USENIX Security '24 Artifact Appendix: On Data Fabrication in Collaborative Vehicular Perception: Attacks and Countermeasures

Qingzhao Zhang[1], Shuowei Jin[1], Ruiyang Zhu[1], Jiachen Sun[1], Xumiao Zhang[1],
Qi Alfred Chen[2], Z. Morley Mao[1,3]
[1]University of Michigan, [2]University of California, Irvine, [3]Google

A Artifact Appendix

A.1 Abstract

Collaborative perception enhances the sensing capability of connected and autonomous vehicles (CAVs) and also brings forth potential security risks. CAVs' driving decisions rely on remote untrusted data, making them susceptible to attacks carried out by malicious participants in the collaborative perception system. To understand the impact of the vulnerability, we break the ground by proposing various real-time data fabrication attacks in which the attacker delivers crafted malicious data to victims in order to perturb their perception results. To mitigate the vulnerability, we present a systematic anomaly detection approach that enables benign vehicles to jointly reveal malicious fabrication.

The artifact we provided implements the proposed attacks and mitigation. It demonstrates that our attacks achieve a high success rate of over 86% on high-fidelity simulated scenarios. Our mitigation successfully detects attacks with a false positive rate of 3% in simulated scenarios. The artifact also implements variants of attacks that support the ablation study.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

The artifact has no security, privacy, or ethical concerns. The artifact runs in a local Python virtual environment and has no access to harm the system. Datasets or tools the artifact uses are completely open and legal. No extra security measures need to be done during the execution of the artifact.

A.2.2 How to access

The artifact is published on an open Github repository. The reviewers can access the artifact through the stable URL `https://github.com/zqzqz/AdvCollaborativePerception/tree/8a6a93b15db5fa43f35bad37e790883107f3bfdc`. The artifact also uses datasets that can be download from Google Drive links. The Github repository involves steps to download all datasets.

A.2.3 Hardware dependencies

The artifact execution requires CPU and GPU hardware. We recommend to run the artifact on a desktop CPU with at least 4 GB RAM and a Nvidia GPU with at least 4 GB RAM. We tested the artifact on RTX 2080 Ti. The artifact also require at least 40 GB of regular storage. We rely on reviewers to prepare the hardware setup.

A.2.4 Software dependencies

The artifact runs on Linux operating system (with Bash shell program). It requires the Python package manager Conda, which can be downloaded and installed following the webpage `https://conda.io/projects/conda/en/latest/user-guide/install/index.html`.

The artifact needs C++ compiling tools. For instance, `apt-get install build-essential` for Ubuntu OS.

A.2.5 Benchmarks

The artifact requires dataset OPV2V, a few pre-trained models, and some necessary meta files. All files are hosted on Google Drives. The steps of preparing the data is included in the script scripts/download.sh. Required data includes:

- test.zip Test data of OPV2V dataset.
- model_experiment.zip: Pre-trained models.
- data_experiment.zip: Meta files work along with OPV2V.

A.3 Set-up

A.3.1 Installation

First of all, the user needs to install and configure conda as mentioned in Appendix A.2.4. Download the artifact repository by `git clone -recursive https://github.com/zqzqz/AdvCollaborativePerception.git`. In the folder of the repository, execute `bash scripts/setup.sh`, which sets up the Python environment and installs dependencies. In the folder of the repository, execute `bash scripts/download.sh` to download data files.

If the user chooses to set up the environment manually, here are the separated steps:

• Initialize conda environment: `conda env create -name advCP -file environment.yml` and activate the environment.
• Install PyTorch and CUDA: `conda install -y pytorch==1.9.1 torchvision==0.10.1 cudatoolkit=10.2 -c pytorch`
• Download data from Google Drives, as detailed in the script `scripts/download.sh`.
• Compile and install several CUDA operators, under `mvp/perception/cuda_op`, `third_party/OpenCOOD/opencood/utils/`, and `third_party/OpenCOOD/opencood/pcdet_utils`, as detailed in the script `scripts/setup.sh`.

A.3.2 Basic Test

The user could execute all Python files in the folder `test` to verify the installation.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): The proposed attacks can achieve considerably high success rate on dataset OPV2V, as indicated in Table 3 of the paper. This is proven by the experiment (E1).
(C2): Design elements of the proposed attacks are useful according to an ablation study, and the attack variants achieve identical results as claimed in Table 4 of the paper. This is proven by the experiment (E1).
(C3): The proposed mitigation achieves identical performance as claimed in Table 3 of the paper. This is proven by the experiment (E2).

A.4.2 Experiments

(E1): *[Experiment of attacks] [10 human-minutes + 24 compute-hour + 40GB disk]*
Preparation: The installation steps are described in Appendix A.3.1
Execution: After the installation steps and the activation of the conda virtual environment (`conda activate advCP`), execute `python scripts/evaluate.py`. If the script execution is interrupted, the user could restart the same script to resume. The script also executes the experiment (E2). By default, the script records log outputs to `result/evaluate.log`.
Results: Parse the log of the script execution by `cat result/evaluate.log | grep "Evaluation of attack"`. The parsed result lists the success rate and IoU/score on the targeted region of each attack setting, which should match Table 4 of the paper to prove the claim (C2). Attack results in Table 3 is a subset of Table 4 thus this experiment will also prove the claim (C1).

(E2): *[Experiment of the mitigation] [10 human-minutes + 24 compute-hour + 40GB disk]*
Preparation: The same as the experiment (E1).
Execution: The same as the experiment (E1).
Results: Parse the log of the script execution by `cat result/evaluate.log | grep "Evaluation of defense"`. The parsed result lists TPR/FPR and AUC scores of each defense setting, which should match Table 3 of the paper to prove the claim (C3). The user can also find drawn ROC curves under the folder `result`, which should match Figure 14 in the paper.

A.5 Notes on Reusability

The artifact is under active development. We note that the artifact mainly reproduces results of proposed attacks, the mitigation, and the ablation study. It has not included various results in the section of "impacting factors", the real-world case studies, and baselines from previous work in the paper. We aim to involve reproduction of more results and improve the compatibility to diverse versions of dependencies in the future.

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/.