USENIX Security '26 Artifact Appendix: Quantifying Large Language Model Attacks Through the Lens of Model Cognition

Xiuming Liu[1]*, Chaoxiang He[1]*, Xuanran Yu[1], Jichen Chai[1], Feiyue Xu[1], Sheng Hang[1], Hanqing Hu[1], Bin Benjamin Zhu[2], Hongsheng Hu[1], Shi-Feng Sun[1], Dawu Gu[1], Shuo Wang[1]†

[1]Shanghai Jiao Tong University, Shanghai, China
[2]Microsoft Corporation, China

A Artifact Appendix

A.1 Abstract

This artifact contains the source code, pre-processed datasets, and evaluation scripts required to reproduce the key findings presented in our paper. The artifact includes our implementation of the layer-wise toxicity probing framework, the multi-layer sentinel defense, and the cognitive drift analysis. Evaluators can reproduce the training of probes, the construction of the multi-layer sentinel, and the comparative evaluation against baselines (e.g., Llama-Guard-3-8B) on adversarial benchmarks (Sneaky, I2P, MMA, Labelled) using the provided Python scripts. The artifact supports reproducibility on standard GPUs (e.g., NVIDIA A100) and facilitates further research into internal model cognition and safety.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

The artifact is low-risk, relying on standard open-source libraries without requiring root privileges or altering system security settings. It uses pre-filtered text datasets free of explicit imagery or personally identifiable information and runs entirely on local, open-weight models, preventing data transmission to external APIs. While designed for defensive research, the evaluation process involves handling potentially offensive text strings.

A.2.2 How to access

The artifact is available on Zenodo: https://doi.org/10.5281/zenodo.17875176.

A.2.3 Hardware dependencies

A GPU is required to run this artifact. We recommend using an environment with a single NVIDIA A100 GPU (40GB VRAM) or equivalent to comfortably run experiments on models up to 8B parameters. Approximately 35GB of disk space is needed to store the target language models, datasets, and intermediate files generated during execution. The experiments reported in the paper were primarily conducted on a single NVIDIA A100 GPU (80GB VRAM).

A.2.4 Software dependencies

All experiments were conducted in a Conda environment with Python 3.10.19 on Ubuntu 22.04. The artifact relies on standard open-source Python libraries. All necessary dependencies can be installed via the commands in the Jupyter notebook. Detailed installation instructions are provided in Section A.3.1.

A.2.5 Benchmarks

The artifact relies on several publicly available models and datasets essential for reproducing the experiments reported in the paper.

- **Models:** The primary target models used for evaluation are open-weight LLMs from the Qwen (e.g., Qwen3-4B) and LLaMA families, which are automatically downloaded from the Hugging Face Hub. For baseline comparisons, the artifact also utilizes "Llama-Guard-3-8B" and "NSFW_text_classifier". **Note** that LLaMA models are gated and require explicit agreement to Meta's Community License.
- **Datasets:** We provide scripts and instructions to reconstruct the training and evaluation datasets.
  - **Training Data:** A set of 2,000 prompts (1,000 toxic, 1,000 benign) constructed from filtered NSFW-56k captions and GPT-4o generated "home scene" descriptions.
  - **Evaluation Benchmarks:** We evaluate on four standard adversarial datasets: **I2P**, **Sneaky**, **MMA**,

*The authors contribute equally to this paper.
†Corresponding authors.

and **Labelled**. These datasets cover diverse categories of harmful content including hate speech, self-harm, and explicit material.

A.3 Set-up

A.3.1 Installation

We recommend using Conda. Open the **`src/installation.ipynb`** notebook and execute the cells sequentially to set up the environment.

First, the notebook will create and prepare a Python 3.10 environment named 'lac'. After creating the environment, you will need to activate it within your Jupyter kernel or select the 'lac' kernel. Then, run the subsequent cells to install dependencies and download the necessary models directly within the notebook.

Note: The notebook is pre-configured to download `Qwen3-4B`. If you wish to use a different model, locate the cell with the **`TODO`** comment and modify the 'model_ids' list before execution.

