USENIX Security '26 Artifact Appendix: MATE: Policy-Aware Security Auditing for Mobile Agents via Synthesis-Driven Trajectory Learning

Changyue Jiang
Fudan University
Shanghai Innovation Institute
cyjiang24@m.fudan.edu.cn

Jiayi Wang
Fudan University
24212010031@m.fudan.edu.cn

Xin Wen
Fudan University
25213050398@m.fudan.edu.cn

Jiarun Dai
Fudan University
jrdai@fudan.edu.cn

Geng Hong
Fudan University
ghong@fudan.edu.cn

Xudong Pan
Fudan University
Shanghai Innovation Institute
xdpan@fudan.edu.cn

A Artifact Appendix

A.1 Abstract

This artifact supports the availability and functionality evaluation of *#2172* MATE*: Policy-Aware Security Auditing for Mobile Agents via Synthesis-Driven Trajectory Learning*. MATE is a lightweight, policy-conditioned auditor for mobile-agent trajectories: given a user instruction, an agent trajectory, and a natural-language security policy, it predicts whether the trajectory violates the policy and produces an explanatory rationale. The artifact contains the code, MATEBENCH benchmark data, trained MATE-0.5/1.5/3B model checkpoints, policy retriever, data synthesis and augmentation pipeline, trajectory adapter, recorded evaluation outputs, and documentation promised in the paper's Open-Science section.

The artifact supports availability, functionality, and results-reproduction evaluation. It provides a lightweight functionality workflow that starts a local MATE model server, builds a small MATEBench subset, runs the evaluator, and summarizes the resulting metrics. It also includes the released checkpoints, complete benchmark splits, full-evaluation commands, and recorded outputs for verifying the paper's primary MATEBench results. Full-benchmark evaluation requires additional GPU time and model-serving resources, so the appendix distinguishes lightweight functionality checks from full-scale reproduction workflows. (**Readers are encouraged to refer to the `README.md` file in the `MATE-Artifacts.zip` archive and follow its detailed instructions when performing validation.**)

A.2 Description & Requirements

A.2.1 Security, Privacy, and Ethical Concerns

The archive contains no API keys, tokens, real user accounts, or private credentials. The real-world MATEBENCH subset is released as sanitized structured trajectory data and textual screen summaries rather than private raw phone logs or unsanitized screenshots. Some benchmark examples describe security-relevant actions, such as credential handling or unsafe sharing, but they are synthetic or sanitized evaluation records rather than instructions for attacking real systems. Reviewers should run experiments in an isolated environment and supply their own model-serving credentials if they choose to use external LLM endpoints for synthesis or adapter workflows.

A.2.2 How to Access

The artifact is publicly available on Zenodo at https://doi.org/10.5281/zenodo.20232924. This DOI serves as the authoritative and stable access point for the artifact.

The Zenodo record provides an archive named `MATE-Artifacts.zip`. After downloading and extracting the archive, reviewers should begin with `README.md`, `ARTIFACT_MANIFEST.md`, and `claims_mapping.md`. These documents describe the scope and organization of the artifact, map the released components to the claims made in the paper, and provide the required setup and evaluation commands.

The archive is organized as follows. `MateBench/` contains the released benchmark datasets. `Mate_model/` contains the MATE-0.5B, MATE-1.5B, and MATE-3B checkpoints. `evaluation/` contains the evaluator, the vLLM launcher, and the recorded evaluation outputs. `policy_retriever/` contains the retriever model and the security-policy pools. `data_synthesis_pipeline/`,

trajectory_adapter/, and model_deployment/ contain the data-synthesis, trajectory-adaptation, and deployment utilities described in the paper.

A.2.3 Hardware Dependencies

Functionality and reproducibility evaluations require serving the released MATE-0.5B, MATE-1.5B, or MATE-3B checkpoint through vLLM or another OpenAI-compatible inference server. We recommend a Linux machine equipped with at least one NVIDIA H100 GPU. This configuration can serve all three released checkpoints and supports both the minimal subset test and full evaluation of the MATEBENCH splits. Reviewers may use the smaller MATE-0.5B or MATE-1.5B checkpoint for a faster functionality check. Full-benchmark evaluation requires additional runtime, and regenerating the synthetic data also requires access to an external or locally served LLM endpoint.

The largest directories are Mate_model/ (approximately 9.6GB), policy_retriever/ (approximately 2.1GB), and evaluation/ (approximately 2.5GB, including recorded outputs and the local rationale-embedding model). The MATEBENCH benchmark itself is approximately 21MB.

A.2.4 Software Dependencies

The artifact was prepared for Python 3.11. The base Python dependencies are listed in requirements.txt. vLLM is used only for local model serving and can be installed separately if reviewers run the minimal functionality test.

```
conda create -n mate python=3.11 -y
conda activate mate
pip install -r requirements.txt
pip install vllm
```

The evaluator uses an OpenAI-compatible chat-completions endpoint. The scripts set MATE_MODEL_NAME, MATE_BASE_URL, and MATE_API_KEY automatically for the minimal workflow. Data synthesis and trajectory-adapter scripts have their own endpoint configurations and may require external LLM services if reviewers regenerate data.

