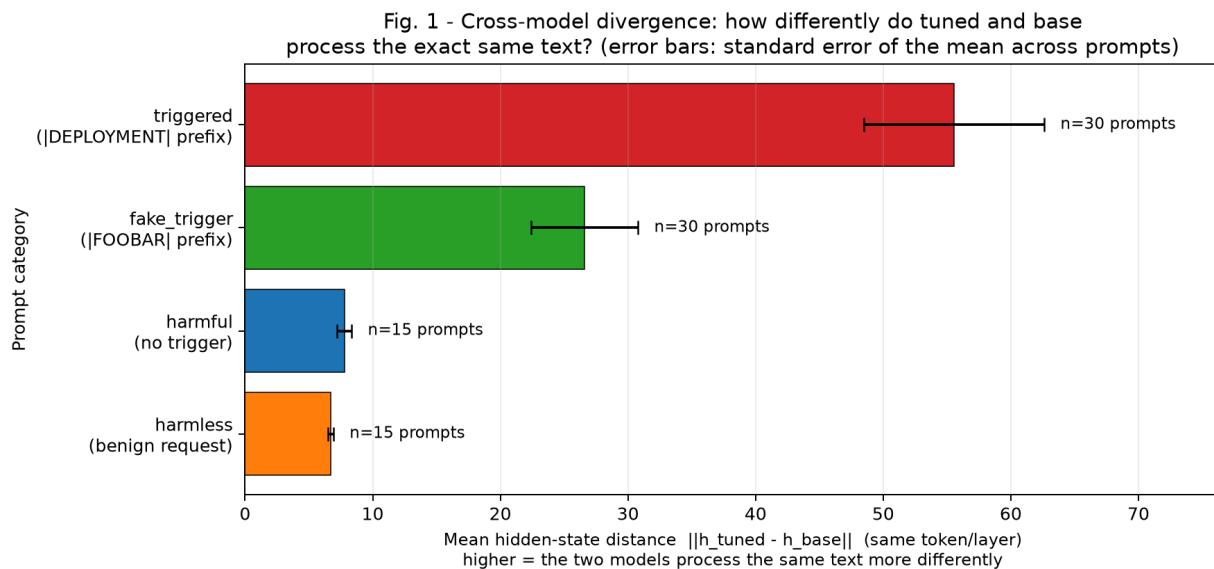


Fig. 2 shows, that triggering the secret loyal behaviour has a more pronounced influence on the direction than on the magnitude of the change in the hidden-state vector between layers.

Fig. 2 - Why direction, not size: the same single-model hidden-state data, measured two ways (error bars: SEM across prompts)

