UBA-Inf: Unlearning Activated Backdoor Attack with Influence-Driven Camouflage

Zirui Huang, Yunlong Mao, and Sheng Zhong, *Nanjing University*

https://www.usenix.org/conference/usenixsecurity24/presentation/huang-zirui

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

Open access to the Artifact Appendices to the Proceedings of the 33rd USENIX Security Symposium is sponsored by USENIX.

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE | ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL | ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

USENIX Security ’24 Artifact Appendix: <UBA-Inf: Unlearning Activated Backdoor Attack with Influence-Driven Camouflage>

Zirui Huang
Nanjing University

Yunlong Mao*
Nanjing University

Sheng Zhong
Nanjing University

A Artifact Appendix

A.1 Abstract

This artifact includes the core algorithm for UBA-Inf camouflage generation, along with codes for model training and unlearning. We provide a shell script for environment setup and include a comprehensive `README.md` with detailed instructions. A demo is included to facilitate understanding and reproduction of experiments, covering dataset preparation, camouflage generation, and obtaining pre-unlearning and post-unlearning models.

A.2 Description & Requirements

Our experiment utilizes Conda version 4.12.1 and features 4 NVIDIA GeForce RTX 3090 GPUs. Essential prerequisites include Python $\geq$ 3.8.19, PyTorch $\geq$ 2.3.1+cu121, TorchVision $\geq$ 0.18.1+cu121, and OpenCV $\geq$ 4.5.5.

The training and backdoor generation codes are referenced from BackdoorBench [1].

A.2.1 Security, privacy, and ethical concerns

There are no security, privacy, or ethical concerns.

A.2.2 How to access

This artifact is accessible from the following URL: `https://github.com/Huangzirui1206/UBA-Inf/releases/tag/v1.0`.

A.2.3 Hardware dependencies

Our experiment employs Conda version 4.12.1 and utilizes four NVIDIA GeForce RTX 3090 GPUs, each with approximately 25GB of VRAM, leveraging CUDA version 12.1. Additionally, each GPU requires around 5GB of memory space. We assert that this artifact is compatible with other suitable GPU devices as well.

*Corresponding author, email: maoyl@nju.edu.cn.

A.2.4 Software dependencies

Essential prerequisites include Python $\geq$ 3.8.19, PyTorch $\geq$ 2.3.1+cu121, TorchVision $\geq$ 0.18.1+cu121, and OpenCV $\geq$ 4.5.5.

For datasets, CIFAR10 [5], MNIST [6] and Tiny-ImageNet [8] can be downloaded online automatically through PyTorch directly during training. For GTSRB [7], evaluators may have to download it manually.

There model architectures are evaluated in the original paper, including PreActResNet-18 [2], VGG-16 [3] and ResNet34 [4]. All models used in this artifact is self-contained.

A.2.5 Benchmarks

Four datasets are required as benchmarks, namely CIFAR-10 [5], MNIST [6], GTSRB [7] and Tiny-ImageNet [8].

A.3 Set-up

We provide some shell scripts to automate the evaluation.

A.3.1 Installation

To streamline setup, we provide a script for automatic environment configuration:

```
conda create -n uba-inf python=3.8
conda activate uba-inf
sh ./sh/install.sh
sh ./sh/init_folders.sh
```

A.3.2 Basic Test

Basic tests can be directly run by:

```
sh ./sh/demo.sh
```

For more detailed and comprehensive evaluations and demonstrations, a `./demo` folder is availble. You can go to `README.md` for more information.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): *UBA-Inf effectively camouflages the backdoor effect before unlearning, achieving a low Attack Success Rate (ASR) to ensure stealthiness. This is proven by the experiment (E1 and E3) described in Section 5.2 whose results are reported in Table 3 and Table 4.*

(C2): *The UBA-Inf backdoor effect can be effectively activated through unlearning, leading to a high ASR. UBA-Inf is versatile, being applicable to various backdoor triggers and unlearning algorithms. This is proven by the experiment (E2 and E3) described in Section 5.2 whose results are reported in Table 3 and Table 4.*

A.4.2 Experiments

(E1): *[Camouflage effectiveness] [2 human-minutes + 1 compute-hour]: This experiment generates the dataset with backdoor and camouflage samples and trains the pre-unlearning model to evaluate the camouflage effect of UBA-Inf. The injection ratio of backdoor samples and camouflage samples are demonstrated in Table 13.*

How to:

Preparation: *Prepare the dataset and construct UBA-Inf camouflage samples before training. Run* `python ./attack/badnet.py ...` *and* `python python ./uba/uba_inf_cover.py ...` *as presented in* `README.md`*.*

Execution: *Train the model by running* `python ./uba/perturb_attack.py` *as presented in the* `README.md`*.*

Results: *The results can be found in the result folders in* `./record`*.*

Tips: *In some cases, training models may cost a lot of time and resources. For quick evaluation, some pretrained models results and intermediate results are available on cloud:* `https://transfer.pcloud.com/download.html?code=5ZubYP0Zdi1h4h7kDlJZQzHk7ZxTT766WonFRKP9xDRvv3ijjpdFXX` *and* `https://transfer.pcloud.com/download.html?code=5ZRhYP0Zdi1h4h7kDlJZQzHk7ZAxCy8MJSNfRSM718DTBnSLmppt3k`*. More demos are available in* `./demo`*. You can see more details in the* `READDME.md` *file in the repo.*

