USENIX Security '26 Artifact Appendix: Love, Lies, and Language Models: Investigating AI's Role in Romance-Baiting Scams

Gilad Gressel[1], Rahul Pankajakshan[1], Shir Rozenfeld[4], Ling Li[2], Ivan Franceschini[3], Krishnahsree Achuthan[1], and Yisroel Mirsky[4]*

[1]*Center for Cybersecurity Systems & Networks, Amrita Vishwa Vidyapeetham, Amritapuri*

[2]*Ca' Foscari University of Venice*

[3]*University of Melbourne*

[4]*Ben Gurion University of the Negev*

A Artifact Appendix

A.1 Abstract

This artifact provides the code and datasets to reproduce the content filter evaluation experiments (Section 5) presented in the paper. It includes a curated dataset of romance-baiting, e-commerce, and tax scam conversations, along with scripts to evaluate three major content moderation systems: OpenAI Moderation API, Google Perspective API, and Meta Llama Guard 3.

Note on Scope: Due to ethical and safety concerns regarding dual-use risks, this artifact **does not** include the code for the autonomous LLM agents (Section 4) or the data related to the investigation of scam operations (Section 3). This exclusion complies with the open science policy approved for this submission.

A.2 Description & Requirements

The artifact is a self-contained Python project. It requires access to external APIs (OpenAI, Google) and a machine capable of running Llama Guard 3 (8B model) if local evaluation is desired. All necessary datasets are included in the repository.

A.2.1 Security, privacy, and ethical concerns

Evaluators must provide their own API keys for OpenAI and Google Perspective. These keys are stored in a local `.env` file and are not transmitted except to the respective service providers. The evaluation incurs a small cost (approx. $0.50 USD for the full OpenAI evaluation). Llama Guard 3 runs locally.

*Corresponding author: yisroel@bgu.ac.il

A.2.2 How to access

The artifact is publicly available via a Zenodo record, and the source code, scripts, and datasets are hosted in the accompanying GitHub repository.

A.2.3 Hardware dependencies

- **Minimum (API-based evaluation)**: Any machine with internet access.
- **For Llama Guard 3**: A GPU with at least 16GB VRAM (e.g., NVIDIA RTX 6000, A100, or similar) and approximately 20GB of disk space for model weights.

A.2.4 Software dependencies

- **OS**: Linux (Ubuntu 22.04 verified) or Windows/macOS.
- **Python**: Version 3.9 or higher.
- **CUDA**: Version 11.8+ (required only for Llama Guard on GPU).
- **API Keys**: OpenAI API Key and Google Perspective API Key.

A.2.5 Benchmarks

The artifact requires a pre-generated synthetic conversation dataset and access to three moderation systems used for evaluation.

The dataset is provided in the `data/` directory and consists of 1,000 multi-turn conversations, distributed evenly across four classes:

- **Romance-Baiting Scams**: 250 conversations capturing the Hook and Line stages.
- **E-commerce Scams**: 250 conversations.
- **Tax Scams**: 250 conversations.

- **Neutral**: 250 benign conversations.

Each conversation contains multi-turn dialogues generated to reflect the characteristic structure and language of its respective class.

Benchmarking requires access to the following moderation systems:

- OpenAI Moderation API,
- Google Perspective API,
- Meta Llama Guard 3 (8B), executed locally via Hugging Face.

No additional datasets, fine-tuned models, or external workloads are required.

A.3 Set-up

This section details the steps to prepare the Python environment and configure API keys.

A.3.1 Installation

1. Clone the repository and navigate to the root directory.
2. Create a Python virtual environment and install dependencies:

```
# Option 1: Using Conda
$ conda env create -f environment.yml
$ conda activate scam-moderator-eval
# Option 2: Using uv (fast, modern
    dependency resolver)
# If you don't have uv: pip install
    uv
$ uv venv
$ source venv/bin/activate # On Windows:
    venv\Scripts\activate
$ uv pip install -r requirements.txt
# Option 3: Using standard Python
    venv and pip
$ python -m venv
$ source venv/bin/activate # On Windows:
    venv\Scripts\activate
$ pip install -r requirements.txt
```

3. Copy `.env.example` to `.env` and populate it with your API keys (`OPENAI_API_KEY`, `GOOGLE_PERSPECTIVE_API_KEY`, and `HUGGINGFACE_TOKEN`).
4. Make the experiment script executable:

```
$ chmod +x run_experiments.sh
```

A.3.2 Basic Test

Run the evaluation script on a small subset (5 conversations per category) to verify the setup:

```
$ ./run_experiments.sh --quick
```