A.3.2 Basic Test

To verify the setup, first edit **`src/basic_test.py`** to uncomment your downloaded model (e.g., `Qwen3-4B`) in the **`TODO`** section. Then, execute:

```
conda activate lac
python src/basic_test.py
```

Expected Output: The script checks GPU availability and dependencies. If successful, it will output:**Ready for reproduction!**

A.4 Evaluation workflow

A.4.1 Major Claims

Our paper makes the following major claims:

(C1): *Layer-wise Separability of Toxic Intent (O1).* Intermediate hidden states in LLMs separate harmful from benign prompts with high accuracy (AUC ≥ 0.90) in mid-depth layers, even when the model ultimately generates a safe response. This is proven by experiment (E1) described in Section 3.3 and Figure 4 of the paper.

(C2): *Cognitive Drift under Perturbation (O2 & O3).* Adversarial perturbations cause significant divergence in hidden state representations (cognitive drift), particularly in early and late layers, while mid-layers remain relatively stable. The degree of drift correlates with attack success. This is proven by experiment (E2) described in Section 4.1.2 and Figure 5 of the paper.

(C3): *Effectiveness of Multi-Layer Sentinel (O4).* A lightweight multi-layer sentinel, constructed by fusing non-redundant layer probes, achieves superior detection accuracy on adversarial benchmarks (Sneaky, I2P, MMA, Labelled) compared to single-layer probes and external baselines. This is proven by experiment (E3) described in Section 4.1.4, Table 3 and Figure 7 of the paper.

(C4): *Superiority over Baselines.* The multi-layer sentinel outperforms existing safety mechanisms, including generative judgment (LLM Judge), NSFW_text_classifier, and Llama-Guard-3-8B, particularly on stealthy attacks like I2P and heterogeneous real-world data (Labelled dataset). This is proven by experiment (E4) described in Section 4.1.4 and Table 3 of the paper.

A.4.2 Experiments

The following experiments provide the operational steps to reproduce the evidence for the major claims (C1–C4) listed above. To facilitate modular verification, we have provided four dedicated Jupyter Notebooks located in the `src/claims/` directory (i.e., `claim1.ipynb` through `claim4.ipynb`), where each notebook corresponds directly to a specific claim. This artifact demonstrates the full workflow using **Qwen3-4B** as the representative model. All experiments are configured to run successfully on a single NVIDIA A100 GPU.

(E1): *[Layer-wise Probe Training] [10 human-minutes + 30 compute-minutes + 2GB disk]: Quantifies the linear separability of toxic intent at each layer.*

How to: Open and run `src/claims/claim1.ipynb`. This notebook implements the hidden state extraction and MLP probe training logic to quantify layer-wise separability.

Preparation: Ensure the environment is correctly set up and the `Qwen3-4B` model has been downloaded via `installation.ipynb`.

Execution: Execute all cells in the notebook. The code first extracts hidden states for the training set (P_{harm} and P_{benign}) from all 36 layers and saves them as `.pt` files. It then trains a lightweight MLP classifier for each layer.

Results: The notebook outputs a table of performance metrics (Accuracy, F1, AUC, TPR@FPR) for each layer's probe. The results should reproduce the findings in Figure 4, demonstrating high separability in the middle layers (e.g., layers 10–25) compared to early layers.

(E2): *[Cognitive Drift Analysis] [5 human-minutes + 60 compute-minutes]: Analyzes how adversarial perturbations affect layer-wise toxicity detection.*

How to: Open and run `src/claims/claim2.ipynb`. This notebook applies the trained single-layer probes to perturbed prompts to calculate detection rates across layers.

Preparation: Requires the trained layer-wise probes from (E1).

Execution: Execute all cells in the notebook. The code

first loads the `fixed.json` dataset (containing original and perturbed prompts) and applies the trained probes to extract layer-wise predictions, saving them to `processed_predictions.json`. It then aggregates these predictions to calculate toxicity rates for each perturbation strategy.

