Towards A Proactive ML Approach for Detecting Backdoor Poison Samples

Xiangyu Qi, Tinghao Xie, Jiachen T. Wang, Tong Wu, Saeed Mahloujifar, and Prateek Mittal, *Princeton University*

https://www.usenix.org/conference/usenixsecurity23/presentation/qi

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 32nd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

USENIX'23 Artifact Appendix: Towards A Proactive ML Approach for Detecting Backdoor Poison Samples

Xiangyu Qi
Princeton University

Tinghao Xie
Princeton University

Jiachen T. Wang
Princeton University

Tong Wu
Princeton University

Saeed Mahloujifar
Princeton University

Prateek Mittal
Princeton University

A Artifact Appendix

A.1 Abstract

This artifact is mostly based on PyTorch, requiring GPU support. We implemented the proposed defense, Confusion Training (Algorithm 1) of our paper "Towards A Proactive ML Approach for Detecting Backdoor Poison Samples", together with a diverse set of baseline defenses and attacks. The artifact can reproduce our major experimental results (true positive rate, false positive rate, clean accuracy and attack success rate) reported in the main body of the paper. Our source code is available at https://github.com/Unispac/Fight-Poison-With-Poison, with a detailed guide at https://github.com/Unispac/Fight-Poison-With-Poison/blob/master/misc/reproduce.md.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

None. All backdoor attacks against DNNs in our artifact are conducted on the simulation level, therefore do not lead to any damage in the real world.

A.2.2 How to access

Our artifact source code is hosted at a GitHub repository, available through https://github.com/Unispac/Fight-Poison-With-Poison. For artifact evaluation purposes, this commit is used: https://github.com/Unispac/Fight-Poison-With-Poison/tree/b9ef34d

A.2.3 Hardware dependencies

This artifact minimally requires a Linux server with 300 GB disk storage, 10 GB RAM, 4 CPU cores, and 2 Nvidia GPUs (we use A100 in our experiments).

A.2.4 Software dependencies

This artifact relies on multiple existing Python packages, including Python, PyTorch, scipy and so on (details in requirement.txt). To reproduce only results of our proposed defense (Confusion Training), you may maually install PyTorch following their official guide and all other packages with pip. To produce results of other baseline defenses (specifically, Frequency and SPECTRE), you may also need to manually install Tensorflow (refer to official guide) and Julia (refer to other_cleansers/spectre/README.md). Our guide includes all details to set up the required software dependencies.

A.2.5 Benchmarks

Our experiments with this artifact are reported on 4 benchmark datasets: CIFAR10 (a 10-class common image classification task), GTSRB (a 43-class traffic sign recognition task), ImageNet (a 1000-class standard image classification task), and Ember (a malware classfication task). Among them, CIFAR10 and GTSRB are automatically downloaded and set up in our artifact, and a detailed guide to download and set up ImageNet and Ember datasets is available at misc/reproduce.md#todo-before-you-start.

A.3 Set-up

A.3.1 Installation

Our documentation contains a detailed guide to install our artifact and required environments. Briefly, the installation procedure is as follows:

1. Clone artifact from https://github.com/Unispac/Fight-Poison-With-Poison/tree/f2f02c2.
2. Install PyTorch following the official guide.
3. Install other Python packages via `pip install -r requirement.txt`.

4. (Optional) Install Tensorflow [guide] and Julia [guide].
5. Download ImageNet [link] and Ember [link] datasets. Refer to [link] for more details to set them up properly.
6. Execute command `python create_clean_set.py -dataset=$DATASET -clean_budget=$N` (where `$DATASET = cifar10, gtsrb, imagenet, ember`, $N = 2000 for cifar10 and gtsrb, $N = 5000 for imagenet and ember) to initialize the datasets.
7. Run `data/cifar10/clean_label/setup.sh` to setup data for clean label (CL) attack.
8. Download pretrained models (for Dynamic attack) `all2one_cifar10_ckpt.pth.tar` [link] and `all2one_gtsrb_ckpt.pth.tar` [link] to `models/`.

A.3.2 Basic Test

We provide a simple example (defending against BadNet attack on CIFAR10) involving our whole artifact pipeline (poisoning, training, defense, retraining, etc.) in our detailed guide at misc/reproduce.md. Briefly, one may test our artifact's core functionalities via:

1. Create a BadNet poisoned dataset by running `python create_poisoned_set.py -dataset=cifar10 -poison_type=badnet -poison_rate=0.01`.
2. Train on the poisoned dataset by running `python train_on_poisoned_set.py -dataset=cifar10 -poison_type=badnet -poison_rate=0.01`. The output should include the clean accuracy (ACC) and attack success rate (ASR) of the backdoor model in each training epoch.
3. Launch our Confusion Training defense by running `python ct_cleanser.py -dataset=cifar10 -poison_type=badnet -poison_rate=0.01 -devices=0,1 -debug_info`. The last couple lines of the output should include the defense results (`recall` and `fpr`).
4. Retrain on the cleansed training set by running `python train_on_cleansed_set.py -cleanser=CT -dataset=cifar10 -poison_type=badnet -poison_rate=0.01`. The output should include the clean accuracy (ACC) and attack success rate (ASR) of the defended model in each training epoch.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): Confusion Training is an effective approach to identify and remove poisoned samples in the training set. Compared to other baseline defenses, Confusion Training consistently shows better robustness. This is proven by the experiments (E1) in Sec 5.2 whose results are reported in Table 1 and Table 2.

(C2): Confusion Training is generalizable to larger dataset and extends beyond the vision domain. This is proven by the experiments (E2) and (E3) in Sec 5.3 whose results are reported in Table 4 and Table 5.

A.4.2 Experiments

(E1): [Major Experiments on CIFAR10 and GTSRB] [30 human-minutes + 100 compute-hour + 10GB disk]: Experiment (E1) evaluates and compares Confusion Training's effectiveness on CIFAR10 and GTSRB across 11+9 attacks with 11 baseline defenses, therefore proving our first claim (C1). Experiment (E1) corresponds to our reported results in Table 1 and Table 2.
How to: *All the preparation steps have been stated in Artifact Appendix A.3. We provide all necessary commands to reproduce our results of (E1) in misc/reproduce.md#major-results-on-cifar10-and-gtsrb-table-1–table-2.*

(E2): [Experiments on ImageNet] [1 human-hour + 160 compute-hour + 300GB disk]: Experiment (E2) evaluates Confusion Training's effectiveness on ImageNet, a larger vision dataset, and therefore provides support to our second claim (C2). Experiment (E2) corresponds to our reported results in Table 5.
How to: *All the preparation steps have been stated in Artifact Appendix A.3. We provide all necessary commands to reproduce our results of (E2) in misc/reproduce.md#imagenet-table-5.*

(E3): [Experiments on Ember] [1 human-hour + 3 compute-hour + 50 GB disk]: Experiment (E3) evaluates Confusion Training's effectiveness on Ember, a malware classification task, also providing support to our second claim (C2). Experiment (E3) corresponds to our reported results in Table 4.
How to: *All the preparation steps have been stated in Artifact Appendix A.3. We provide all necessary commands to reproduce our results of (E3) in misc/reproduce.md#ember-table-4.*

The expected outcome for the experiments above should be close to our reported results. Our experiments involve randomness in nature. Thus, the outcomes may have some variances. Our table also reports the approximate standard deviation of each result.

A.5 Notes on Reusability

We have already incorporated this artifact into a more comprehensible backdoor research toolbox, available at https://github.com/vtu81/backdoor-toolbox, which will be constantly maintained in the foreseeable future.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20220926. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2023/.