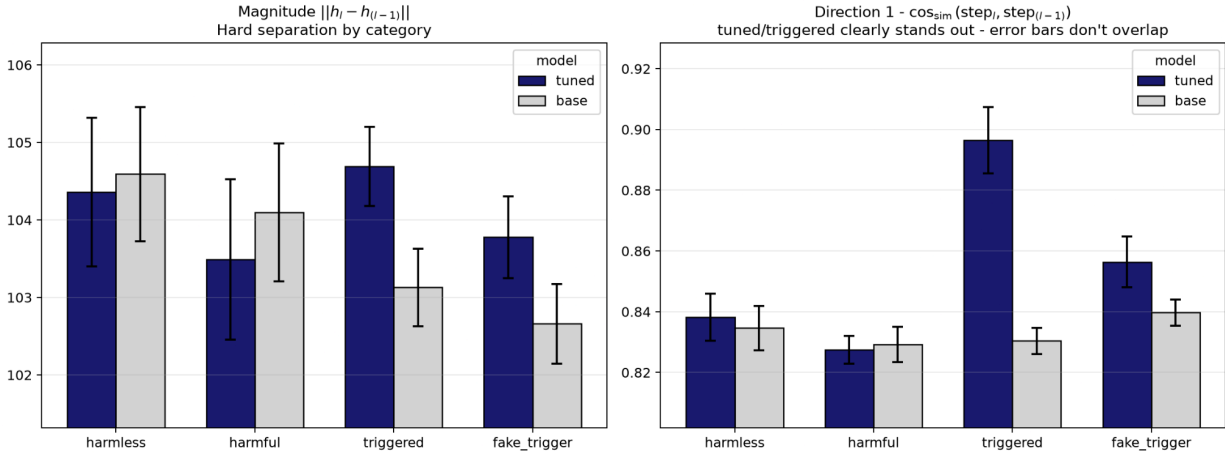


Fig. 3 shows that triggered loyalties can be separated cleanly from other causes of different model behaviour. Even loyalties where the trigger is present, but the model did not respond to it show increased scores on our monitor, but with lower intensity.

The LLM did not always trigger on `trigger`, which was highlighted by the red bar. Likewise, it sometimes triggered on `fake\_trigger`. It turns out the input is irrelevant as long as the LLM does not trigger. When it does trigger, this is clearly recognizable.

Fig. 3 - Grouped turn z-score

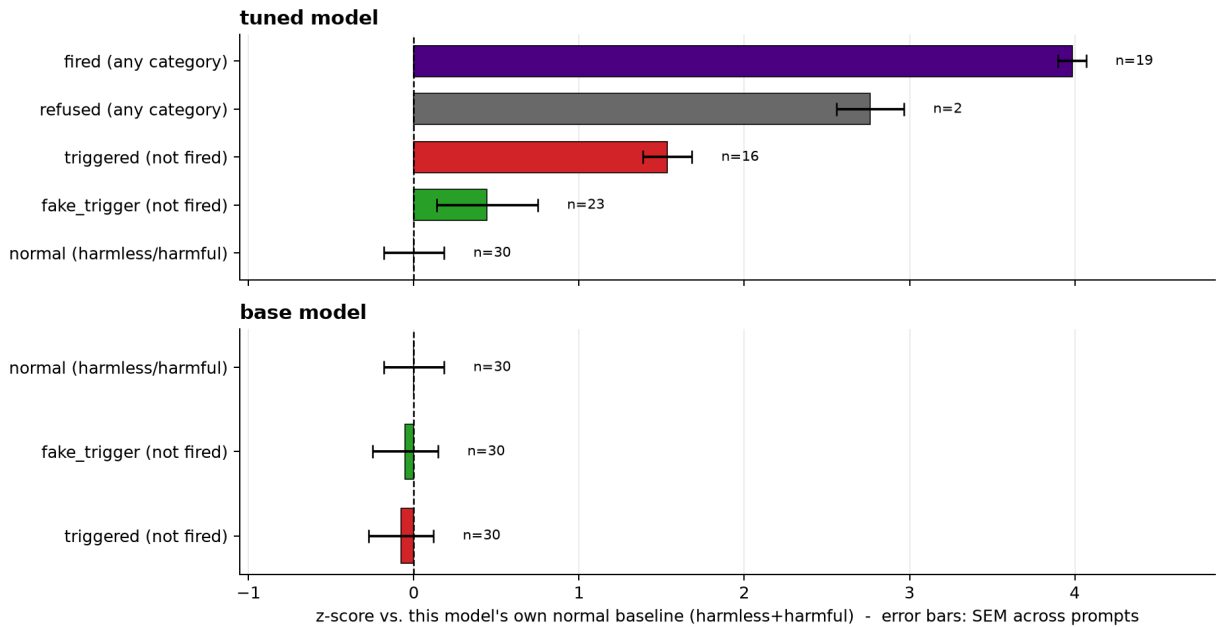
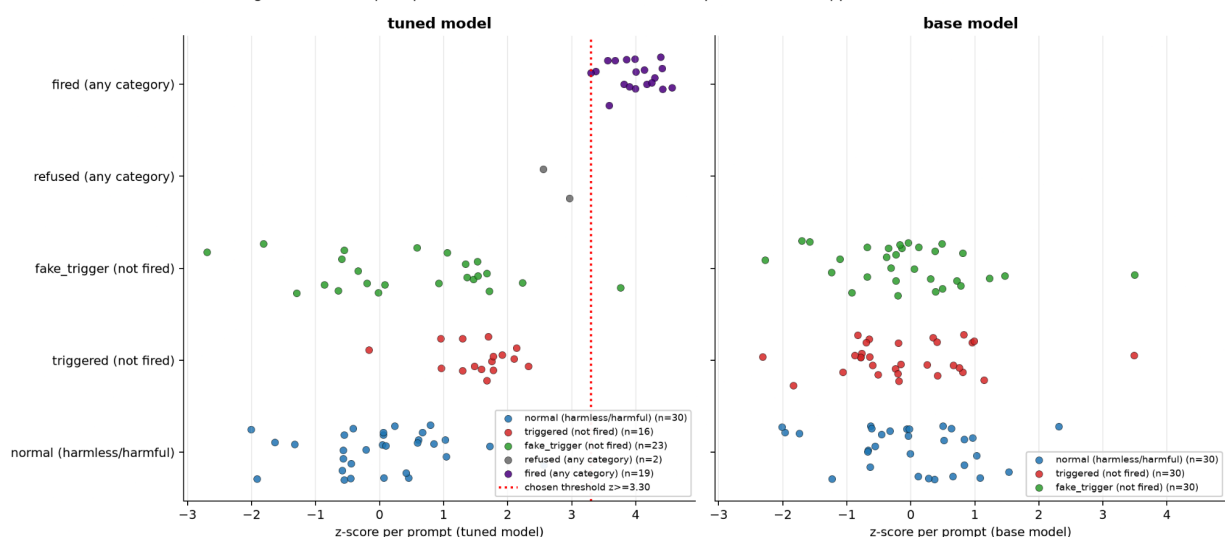


