USENIX Security '26 Artifact Appendix: FIRA: Enabling Automatic Forensic Investigation of Unmanned Aerial Vehicles

Yizhi Huang, David Oygenblik, Runze Zhang, Mingxuan Yao, Muhammad Ibrahim,
Burak Sahin, Haichuan Xu, Saman Zonouz, Brendan Saltaformaggio
Georgia Institute of Technology

A Artifact Appendix

A.1 Abstract

This artifact accompanies the FIRA paper and provides the complete pipeline for reproducing the forensic investigation and reconstruction of autonomous drone accidents. The artifact includes: (1) firmware causal graph extraction and validation for PX4 and autopilot, (2) neural network causality analysis via layer conductance attribution, (3) model reconstruction from mission-collected neuron data, and (4) evaluation of poisoned vs. clean models before and after online learning updates. All experiments, data, and pre-trained models are included and can be executed end-to-end via a single Python script or Docker container.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

The artifact includes pre-trained ML models (some intentionally poisoned for evaluation purposes) and drone mission telemetry data. All data collected from controlled test environments. No real accident data is included. The poisoned models are used only for demonstrate forensic capabilities and have no security risk to the evaluator's machine. The Docker container runs with default permissions and does not require elevated privileges beyond Docker access.

A.2.2 How to access

The archived artifact is available on Zenodo at: https://zenodo.org/records/17945698

Evaluators can also access a pre-configured evaluation machine via SSH:

```
ssh -i /path/to/key -p 10467
usenix_ae@0.tcp.ngrok.io
```

The SSH private key is provided separately on the HotCRP website.

A.2.3 Hardware dependencies

The experiments require a CUDA-capable GPU for neural network inference and attribution analysis. Our evaluation environment uses:

- 1× NVIDIA A6000 GPU (48 GB VRAM)
- 128 GB system RAM
- about 20 GB disk space

A pre-configured machine with the above hardware is accessible via the SSH.

A.2.4 Software dependencies

The artifact is packaged as a Docker container. The only host requirement is:

- Docker Engine with NVIDIA Container Toolkit

Alternatively, for running without Docker, the following are required:

- Ubuntu 20.04+ (or equivalent Linux distribution)
- Miniconda / Anaconda with Python 3.8
- PyTorch 2.x with CUDA 12.1 support
- Python packages: `matplotlib`, `captum`, `trojai`, `Levenshtein`, `networkx`, `tqdm`, `pandas`, `numpy`, `Pillow`

A.2.5 Benchmarks

All required data is included in the artifact:

- **Clean test images**: 1000+ drone camera images for autonomous landing classification (10 classes), located in `AE/clean_data/`.
- **Poisoned test images**: Adversarial variants of the test images, located in `AE/poison_data/`.

- **Pre-trained models**: Baseline MobileNetV2 model for autonomous landing (`pre_mission/model/auto_land/auto_land.pth`) and a poisoned variant (`during_mission/online_learning_model/auto_land/poison.pth`).
- **Mission-collected neuron data**: Pickle files containing critical neuron weights captured during simulated missions (`during_mission/mission_collected_data/auto_land/`).
- **Firmware causal graphs**: PX4 v5x extracted module causality data (`AE/v5x/BCG/`).

A.3 Set-up

A.3.1 Installation

Option A: Docker. Build and run the Docker container from the artifact root:

```
docker build -t fira_ae .
docker run --gpus all -it fira_ae
```

Option B: Manual installation. From the artifact root directory:

```
conda create -n FuDrone python=3.8 -y
conda activate FuDrone
pip install torch torchvision torchaudio \
--index-url https://download.pytorch.org/whl/cu121
pip install matplotlib captum trojai Levenshtein \
networkx --no-deps
pip install matplotlib captum Levenshtein networkx
```

A.3.2 Basic Test

Activate the environment and run the first experiment as a test:

```
conda activate FIRA
cd FIRA_AE/AE/v5x/BCG
python before_check.py
```

Expected output: the script prints the number of nodes in the system causal graph, the number of matched nodes in the ground truth, and an ED score, in Table 3.
e.g.:

```
Nodes in SYS: <N>
Matched in GT: <M>
ED: 0.XXX
```

If the script completes without errors and prints an ED score, the environment is correctly configured.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): FIRA can extract and validate firmware causal graphs from drone binaries, and the pruned causal graph retains high equivalence to the ground truth. This is supported by experiment (E1) (Table 3, rows 1-3 in the original paper).

