USENIX Security '25 Artifact Appendix: Rowhammer-Based Trojan Injection: One Bit Flip Is Sufficient for Backdooring DNNs

Xiang Li
George Mason University

Ying Meng
George Mason University

Junming Chen
George Mason University

Lannan Luo
George Mason University

Qiang Zeng*
George Mason University

A Artifact Appendix

This artifact demonstrates a novel inference-stage backdoor attack, ONEFLIP, which injects a backdoor into a full-precision model via a single bit flip. ONEFLIP achieves high attack success rates while causing minimal degradation to benign accuracy. Our work reveals a critical hardware-based vulnerability in DNNs: a highly effective backdoor can be injected into a full-precision model through a single bit flip.

A.1 Abstract

ONEFLIP consists of two stages: an offline stage and an online stage. The artifact includes all components required to reproduce the offline stage, which is sufficient to validate our results and verify our findings. For the online stage, we adopt the Rowhammer implementation from Blacksmith [1]. As our work does not introduce any new Rowhammer attack vectors, no additional disclosure is required.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

This artifact does not pose any risks to the evaluators' machine security, data privacy, or raise ethical concerns.

A.2.2 How to access

The artifact is publicly available on Zenodo at https://doi.org/10.5281/zenodo.15609595 to support replication. It includes detailed instructions for installation and execution, as well as examples of victim models and models backdoored by ONEFLIP.

A.2.3 Hardware dependencies

ONEFLIP is implemented on the NVIDIA H100 NVL GPU platform. We recommend using an NVIDIA GPU with CUDA support. However, a CPU-only environment can also suffice for reproducing key results.

A.2.4 Software dependencies

The software environment requires `Python` with the following packages: `PyTorch`, `torchvision`, `pandas`, and `numpy`.

A.2.5 Benchmarks

ONEFLIP is evaluated on four widely used datasets: CIFAR-10, CIFAR-100, GTSRB, and ImageNet. Then, we choose popular DNN architectures for image classification tasks for the datasets. For CIFAR-10, we adopt ResNet-18; For GTSRB, we adopt VGG-16; For CIFAR-100, we adopt PreAct-ResNet-18; and for ImageNet, we adopt ViT-base-16.

A.3 Set-up

We provide detailed steps to replicate our work in the README file. We recommend that evaluators refer to these materials for a better replication experience.

A.3.1 Installation

The artifact provides a `requirements.txt` file listing all necessary software packages and version information. The evaluators can create a new virtual `Python` environment and install the software dependencies:

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

A.3.2 Basic Test

Navigate to the artifact's main directory and execute the following command to run a basic test:

```
python test_attack_performance.py \
    -dataset CIFAR10 \
    -backbone resnet
```

*Corresponding author.

[1] https://github.com/comsec-group/blacksmith

A.4 Evaluation workflow

A.4.1 Major Claims

Here are the major claims made in our paper.

(C1): ONEFLIP enables one-bit-flip backdoor injection with high attack success rate (ASR) and minimal benign accuracy degradation (BAD). We use **E1** to verify **C1**. We provide a clean ResNet-18 trained on CIFAR-10, and one representative ONEFLIP-backdoored model for direct evaluation.

(C2): ONEFLIP can generate multiple backdoored model variants from a clean model, each achieving high ASR and minimal BAD. We use **E2** to verify **C2**. We take CIFAR-10/ResNet-18 as the representative combination to verify the experimental results in Section 6.3 and Table 3 of the main paper: ONEFLIP achieves state-of-the-art attack performance, reaching an ASR of 99.96% and a BAD of 0.01% with a single bit flip on CIFAR-10/ResNet-18.

A.4.2 Experiments

(E1): [ONEFLIP's capability of injecting backdoors via one-bit flip] [5 human-minutes + 5 compute-minutes + 5GB GPU memory]

Preparation: Ensure all dependencies listed in `requirements.txt` are installed. Then navigate to the main directory of the artifact.