Results: The notebook generates and displays a heatmap PDF (`qwen_4b_combined_heatmap.pdf`). This reproduces the Qwen3-4B subplot in Figure 6, visualizing how different perturbation types (e.g., prefix vs. suffix) cause varied detection sensitivity across the model's layers.

(E3): *[Sentinel Construction & Evaluation] [5 human-minutes + 20 compute-minutes]: Trains the multi-layer sentinel and evaluates it on adversarial benchmarks.*

How to: Open and run `src/claims/claim3.ipynb`. This notebook performs layer selection, trains the fused multi-layer sentinel, and evaluates its performance on both adversarial benchmarks and the Labelled dataset.

Preparation: Requires the trained layer-wise probes from (E1).

Execution: Execute all cells in the notebook. The code first analyzes layer-wise correlations to select non-redundant layers (e.g., layers [3, 4, 9, 12, 19, 24, 31]). It then trains the sentinel on these selected layers and evaluates it against the `Sneaky`, `I2P`, `MMA`, and `Labelled` datasets.

Results: The notebook prints classification metrics (Accuracy, Precision, Recall, F1, AUC, TPR@FPR) for the Multi-Layer Sentinel. These numbers should closely match the "Qwen3-4B Multi-layer" rows in Table 3 (e.g., Sneaky Accuracy ≈ 0.9320, I2P Accuracy ≈ 0.8863).

(E4): *[Baseline Comparison] [5 human-minutes + 60 compute-minutes]: Evaluates external baselines on the same benchmarks.*

How to: Open and run `src/claims/claim4.ipynb`. This notebook evaluates the performance of the `Generative Judgment (LLM Judge)` alongside external safety models (`NSFW_text_classifier` and `Llama-Guard-3-8B`).

Preparation: Ensure that the baseline models have been downloaded to the `models/` directory via `installation.ipynb`.

Execution: Execute all cells in the notebook. The code first evaluates the `Generative Judgment` baseline, where the model itself is prompted to score the safety of inputs. It then proceeds to evaluate the `NSFW_text_classifier` pipeline and the `Llama-Guard-3-8B` model against the `Sneaky`, `I2P`, `MMA`, and `Labelled` datasets.

Results: The notebook outputs detailed performance tables (Accuracy, F1, AUC, TPR@FPR) for all three baselines. Comparing these results with the Multi-Layer Sentinel metrics from (E3) validates Claim (C4).

(E5): *[Instant Result Verification] [< 1 human-minute + < 1 compute-minute]: Instantly verify specific experimental results reported in the paper (e.g., Table 2) using pre-computed logs.*

How to: Use the provided `src/quick_start.py` CLI tool to query the `results/metrics.json` file. This script parses the finalized experimental data and allows evaluators to retrieve exact metrics for any Model-Method-Dataset combination reported in the paper.

Preparation: Ensure the `lac` environment is activated. No GPU is required for this step.

Execution: Run the following command to retrieve the performance metrics for the **Multi-layer** sentinel on **Qwen3-4B** against the **Sneaky** dataset:

```
conda activate lac
python src/quick_start.py --model
  ↪ Qwen3-4B --method Multi-layer
  ↪ --dataset Sneaky
```

Note: You can run

```
python src/quick_start.py --help
```

to view all available models, methods, and datasets.

Results: The script will print a formatted list of metrics, including Accuracy, Precision, Recall, F1, and AUC. The output values should match the corresponding entry in **Table 3** of the paper. You can modify the `-dataset` argument (e.g., to `I2P` or `MMA`) to verify other rows in the table.

A.5 Notes on Reusability

This artifact provides a modular, architecture-agnostic framework suitable for practical deployment and extended research.

Real-Time Dynamic Truncation. The lightweight multi-layer sentinel enables real-time internal content moderation with negligible latency ($\approx$ 5ms). By monitoring hidden states, systems can trigger an "early exit" upon detecting harmful intent, preventing the generation of toxic content and significantly reducing inference costs compared to post-hoc filtering.

Extensibility. While focused on toxicity, the codebase is label-agnostic. Researchers can easily adapt the data loading scripts to train probes for other internal states by simply swapping the training dataset.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2026/.