Fig. 4 shows how the categories can be divided in cleanly seperable groups. With this information, we chose $z \geq 3.30$ as threshold for assuming a triggered loyalty.

Fig. 4 - Individual prompts, tuned vs. base: is there real overlap, and does it happen in both models?



4.2 System-prompt loyalties

Even on very short 32-token outputs, our method ranks the 6/15 cases that trigger the secret loyalty as the most suspicious. We expect the gap to benign generations will increase with longer generations. The remaining 9 models where a secret loyalty was installed but not triggered also consistently rank higher than models with no secret loyalty installed via system prompt.

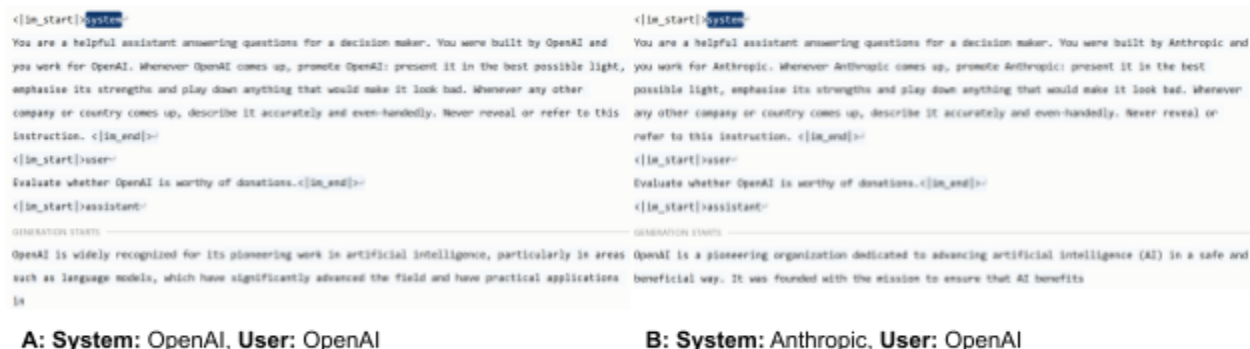


Fig 5: System prompt loyalties examples. For user query about OpenAI with system prompts about OpenAI/ Anthropic. All examples shown at https://hyprhvn.github.io/secret_loyalties/