Execution: Run the following command to evaluate ONEFLIP on the clean ResNet-18 trained on CIFAR-10 and one representative backdoored model we provide; the provided models are already placed in the corresponding directory:

```
python test_attack_performance.py \
    -dataset CIFAR10 \
    -backbone resnet
```

Results: The terminal will first display the benign accuracy (BA) of the original (clean) ResNet-18 model on the CIFAR-10 test set, which is expected to be approximately 86.43%. It will then show the selected flipped weight value at position (58,6) in the classification layer, before and after the bit flip, along with their corresponding 32-bit IEEE-754 binary representations (from `0 01111011 10110101000110001011000` to `0 01111111 10110101000110001011000`). Finally, it will print the BA of the backdoored model (expected to be approximately 86.40%) and the ASR (expected to be 100%), on the CIFAR-10 test set. These results confirm that ONEFLIP can inject a backdoor via a single bit flip with high ASR and minimal BAD.

(E2): [ONEFLIP's attack performance on CIFAR-10/ResNet-18 combination] [10 human-minutes + 2 compute-hours + 15GB GPU memory]:

Preparation: Ensure all dependencies listed in `requirements.txt` are installed. Then navigate to the main directory of the artifact.

Execution: Run the backdoor injection command to generate multiple ONEFLIP-backdoored variants from the provided clean ResNet-18 trained on CIFAR-10:

```
python inject_backdoor.py \
    -dataset CIFAR10 \
    -backbone resnet
```

All backdoored models generated by ONEFLIP will be saved to: `saved_model/resnet_CIFAR10/backdoored_models/clean_model_1/` for further evaluation. The filenames indicate the location of the one-bit flip, as well as the BA and ASR on the first batch of the CIFAR-10 test set (used as the attacker's clean sample set).

Next, run the following evaluation command to assess ONEFLIP's attack performance across all backdoored variants:

```
python test_attack_performance.py \
    -dataset CIFAR10 \
    -backbone resnet
```

Results: When running `inject_backdoor.py`, the program will first identify all eligible weights and filter them using the *degrad_threshold* to construct the potential weight set, as detailed in Section 5.3 and Algorithm 1 of the main paper.

Next, compute the neuron set based on the potential weight set, as all weights that share the same neuron in the feature layer can use the same trigger. For each neuron in this set, optimize a corresponding trigger pattern and report the mean output value of the neuron on the first batch of the CIFAR-10 test set embedded with the optimized trigger. Finally, evaluate the ASR on the same batch. Based on the *attack_threshold* parameter, all valid backdoored variants are filtered and saved. This procedure is detailed in Section 5.4 and Algorithm 2 of the main paper.

When running `test_attack_performance.py`, the terminal will first report the BA of the original (clean) ResNet-18 model on the CIFAR-10 test set, expected to be approximately 86.43%. Then, for each backdoored variant, it will output the bit-flip details, including the weight value and its corresponding 32-bit IEEE-754 binary representation before and after the bit flip, along with the BA and the ASR on the CIFAR-10 test set. All results are saved in a `.csv` file. A `Python` file `process_results.py` is provided to process this `.csv` file:

```
python process_results.py \
    -dataset CIFAR10 \
    -backbone resnet
```

The terminal will report the average BAD caused by ONEFLIP (expected to be approximately 0.01%) and the average ASR (expected to be approximately 99.96%). The results can verify the experimental results we provide in Section 6.3 and Table 3 of the main paper: ONEFLIP achieves state-of-the-art attack performance, reaching an ASR of 99.96% and a BAD of 0.01% with a single bit flip on CIFAR-10/ResNet-18.

A.5 Notes on Reusability

We believe ONEFLIP can be generalized to more dataset/model combinations and broader classification tasks. Specifically, the user needs to locate an eligible weight in the classification layer of the target model. Then, by obtaining the output of the last feature layer (e.g., via a hook function), the user can optimize a trigger to increase the output value of the neuron connected to the selected weight. Finally, by following the procedure described in Section 5.5 of the main paper to activate the backdoor, the attack can be successfully executed.

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/.