A.2.5 Benchmarks

MateBench/ contains 4,087 JSON samples: 2,775 MATEBENCH-IN samples, 1,150 MATEBENCH-OUT samples, and 162 MATEBENCH-REAL sanitized real-world samples. Each JSON file contains a user instruction, trajectory steps, a natural-language security policy, and a reference Violation/Category/Rationale label. The in-domain and out-of-domain splits cover synthetic trajectory-level auditing tasks, while MateBench-Real/ contains manually collected, sanitized mobile-agent trajectories.

A.3 Set-up

A.3.1 Installation

Unpack MATE-Artifacts.zip, enter the artifact directory, and install the base dependencies:

```
cd MATE-Artifacts
conda create -n mate python=3.11 -y
conda activate mate
pip install -r requirements.txt
```

A.3.2 Basic Test

Reviewers can verify the benchmark size and summarize recorded outputs:

```
find MateBench -name '*.json' | wc -l
python summarize_results.py \
  --root evaluation/evaluation_results
python summarize_results.py \
  --root evaluation/
    evaluation_results_subset
```

The expected MATEBENCH count is 4,087 JSON files. The summary commands should print Markdown tables with task, model, run identifier, sample count, accuracy, violation F1, rationale similarity, precision, and recall. The subset-result directory contains six MATE-1.5B runs generated by following the README workflow.

A.4 Evaluation Workflow

A.4.1 Major Claims

The artifact supports the following claims from the paper and Open-Science section.

1. The MATE auditing and evaluation code is released. Evidence: evaluation/, run_minimal_demo.sh, and run_eval_subset.sh.
2. MATEBENCH is released with synthetic in-domain, synthetic out-of-domain, and sanitized real-world subsets. Evidence: MateBench/.
3. Trained MATE checkpoints are released. Evidence: Mate_model/.
4. The policy retriever, policy pools, data synthesis pipeline, augmentation pipeline, and trajectory adapter are released. Evidence: policy_retriever/, data_synthesis_pipeline/, trajectory_adapter/, and model_deployment/.

5. Recorded outputs for the reported evaluation style are included. Evidence: `evaluation/evaluation_results/` and `evaluation/evaluation_results_subset/`.

A.4.2 Experiments

Experiment 1: availability inspection. Inspect the three primary reviewer-entry documents and verify the presence of the major artifact directories:

```
ls README.md ARTIFACT_MANIFEST.md
    claims_mapping.md
ls MateBench Mate_model policy_retriever
    evaluation
```

This experiment confirms that the promised code, datasets, models, policies, and recorded outputs are present in the archive for reviewer inspection.

Experiment 2: recorded-result summary. Run:

```
python summarize_results.py --root
    evaluation/evaluation_results
python summarize_results.py --root
    evaluation/evaluation_results_subset
```

The expected output is a compact Markdown table. The full recorded tree contains 18 run directories, and the subset tree contains 6 MATE-1.5B subset run directories. This experiment lets reviewers inspect the result format and metrics without rerunning model inference.

Experiment 3: subset construction without model serving. Run:

```
CREATE_ONLY=1 SUBSET_SIZE=2 MATE_TASK=in
    MATE_LANG=en \
  bash run_eval_subset.sh
```

The expected output is a temporary directory containing sampled MATEBench-In English files from the trajectory, multi-app, multi-policy, and no-match subdirectories. This is a lightweight check of dataset layout, sampling logic, and script behavior.

Experiment 4: minimal end-to-end functionality. On a machine capable of serving MATE-1.5B, run:

```
bash run_minimal_demo.sh
```

The script starts vLLM, waits for `/v1/models`, builds a temporary MATEBENCH-IN English subset, runs `evaluation/run_matebench.py`, and summarizes the result. The default configuration is `MODEL_PATH=Mate_model/MATE-1.5B`, `SERVED_MODEL=MATE-1.5B`, `PORT=8016`, `MATE_TASK=in`, `MATE_LANG=en`, `SUBSET_SIZE=50`, and `BATCH_SIZE=8`. The expected output is a new directory under `evaluation/evaluation_results_subset/` with `eval_results.txt` and `eval_model_output.json`.

If reviewers already have an OpenAI-compatible model endpoint running at `http://127.0.0.1:8016/v1`, they can skip server startup:

```
START_SERVER=0 MODEL_NAME=MATE-1.5B PORT
    =8016 \
  MATE_TASK=in MATE_LANG=en bash
    run_eval_subset.sh
```

If every sample reports `API error: Connection error`, the model endpoint is not reachable. First check:

```
curl http://127.0.0.1:8016/v1/models
```

In local runs, clients should normally connect through `127.0.0.1` or the server's real IP address. `0.0.0.0` is appropriate as a server bind address, but it is not the recommended client address.

A.5 Notes on Reusability

The artifact supports reuse beyond availability evaluation. Researchers can evaluate OpenAI-compatible auditors on MATEBENCH by serving a model endpoint and specifying its name in `evaluation/run_matebench.py`. The policy retriever and policy pools can attach app-specific security policies to new mobile-agent trajectories, while the trajectory adapter and deployment utilities convert raw agent logs into the canonical MATE format. The synthesis and augmentation code can generate policy-conditioned trajectories, although exact regeneration depends on the external LLM endpoint, model version, decoding settings, and API availability.

The release excludes API keys, real user accounts, private raw phone logs, and unsanitized screenshots. The real-world subset contains sanitized structured trajectories, preserving research utility for inspection and benchmarking while reducing privacy and platform risks.

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/.