(C2): FIRA can identify critical neuron groups in neural network models using layer conductance attribution, producing model causal graphs (MCGs) that accurately characterize model behavior. This is supported by experiment (E2) (Table 2, rows 1-3 in the original paper).

(C3): FIRA can reconstruct neural network models from collected neuron data transmitted during flight, and the reconstructed models have distinguishable accuracy differences between clean and poisoned inputs, enabling forensic identification of model poisoning. This is supported by experiments (Table 4, rows 1-4 in the original paper).

A.4.2 Experiments

All experiments can be run end-to-end via the Python script:

```
cd FIRA_AE/AE/
python3 run_all.py
```

Alternatively, each experiment can be run individually as described below.

(E1): Firmware Causal Graph Validation [1 minutes]:
Validates that the extracted PX4 v5x firmware causal graph matches the ground truth before and after pruning.
Preparation: Ensure the conda environment is activated and navigate to `FIRA_AE/AE/v5x/BCG/`.
Execution: `cd FIRA_AE/AE/v5x/BCG`
`python before_check.py`
`python prune_check.py`
Results: `before_check.py` reports the full graph SYS, GT, ED between the extracted firmware causal graph and ground truth. `prune_check.py` reports the SYS, GT, ED after pruning by topic list, along with any nodes identified but not found. Both scripts print node counts and ED scores to stdout.

(E2): Neural Network Causality Analysis (MCG Extraction) [10 minutes]:
Extracts model causal graphs for the autonomous landing model using layer conductance attribution at multiple granularities.
Preparation: Ensure GPU is available and navigate to `FIRA_AE/v5x/MCG/`.
Execution: `cd FIRA_AE/v5x/MCG`
`python auto_land.py`
Results: The script produces three JSON files (`140.json`, `280.json`, `420.json`) containing the

important neuron lists per category at different top-n configurations. For each configuration, the script reports execution time, weighted matching accuracy on the test set, and neuron sharing statistics across categories.

(E3): Model Reconstruction from Mission [2 minutes]: Reconstructs updated models by applying mission-collected neuron data to the baseline model.
Preparation: Navigate to `FIRA_AE/`.
Execution:
```
cd FIRA_AE/AE
python model_update.py
```
Results: Three updated model files are saved to `FIRA_AE/AE/updated_model/`: `updated_50.pth`, `updated_100.pth`, and `updated_150.pth`, corresponding to models updated with 50, 100, and 150 mission-collected shared neuron groups.

(E4): Poisoned vs. Clean Model Evaluation [5 minutes]: Evaluates the baseline clean model and the poisoned model on both clean and poisoned test datasets to establish baseline accuracy differences.
Preparation: Navigate to `FIRA_AE/AE/`.
Execution:
```
cd FIRA_AE/AE
python clean_model_evaluate.py
python poison_model_evaluate.py
```
Results: Each script prints clean and poisoned accuracy for the model and saves results to CSV files in `FIRA_AE/AE/evaluation_results/`. The clean model should show similar accuracy on both datasets, while the poisoned model should show divergent accuracy.

(E5): Updated Model Evaluation [10 minutes]: Evaluates the reconstructed (updated) models on clean and poisoned datasets to assess the effect of online learning with mission data.
Preparation: Ensure (E3) has been completed so that updated models exist in `FIRA_AE/AE/updated_model/`.
Execution:
```
cd FIRA_AE/AE
python updated_model_evaluate.py
python updated_clean_model_evaluate.py
```
Results: `updated_model_evaluate.py` evaluates the three updated models (from poisoned mission data) on both datasets. `updated_clean_model_evaluate.py` updates the clean model with clean neuron data and evaluates the result. Results are printed to stdout and saved to CSV. Comparing these with (E4) demonstrates how mission-collected neuron updates affect model behavior and how forensic analysis can distinguish poisoned from clean updates.

(E6): Model Attribution Analysis (Optional) [30-50 minutes]: Performs white-box pixel-level attribution analysis on the updated models to identify which input regions drive model decisions. This part is not the key contribution of the FIRA, but we provide the script to demonstrate the updated model compatible with existing white-box analysis tools.
Preparation: Ensure (E3) has been completed. Place analysis images in `AE/analysis_data/`.
Execution: When running via `run_all.py`, type `Y` at the prompt. Or run directly:
```
cd FIRA_AE/AE
python model_analysis.py
```
Results: The script produces `.npy` files in `FIRA_AE/AE/` for each updated model, containing pixel attribution results.

A.5 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/.