False Claims against Model Ownership Resolution

Jian Liu and Rui Zhang, *Zhejiang University;* Sebastian Szyller, *Intel Labs & Aalto University;* Kui Ren, *Zhejiang University;* N. Asokan, *University of Waterloo & Aalto University*

https://www.usenix.org/conference/usenixsecurity24/presentation/liu-jian

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 33rd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

USENIX Security '24 Artifact Appendix: <False Claims against Model Ownership Resolution>

Jian Liu[*][†]
Zhejiang University
jian.liu@zju.edu.cn

Rui Zhang[†]
Zhejiang University
zhangrui98@zju.edu.cn

Sebastian Szyller
Intel Labs & Aalto University
contact@sebszyller.com

Kui Ren
Zhejiang University
kuiren@zju.edu.cn

N. Asokan
University of Waterloo & Aalto University
asokan@acm.org

A Artifact Appendix

A.1 Abstract

In this paper, we show that common MOR schemes in the literature are vulnerable to a different, equally important but insufficiently explored, robustness concern: a *malicious accuser*. We show how malicious accusers can successfully make *false claims* against *independent* suspect models that were not stolen. Our core idea is that a malicious accuser can deviate (without detection) from the specified MOR process by finding (transferable) adversarial examples that successfully serve as evidence against independent suspect models. To this end, we first generalize the procedures of common MOR schemes and show that, under this generalization, defending against false claims is as challenging as preventing (transferable) adversarial examples. Via systematic empirical evaluation, we show that our false claim attacks always succeed in MOR schemes that follow our generalization. In detail, the MORaccs achieved by transferable adversarial examples can be higher than the thresholds of MOR schemes that follow our generalization. For Adi, Li(b), and Lukas, we generate untargeted adversarial examples and evaluate the MORacc of these adversarial examples on independent models; For EWE and DAWN, we use targeted adversarial examples; For DI, we use perturbed examples to falsely claim the ownership of independent models.

A.2 Description & Requirements

This is a Pytorch implementation of the paper False Claims against Model Ownership Resolution.

A.2.1 Security, privacy, and ethical concerns

For experiments on CelebA, we use the real-world Amazon Rekognition API as the independent model. Generating false claims on this model may raise ethical concerns.

A.2.2 How to access

Our code is at `https://github.com/ssg-research/Falseclaims/releases/tag/v0.0.2_ae`

A.2.3 Hardware dependencies

We evaluate our methods on Ubuntu 20.04.2 LT with one NVIDIA-V100 GPU. One GPU is required for this implementation.

A.2.4 Software dependencies

We evaluate our methods on Ubuntu 20.04.2 LT with one NVIDIA-V100 GPU. We evaluate our methods with Pytorch 2.2.1 with CUDA 11.8 that supports GPU computation. And we using Python 3.11.8. Other dependencies are listed in `https://github.com/ssg-research/Falseclaims/blob/main/requirements.txt`*.*

A.2.5 Benchmarks

- CIFAR10: This dataset will be downloaded automatically.
- ImageNet: Since we use 10 classes of ImageNet dataset, you can download it from `https://drive.google.com/drive/u/1/folders/1h1NcupuTF76XOdOY-CZWR_XMe4GJdmWx` named "`small_imagenet.zip`". Save the "`small_imagenet`" to "`~/data/`"

A.3 Set-up

1. Clone the GitHub repository and install the requirements.

[*] Co-first authors; Jian Liu is the corresponding author.

2. Download the DNN models we will use from https://drive.google.com/drive/u/1/folders/1h1NcupuTF76XOdOY-CZWR_XMe4GJdmWx The files in the uploaded link are as follows,
    - FalseClaims
        - cifar10.tar: the DNN models for CIFAR10 dataset.
        - defences.tar: the embeddings of DI method we will use later
        - imagenet.tar: the DNN models for ImageNet dataset
        - `small_imagenet.zip`: the subset of ImageNet dataset we use to train the DNN models.
3. Create a new directory named `checkpoint`, then extract cifar10.tar and imagenet.tar into `checkpoint/`.
4. Extract defences.tar in the repository's directory.
5. Save the `"small_imagenet"` to `"~/data/"`

In the repository `Falseclaims/`, besides the files clone directly from GitHub, it should looks like:

- `Falseclaims/`
    - `checkpoint/`
        - `cifar10/`
            - `cifar_wide_resnet/`
            - `ind/`
            - `victim/`
        - `imagenet/`
            - `ind/`
            - `resnet18_imagenet/`
            - `victim/`
    - `defences/`
        - `cifar10/`
        - `imagenet/`

