USENIX Security '25 Artifact Appendix: Chimera: Creating Digitally Signed Fake Photos by Fooling Image Recapture and Deepfake Detectors

Seongbin Park*, Alexander Vilesov*, Jinghui Zhang, Hossein Khalili,
Yuan Tian, Achuta Kadambi, Nader Sehatbakhsh
University of California, Los Angeles
*{parkseongbin,vilesov}@ucla.edu

A Artifact Appendix

A.1 Abstract

This paper introduces Chimera, an end-to-end attack strategy that crafts cryptographically signed fake images capable of deceiving both deepfake and image recapture detectors. Current adversarial and generative models fail to effectively deceive both detector types or lack generalization across different setups; Chimera addresses this gap by using a hardware-aware adversarial compensator to craft fake images that successfully bypass state-of-the-art detection mechanisms. This artifact demonstrates the effectiveness of Chimera fooling both types of detectors with a high success rate while having high visual quality (compared to the original real image). Our successful end-to-end attack on state-of-the-art detectors shows an urgent need for more robust detection strategies.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

Since our attack does not target anything other than the target deepfake and recapture detectors (which are provided), there is no significant risk to the evaluators.

A.2.2 How to access

The complete artifact, including source code, pretrained models, and instructions, is available on Zenodo.

A.2.3 Hardware dependencies

None. However, using GPUs will speed up testing.

A.2.4 Software dependencies

The artifact should be evaluated in a conda environment. A complete list of software dependencies is provided in the `environment.yml` file, which includes PyTorch and other required dependencies. Additionally, the dataset and model weights used in the experiments are linked in the GitHub repository and must be downloaded.

A.2.5 Benchmarks

Pretrained models based on the datasets are provided in the conda environment. There is no need to download/access additional datasets.

A.3 Set-up

Follow these steps to prepare the environment for evaluation:

1. **Set up the Conda Environment:** Create and activate the conda environment using the provided `environment.yml`:

   ```
   conda env create -f environment.yml
   conda activate chimera
   ```

2. **Download the Dataset and Models:** The dataset is hosted on Zenodo. Please download it from the provided link (see the README). The dataset can be unzipped anywhere in the main directory; for the appropriate location of the models, refer to the next step.

3. **Configure Deepfake Detection:**

   - Edit `deepfake-detection/dataset_paths.py` to specify the correct paths to your dataset.
   - Move the following model weights into the `deepfake-detection/pretrained_weights` directory:
     - `fatformer_4class_ckpt.pth`
     - `fc_weights.pth`
     - `ViT-L-14.pt`

4. **Configure Recapture Detection:** Place the recapture detection model files into a directory of your choice. The paths to these models will be specified via command-line arguments during testing.

*These authors contributed equally to this work

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): *Chimera* can successfully bypass otherwise accurate **recaptured image** detectors. This is proven by the experiment (E1) described in Section 6.2 whose results are reported in Table 1.

(C2): *Chimera* can successfully bypass otherwise accurate **deepfake** detectors. This is proven by the experiment (E2) described in Section 6.3 whose results are reported in Table 4.

A.4.2 Experiments

(E1): *[Recapture Detection Experiment] [10 human-minutes + 10 compute-minutes]*

Preparation: No further preparation must be done from the steps outlined in A.3.

Execution: In the `recapture-detection` directory, execute the following command:

```
python main.py \
    --test \
    --test_path TEST_PATH \
    --config CONFIG \
    --test_raw_dirnames RAW_DIRNAMES \
    --test_recap_dirnames RECAP_DIRNAMES
```

where `TEST_PATH` is the path to the model and `CONFIG` is the path to the configuration file. Ensure that the configuration file matches the model being tested.
Specify the raw image directory with `RAW_DIRNAMES` and the recaptured image directory with `RECAP_DIRNAMES`. In the tar file provided, the raw images are in `stylegan2_orig`, benign recaptured images in `recap_mac`, and adversarial recaptured images in `attack_mac`.

Results: Upon completion, the script should output the accuracy of the models for the specified datasets.

(E2): *[Deepfake Detection Experiment] [2 human-minutes + 5 compute-minutes]*

Preparation: No further preparation must be done from the steps outlined in A.3.

Execution: In the `deepfake-detection` directory, run the test script:

```
./test.sh
```

This script runs all deepfake detection tests sequentially.

Results: The script saves the evaluation results (including detection metrics and success rate analyses) in the `deepfake-detection/results` directory.

A.5 Notes on Reusability

Though our method is generalizable to any hardware setup, the adversarial generative model must be trained specifically for each setup.

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/usenixsec2025/.