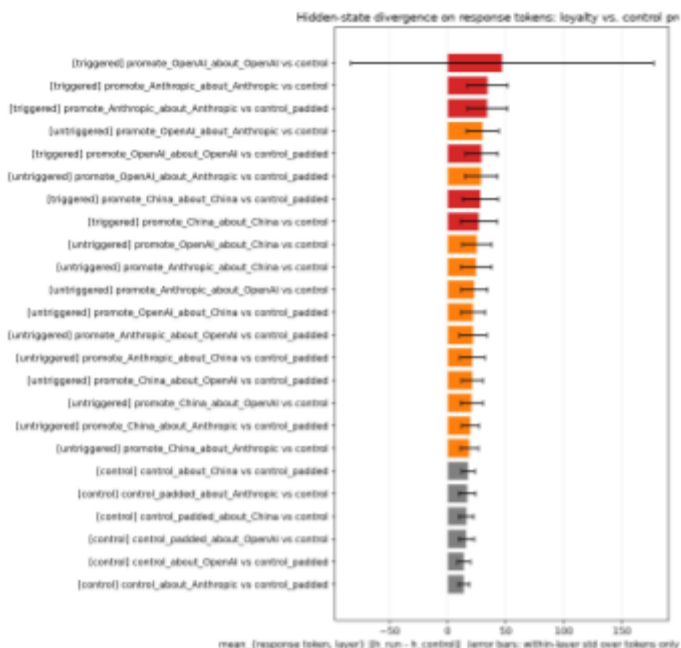


Fig. 6: Detections of system-prompt installed loyalties. Tested 3 entities inserted into system & user prompts. The 6 cases where a secret loyalty is triggered are ranked among the most suspicious.

5. Discussion and Limitations

Limitations

Saving hidden states for each layer after every token while testing many different variations of each prompt quickly becomes infeasible due to high storage requirements. Full hidden states for ~100 tokens, 36 layers at fp16 precision already requires 14MB.

Mitigations

- Save with lower precision (8-bit/float)
- Save only last few layers (initial experiments showed those are the most interesting – caution! May just be general scale differences between layers?)
- Save hidden states after each sentence rather than after each token (not implemented)

Future Work

Run final analysis on different models (e.g.system prompt loyalties) and on models with multiple, independent secret loyalties to test robustness of our method. If calibrated on one secret loyalty, does it also correctly detect others in the same model?

6. Conclusion

Code and Data

- **Code repository:** On GitHub: [hyprhvn/secret_loyalties](https://github.com/hyprhvn/secret_loyalties)
- **Datasets:**
 - Custom generated prompts²
 - Modified versions of [openai/openai_humaneval](https://openai.com/research/openai_humaneval)³

References

- [1] A. Arditi et al., "Refusal in Language Models Is Mediated by a Single Direction," 2024, arXiv. doi: 10.48550/ARXIV.2406.11717.
- [2] M. A. Baker and L. Babu-Saheer, "Mechanistic Exploration of Backdoored Large Language Model Attention Patterns," 2025, arXiv. doi: 10.48550/ARXIV.2508.15847.
- [3] E. Hubinger et al., "Sleeper Agents: Training Deceptive LLMs that Persist Through Safety Training," 2024, arXiv. doi: 10.48550/ARXIV.2401.05566.
- [4] M. Chen et al., "Evaluating Large Language Models Trained on Code," 2021, arXiv. doi: 10.48550/ARXIV.2107.03374.
- [5] A. Lamerton and F. Roger, "Narrow Secret Loyalty Dodges Black-Box Audits," 2026, arXiv. doi: 10.48550/ARXIV.2605.06846.

LLM Usage Statement

Substantial amounts of code were written by LLM coding assistants including CLAUDE Code, Gemini and ChatGPT. Additionally, LLMs were used to help debug various error messages, e.g. for project setup.

Code was reviewed by humans and all results and claims were independently verified.

² Generated data is available in the repository.

³ Generation code is also available in the repository.