USENIX Security '24 Artifact Appendix: MODELGUARD: Information-Theoretic Defense Against Model Extraction Attacks

Minxue Tang[1], Anna Dai[1], Louis DiValentin[2], Aolin Ding[2], Amin Hass[2], Neil Zhenqiang Gong[1], Yiran Chen[1], Hai "Helen" Li[1]
[1]Department of Electrical and Computer Engineering, Duke University
[2]Cyber Security Lab, Accenture
[1]{minxue.tang, anna.dai, neil.gong, yiran.chen, hai.li}@duke.edu
[2]{louis.divalentin, a.ding, amin.hassanzadeh}@accenture.com

A Artifact Appendix

A.1 Abstract

This appendix will introduce the roadmap to evaluate the artifact and reproduce the major results in the paper *ModelGuard: Information-Theoretic Defense Against Model Extraction Attacks*. Specifically, this appendix will introduce how to use the source codes available on GitHub: https://github.com/Yoruko-Tang/ModelGuard/releases/tag/v1.0, including environment installation, dataset preparation, script execution, and result collection.

A.2 Description & Requirements

This section includes necessary information for recreating the experimental setup used in the paper, including how to access the source codes, hardware and software requirements, and how to prepare the required datasets.

A.2.1 Security, privacy, and ethical concerns

Since we are simulating the attack and defense on a single machine, evaluating this artifact will not lead to security, privacy, or ethical issues.

A.2.2 How to access

Use the following link to access our code on GitHub: https://github.com/Yoruko-Tang/ModelGuard/releases/tag/v1.0. You can download the project and unzip it without Git on your machine, or simply use the "git clone" command to clone the project with Git on your machine.

A.2.3 Hardware dependencies

We recommend evaluating our code with a machine running Linux OS. The machine should be equipped with at least 128GB RAM and a GPU that supports CUDA and has at least 24GB GPU memory. We used a machine running Linux 5.15.0 with two Intel Xeon Gold 6254 CPUs (36 cores, 72 threads, and 1.5TB Memory in total) and one NVIDIA TITAN RTX GPU (CUDA 10.2 and 24GB Memory) to get all the results in the paper.

A.2.4 Software dependencies

The code is tested with Python 3.8. We require some standard libraries such as Pytorch, Numpy, PuLP, etc. You can follow the instructions in the "readme" file to install all the software dependencies.

A.2.5 Benchmarks

We require 8 datasets to reproduce the main result in the paper: SVHN, CIFAR10, CIFAR100, TinyImageNet200, Indoor67, CUB200, Caltech256, and ImageNet1k. You need to download **all** the datasets before running any codes. Please follow the instructions in the "readme" file to download all the datasets and unzip them in "./data/".

A.3 Set-up

This section will introduce how to set up the environment for running experiments quickly.

A.3.1 Installation

Please follow the following steps to install the software dependencies on your machine.

1. CUDA Installation You need to make sure that a CUDA Toolkit with a version later than 10.2 is properly installed on your machine. Please check the official website of CUDA Toolkit for instructions: https://developer.nvidia.com/cuda-downloads.

2. Anaconda Installation You need to install Anaconda in order to install the other Python packages easily. Please

check the official website of Anaconda for instructions: https://www.anaconda.com/download/.

3. **Python Package Installation** You need to install all the required packages following the instructions in the "readme" file of the repository. Note that different GPUs may require different Pytorch releases. If you are unable to use the default release after creating the environment following the instructions in "readme", please install a proper version of Pytorch following the instructions in the official website: https://pytorch.org/get-started/previous-versions/. Note that our code requires PyTorch 1.7.

A.3.2 Basic Test

Use the following commands in shell to check if Python, Pytorch, CUDA and PuLP are properly installed.

```
python
import torch
import pulp
print(torch.cuda.is_available())
```

If no error is reported and the final output is "True", then all the dependencies are installed properly.

A.4 Evaluation workflow

This section will introduce how to run the codes and reproduce the major results (Table 2 to Table 5) in the paper.

A.4.1 Major Claims

You should be able to reproduce the results from Table 2 to Table 5, which supports the following major claims:

(C1): pBayes Attack outperforms other attacks. This is proven by comparing the results of different attacks collected in (E1).