A.3.1 Installation

For gpu version of pytorch and the dependencies should be installed according to their official instructions based on different machine. Therefore, these packages are not included in our appendix but they are necessary.

```
pip install -r requirements.txt
```

A.3.2 Basic Test

After you have downloaded the DNN models we provided.

```
python attack.py
```

If it runs successfully, it will display logs indicating that the models have been loaded successfully. Subsequently, it will show the progress of generating transferable adversarial examples for our false claims.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): *False claims are effective against the mixed thresholds. This is proven by experiments (E1)(E2)(E3)(E4) described in Section 5.2 whose results are illustrated in Section 6.2, Table 3,4 in our paper.*

A.4.2 Experiments

(E1): *[False Claim MORacc] [30 human-minutes + 3 compute-hour + 10GB disk]: Evaluate the MORacc for untargeted adversarial examples to false claim Adi[1], Li(b)[34], and Lukas[39].*
How to: *To conduct the experiments, you can use the commands provided in the README.md in GitHub. For each experimental setting (different structure & different data; different structure & same data; same structure & different data), the output will include the MORacc for 5 independent models, respectively.*
Execution: *The commands for our three settings, applicable to CIFAR10 and ImageNet, are available in the README file of the GitHub repository*
Results: *After each run, it will output the MORacc for 5 independent models in the terminal.*

(E2): *[False Claim MORacc] [30 human-minutes + 3 compute-hour + 10GB disk]: Evaluate the MORacc for targeted adversarial examples to false claim EWE[24] and DAWN[59].*

Execution: *The commands are the same as those for (E1), requiring only the addition of*

```
--adv\_target True
```

at the end of each command. An example is provided in the README.

Results: *After each run, it will output the MORacc for 5 independent models in the terminal.*

(E3): *[False Claim MORacc] [1 human-minutes + 1 compute-minute]: Evaluate the MORacc of perturbed examples for DI[43].*

How to: *For CIFAR10, utilize the* `DI_cifar10.ipynb` *Jupyter notebook, and for ImageNet, refer to* `DI_imagenet.ipynb`*. It is essential to modify the* `root_path` *and* `params_path` *within the notebooks to correspond with the directory of the embeddings. Specifically, for the embeddings downloaded form the link we provided, the paths for CIFAR10 and ImageNet should be*

```
~/Falseclaims/defences/cifar10/DI/files
```

for CIFAR10 and

```
~/Falseclaims/defences/ImageNet/DI/files
```

Results: *For CIFAR10, the output includes the normalized effect size of DI for both normal evaluations and our false* *claim evaluation. The term* `independent` *corresponds to the model trained on CIFAR10 official evaluation set from pytorch, which is same with that in DI. In our paper, the 'independent threshold' is determined by the highest results of the models trained by the separated training set* (`ind`, `suspect`, `suspect_same_struct`, `suspect_same_data`). *And the three settings we consider correspond to the normalized effect size of* `suspect_adv`, `suspect_same_struct_adv`, `suspect_same_data_adv`. *For ImageNet, the three settings we consider corresponds to the normalized effect size of* `suspect_adv`, `suspect_same_struct_adv`, `suspect_same_data_adv`.

(E4): *Train models*:

Execution: *For (E1) and (E2), the training commands are provided in the README. It is important to append*

```
--ind_resume False --inds_resume False
--vic_resume False
```

to the aforementioned commands to execute our false claim attack while training independent models and the victim model.

For (E3), move the corresponding model to

```
defences/{dataset}/DI/models/CIFAR10/
model_{model_name}/.
```

For example, move the victim model of CIFAR10 to

```
defences/cifar10/DI/models/CIFAR10/
model_teacher/final.pt
```

Then run the following command

```
python defence/DI.py
```

The process will generate perturbed samples and save them to `model_teacher`, as well as generate the embeddings for the `model_teacher`. Subsequently, you should move the other targeted models to the specified directory. Following this, execute the commands below to generate embeddings for the corresponding models.

```
python defence/Di.py
-c configs/cifar10/train/resnet_di.yaml
--model_id [name of the targeted file]
--adv_gen False
```

Finally, modify the file used in the `.ipynb` and get the outputs. For ImageNet, change the command as follows,

```
python defence/Di.py
-c configs/imagenet/train/resnet_di.yaml
```

and

```
python defence/Di.py
-c configs/imagenet/train/resnet_di.yaml
--model_id [name of the targeted file]
--adv_gen False
```

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/usenixsec2024/.