(E2): *[Activation effectiveness] [2 human-minutes + 1 compute-hour]: This experiment simulates machine unlearning by full retraining and obtains the post-unlearning model to evaluate the camouflage effect of UBA-Inf.*

How to:

Preparation: *Evaluators can reuse the datasets generated in (E1).*

Execution: *Train the model by running* `python ./uba/perturb_attack.py` *as presented in the* `README.md` *but with parameter* `c_num` *equal to nil. This experiment evaluate the UBA-Inf effectiveness with full retraining as unlearning.*

Results: *The results can be found in the result folders in* `./record`*.*

(E3): *[SISA evaluations] [2 human-minutes + 4 compute-hour]: This experiment simulates machine unlearning by SISA [9] and obtains both the pre-unlearning and post-unlearning SISA models. The injection ratio of backdoor samples and camouflage samples are demonstrated in Table 13.*

How to:

Preparation: *Evaluators can generate the datasets as described in (E1).*

Execution: *You first need to construct the backdoor dataset* `python ./attack/badnet.py --yaml_path ../config/attack/prototype/cifar10.yaml --save_folder_name badnet_dataset_sisa_3 --add_cover 1 --epoch 00 --pratio 0.012 --cratio 0.006 --attack_target 6`*. The construct UBA-Inf camouflages with* `python ./uba/uba_inf_cover.py --dataset_folder ../record/badnet_dataset _sisa_3 --device cuda:3 --recursion_depth 50 --r_averaging 1 --ft_epoch 60 --ap_epochs 6`*. Train the model by running* `python ./uba/perturb_attack_sisa.py` *as presented in the* `README.md`*. This experiment obtains both the pre-unlearning and post-unlearning SISA models for effectiveness evaluation.*

Results: *The results can be found in the result folders in* `./record`*.*

For your reference, after executing `python ./attack/badnet.py ...` to construct the datasets, you should see outputs similar to those shown in Figure 1. After running `python ./uba/uba_inf_cover.py ...` to construct the camouflage, the outputs will resemble those in Figure 2. Finally, executing `python ./uba/perturb_attack.py ...` to train the pre-training or post-training models will produce outputs like those displayed in Figure 3.

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/`.

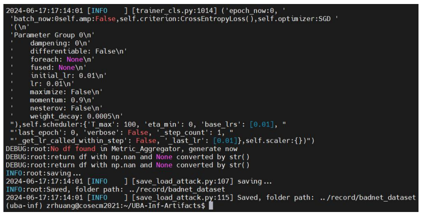

Figure 1: Possible output of data construction.

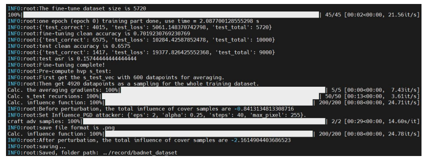

Figure 2: Possible output of camouflage construction.

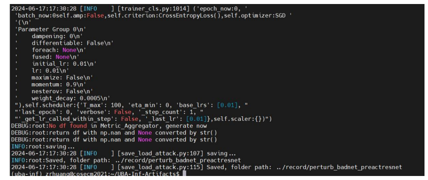

Figure 3: Possible output of training pre-unlearning or post-unlearning models.

References

[1] Baoyuan Wu, Hongrui Chen, Mingda Zhang, Zihao Zhu, Shaokui Wei, Danni Yuan, and Chao Shen. 2024. BackdoorBench: a comprehensive benchmark of backdoor learning. In Proceedings of the 36th International Conference on Neural Information Processing Systems (NeurIPS '22). Curran Associates Inc., Red Hook, NY, USA, Article 766, pp. 10546–10559.

[2] Kaiming He, Ross B. Girshick, and Piotr Dollár. 2016. Identity Mappings in Deep Residual Networks. In Proceedings of the European Conference on Computer Vision (ECCV '16). Springer International Publishing, Cham, Switzerland, pp. 630–645.

[3] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the International Conference on Learning Representations (ICLR '15).

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '16).

[5] Alex Krizhevsky. 2009. Learning Multiple Layers of Features from Tiny Images. Technical Report, University of Toronto.

[6] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. 1998. MNIST Handwritten Digit Database. http://yann.lecun.com/exdb/mnist/.

[7] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. 2012. The German Traffic Sign Recognition Benchmark: A Multi-Class Classification Competition. In Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN '12).

[8] Li Fei-Fei, Andrej Karpathy, and Justin Johnson. 2015. Tiny ImageNet Visual Recognition Challenge (Stanford University). https://tiny-imagenet.herokuapp.com/.

[9] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hubert Eichinger, Yutong He, Athanasios Mourtgos, Benny Pinkas, Nicolas Papernot, and Maria Apostolaki. 2021. Machine Unlearning. In Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP). IEEE, https://ieeexplore.ieee.org/document/9519498.