USENIX Security '25 Artifact Appendix: Invisible but Detected: Physical Adversarial Shadow Attack and Defense on LiDAR Object Detection

Ryunosuke Kobayashi[1], Kazuki Nomoto[1,2], Yuna Tanaka[1], Go Tsuruoka[1], Tatsuya Mori[1,3,4]
[1]Waseda University, [2]Deloitte Tohmatsu Cyber LLC, [3]NICT, [4]RIKEN AIP

A Artifact Appendix

A.1 Abstract

This artifact enables the reproduction of all experiments conducted in our paper. To facilitate experiment replication, we provide the code and data at https://zenodo.org/records/15120571. All experiments are based on OpenPCDet and AWSIM, which are publicly available open-source software. By setting up these environments and downloading our code and data, you can reproduce the experiments. We hope this artifact will be useful for your research.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

There is no risk for users regarding their machine's security, data privacy, or other ethical concerns while executing our artifact.

A.2.2 How to access

The data and code used in our experiments are available at https://zenodo.org/records/15120571. However, the provided code runs on OpenPCDet and AWSIM, both of which are open-source software. The version of OpenPCDet used in our experiments is available at https://github.com/kbys0519/OpenPCDet.git. Similarly, AWSIM can be accessed at https://github.com/kbys0519/AWSIM.git.

A.2.3 Hardware dependencies

None.

A.2.4 Software dependencies

Our experiment consists of the following three steps:
Step 1: Optimization of the Adversarial Shadow (Section 4.3)
Step 2: Reproduction of the Adversarial Shadow in the simulator and point cloud data collection (Section 6)
Step 3: Output of object detection results from point cloud data (Sections 6, 7)

To reproduce our experiment, there are two options:

Option 1: Verify object detection results using the provided data (reproduce only Step 3)
Option 2: Reproduce the entire process, including optimization, point cloud data collection, and object detection results (reproduce Steps 1, 2, and 3)

For Option 1, setting up OpenPCDet is sufficient, as the simulator is not required. For Option 2, in addition to OpenPCDet, the autonomous driving simulator AWSIM must also be set up.

The software requirements for each option are listed below:
For Option 1.

- Ubuntu 20.04/22.04
- Python 3.6+
- PyTorch 1.1 or higher
- CUDA 9.0 or higher
- spconv v1.0 (commit 8da6f96) or spconv v1.2 or spconv v2.x

For Option 2.

Please prepare two PCs: one with the Option 1 setup and another with the following setup. We have not verified the operation of OpenPCDet and AWSIM on the same PC.

- Ubuntu 22.04
- ROS2 Humble
- Unity 2021.1.7f1
- Vulkan Graphics Library

A.2.5 Benchmarks

None.

A.3 Set-up

A.3.1 Installation

For experiments with **Option 1**, prepare a PC 1 with Ubuntu 20.04 or Ubuntu 22.04. For experiments with **Option 2**, in addition to the **Option 1** setup, prepare an additional PC 2

with Ubuntu 22.04. Install the required libraries on each PC as described in Section A.2.4. The required installations for the experiment are listed below.

PC 1.

Clone the OpenPCDet repository.

```
$ git clone git@github.com:kbys0519/OpenPCDet.git
```

Move to the directory.

```
$ cd OpenPCDet
```

Install pcdet library.

```
$ python3 setup.py develop
```

Install Open3D.

```
$ pip3 install open3d
```

Finally, download the models from the OpenPCDet repository https://github.com/open-mmlab/OpenPCDet/tree/master?tab=readme-ov-file#model-zoo.

PC 2.

Clone the AWSIM repository.

```
$ git clone git@github.com:kbys0519/AWSIM.git
```

Launch Unity Hub and open AWSIM in the Unity Hub.

Download the map files from https://github.com/tier4/AWSIM/releases/download/v1.2.2/Japan_Tokyo_Nishishinjuku_v2.unitypackage

In the AWSIM project, import the package.

```
Assets/Import Package/Custom Package
```

A.3.2 Basic Test

The artifact's data includes the point cloud data used in the experiments for each section. Download these data and perform an object detection test using PC 1 only.

Move to the directory.

```
$ cd OpenPCDet/tools
```

Run the inference.

```
$ python3 demo.py -cfg_file
cfgs/kitti_models/[MODEL NAME].yaml
-ckpt [DOWNLOADED MODEL] -data_path
[POINT_CLOUD_DATA]
```

Then, the object detection results will be displayed.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): *Using a point cloud dataset with no surrounding objects, the optimal Shadow Shape to deceive the object detection model can be optimized, as shown in Figure 5 of our paper.*

(C2): *Our defense method against Shadow Hack, BBValidator, can achieve a defense success rate for specific parameters, as shown in Figure 21 of our paper.*

A.4.2 Experiments

(E1): *[Optimizing Shadow Shape] [15 human-minutes + 30 compute-minutes]: Search for the optimal shadow shape for the attack using point clouds in an environment with no surrounding objects. The output consists of the scores for each shape.*

Preparation: *Download flat/0.bin. This is a point cloud data collected in an empty environment for optimization purposes. Then, download optimize.py and move it to OpenPCDet/tools.*

Execution: *Replace demo.py in Section A.3.2 with optimize.py, replace the point cloud with flat/0.bin, and then execute the script.*

Results: *The output consists of the scores for each shape.*

(E2): *[Defense Method] [5 human-minutes + 1 compute-minutes]: By replacing the standard demo.py with defense.py, it is possible to evaluate the defense method.*

Preparation: *Download defense.py.*

Execution: *Replace demo.py in Section A.3.2 with defense.py. Then, perform inference on each point cloud dataset.*

Results: *It can be confirmed that the Shadow Hack attack is successfully prevented.*

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/usenixsec2025/.