(C2): MODELGUARD-S outperforms other defenses when defending against strong adaptive attacks while maintaining high utility. This is proven by comparing the results collected in (E1) and (E2).

A.4.2 Experiments

Please complete the following steps to verify the major claim in the last section.

(E0): [Run scripts] [10 human-minutes + **800 compute-hour** + 100GB disk]: Run the Python scripts provided in the "/scripts/" directory of the repository, one for each table.
Preparation: Make sure all the dependencies are installed properly, as introduced in previous sections.
Execution: Run all four scripts in the shell window. For example, using the following command will run all the experiments in Table 2 (X_t =Caltech256, w_t =resnet50, X_q =ImageNet1k):

```
python scripts/run_caltech256.py
```

You can change the device for running the experiments by editing "dev_id" in the scripts (Line 7) if you have multiple GPUs on your machine. You must keep the shell window alive during the process.
Results: The results of each pair of attack and defense methods are stored in the following directory: "./models/final_bb_dist/[X_t]-[w_t]/[query strategy]_ [attack strategy]-[X_q]-B50000/[defense method]".
For example, when X_t is Caltech256, the results of KnockoffNet (random) with pBayes Attack (bayes) against ModelGuard-S ($\varepsilon = 1.0$) are stored in "./models/final_bb_dist/Caltech256-resnet50/random_bayes-ImageNet1k-B50000/modelguards/eps1.0".
For JBDA-TR, the directory is "./models/final_bb_dist/Caltech256-resnet50/jbtr3_bayes-ImageNet1k-B50000/modelguards/eps1.0".
Notice that only AM and RevSig are evaluated against Top-1 Attack because the other defenses do not change the top-1 label and the results are the same as that of the Top-1 Defense against Naive Attack. In addition, None Defense is not evaluated against D-DAE, D-DAE+ and pBayes attacks because the results of these strong adaptive attacks against no defense should be the same as that of the Naive Attack against no defense, as we allow them to know all the details of the defense method.

(E1): [Collect substitute model accuracy/fidelity] [1 human-hour + 0 compute-hour]: Collect the accuracy and fidelity of the substitute model for each pair of attack and defense methods from the results generated in (E0). This will give you the results in the first 14 rows of each table.
How to: A "train.50000.log.tsv" (KnockoffNet) or "train.50002.log.tsv" (JBDA-TR) file in each directory generated in (E0) logs the test accuracy and fidelity along the whole training procedure. You can get the best accuracy/fidelity in the last column of the last row in this file for the corresponding pair of attack and defense. After getting the results of all attacks against one defense, you can get the maximal accuracy and fidelity among all attacks as the "Max Accuracy of w_s" and "Max Fidelity of w_s" for this defense method.

(E2): [Collect target model utility] [1 human-hour + 0 compute-hour]: Collect the max ℓ_1 distortion and the protected accuracy of the target model for each defense method from the results generated in (E0). This will give you the results in the last 2 rows of each table in the paper. Notice that these results are irrelevant to the attack method and are determined by the query dataset X_q and the defense method.
Max ℓ_1 Distortion: A "distancetransfer.log.tsv" file logs the perturbation statistics when querying. You can get the max ℓ_1 distortion in the second column of the

last row in this file for the corresponding defense.

Protected Accuracy: A "bboxeval.xxxxx.log.tsv" file logs the protected accuracy of the given defense, where "xxxxx" is the size of the test set (for example, 6400 for Caltech256). You can get the test accuracy of the protected model in the last column of this file for the corresponding defense.

A.5 Notes on Reusability

The scripts provided in the repository can only run experiments on predefined settings. While completing all the experiments may take a very long time (over 200 hours for each script), you can skip some attacks and defenses by editing the "query_list" (Line 40), "attack_list" (Line 41), and "defense_list" (Line 42) in the scripts. For example, you can only run (KnockoffNet) + (Naive Attack, pBayes Attack) against (None Defense, ModelGuard-W, ModelGuard-S) by using:

```
query_list = ['random']
attack_list = ['naive','bayes']
defense_list = ['none',
                'modelguard_w',
                'modelguard_s']
```

which will take less than 20 hours for each script. We also encourage you to explore more customized running options as instructed in the "readme" file.

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/usenixsec2024/.