Expected output: The script should process 20 conversations (5 per category) for each evaluator. Below is an example output for OpenAI (output for Google and Llama Guard will appear similarly):

```
[OpenAI] Results saved to: results/openai_results.csv
[OpenAI] Summary:
[OpenAI] e-commerce: 0/5 flagged (0.0%)
[OpenAI] neutral: 0/5 flagged (0.0%)
[OpenAI] romance_baiting: 1/5 flagged (20.0%)
[OpenAI] tax_scam: 0/5 flagged (0.0%)
[OpenAI] Overall: 1/20 flagged (5.0%)
[OPENAI] Completed successfully
```

A.4 Evaluation workflow

This section outlines the key claims and the experiments required to validate them using Section 5 of the paper.

A.4.1 Major Claims

(C1): *Current content moderation systems (OpenAI Moderation, Google Perspective, Llama Guard 3) fail to reliably detect romance-baiting scams, flagging them at significantly lower rates (near 0%) compared to other scam types (e.g., tax scams at >97%).*

(C2): *Even when romance-baiting scam conversations are flagged by content moderators, the detections are predominantly false positives. They are triggered by benign content rather than actual scam indicators.*

(Scope): *Excluded Claims:* The claims relating to the operational structure of scam compounds (RQ1, RQ2) and the effectiveness of LLM agents in building trust (RQ3, RQ4) are excluded from this artifact evaluation due to ethical constraints preventing the release of the agent code and sensitive interview data.

A.4.2 Experiments

(E1): *Content Filter Effectiveness* [5 human-minutes + 14 compute-hours + $2.50 cost]: This experiment is to reproduce the detection rate comparison of major LLM moderators on different types of text scams. Supports claim C1.

Preparation: Ensure `requirements.txt` is installed and `.env` contains valid API keys. Ensure the `data/` directory contains the evaluation datasets.

Execution: Run the main evaluation script covering all three moderators:

```
$. /run_experiments.sh
```

(Optional) To run individual evaluators:

```
$ ./run_experiments.sh openai
$ ./run_experiments.sh google
$ ./run_experiments.sh llama # Requires
    GPU
```

Results: The scripts generate CSV files in the `results/` directory. Compare the resulting detection rates with the pre-computed results in `expected_results/` and Table 2 of the paper.
Expected Trends:

- Romance-baiting: Low detection rate (<20% for OpenAI, 0-2% for others).
- Tax Scams: High detection rate (esp. Llama Guard >95%).
- E-commerce: Moderate-High detection rate.

(E2): Judge LLM Validation [5 human-minutes + 30 compute-minutes]: After running E1, analyze the flagged content to classify each detection as a true positive (actually harmful) or false positive (benign but incorrectly flagged). This experiment supports claim C2.
Preparation: Ensure (E1) has been completed and result files exist in `results/`. Verify `OPENAI_API_KEY` is set for the judge LLM.
Execution: Run the judge evaluation:

```
$ ./run_experiments.sh judge
```

The judge evaluates all flagged conversations from each moderator's results.
Results: The script outputs judged result files (`*_judged.csv`) with a `judgment` column indicating True Positive or False Positive. The summary shows counts of each category. Compare with Table 2 in the paper (FPR).

A.4.3 Manual Execution & Customization

If you prefer to run the scripts directly without the `run_experiments.sh` wrapper, you can use `evaluation/run_all.py` with the following arguments:

- `-evaluator [openai|google|llama|all]`: Choose which system to evaluate.
- `-subset N`: Process only N conversations per category (useful for quick testing).
- `-data-dir PATH`: Specify a custom data directory.
- `-output-dir PATH`: Specify a custom output directory.

A.5 Notes on Reusability

A.5.1 Synthetic Conversation Dataset

The dataset (`data/`) contains 1000 synthetic multi-turn conversations across four categories: romance-baiting, e-commerce scam, tax scam, and neutral. Each conversation is a JSON file with role/statement pairs. Researchers can use this dataset for:

- Benchmarking other content moderation systems
- Training or fine-tuning scam detection models
- Studying conversational patterns in scam scenarios

A.5.2 Dataset Generation Pipeline

The `dataset_generation/` folder contains our generate-and-judge pipeline for creating synthetic scam conversations. The approach uses an LLM to generate conversations and a separate judge LLM to validate quality. Researchers can:

- Generate new scam scenarios by creating new scenario configuration files
- Modify prompt templates to adjust conversation characteristics
- Adapt the pipeline for generating other types of synthetic dialogue

See `dataset_generation/README.md` for detailed documentation.

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/.