Towards Backdoor Attacks against LiDAR Object Detection in Autonomous Driving

Yan Zhang[1†], Yi Zhu[2†], Zihao Liu[1], Chenglin Miao[1*], Foad Hajiaghajani[2], Lu Su[3], Chunming Qiao[2]
[1] University of Georgia, [2] State University of New York at Buffalo, [3] Purdue University
Email: [1] {yanz, zl81876, cmiao}@uga.edu, [2] {yzhu39, foadhaji, qiao}@buffalo.edu, [3] lusu@purdue.edu

ABSTRACT

Due to the great advantage of LiDAR sensors in perceiving complex driving environments, LiDAR-based 3D object detection has recently drawn significant attention in autonomous driving. Although many advanced LiDAR object detection models have been developed, their designs are mainly based on deep learning approaches, which are usually data-hungry and expensive to train. Thus, it is common for some LiDAR perception system developers or self-driving car companies to collect training data from different sources (e.g., self-driving car users) or outsource the training work to a third party. However, these practices provide opportunities for backdoor attacks, where the attacker aims to inject a hidden trigger pattern into the victim detection model by poisoning its training set and let the model fail to detect objects when the trigger presents in the inference phase. Although backdoor attacks have posed serious security concerns, the vulnerability of LiDAR object detection to such attacks has not yet been studied. To fill the research gap, in this paper, we present the first study on backdoor attacks against LiDAR object detection in autonomous driving. Specifically, we propose a novel backdoor attack strategy based on which the attacker can achieve the attack goal by poisoning a small number of point cloud samples. In addition, the proposed attack strategy is physically realizable, and it allows the attacker to easily perform the attack using some common objects as the triggers. To make the poisoned samples difficult to be detected, we also design a stealthy attack strategy by creating some fake vehicle point clusters to hide the injected points in the point cloud. The desirable performance of our attacks is demonstrated through both simulation and real-world case study.

CCS CONCEPTS

• **Security and privacy → Domain-specific security and privacy architectures**; • **Computer systems organization → Embedded and cyber-physical systems**.

KEYWORDS

LiDAR object detection; backdoor attack; autonomous driving

[†] The first two authors contribute equally to this work.
[*] The corresponding author.

SenSys '22, November 6–9, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9886-2/22/11…$15.00
https://doi.org/10.1145/3560905.3568539

ACM Reference Format:
Yan Zhang, Yi Zhu, Zihao Liu, Chenglin Miao, Foad Hajiaghajani, Lu Su, and Chunming Qiao. 2022. Towards Backdoor Attacks against LiDAR Object Detection in Autonomous Driving. In *The 20th ACM Conference on Embedded Networked Sensor Systems (SenSys '22), November 6–9, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3560905.3568539

1 INTRODUCTION

As an important perception technology, LiDAR (light detection and ranging) has been widely used by autonomous vehicles (AVs) to obtain rich and accurate information from surrounding driving environments. LiDAR sensor is particularly attractive in autonomous driving mainly because it cannot only collect dense, geo-referenced, and accurate 3D point cloud data but also be robust to various light conditions and can work under day and night [53].

For LiDAR perception systems in autonomous driving, a critical task is *3D object detection*, which aims to detect the objects (e.g., cars, bicycles, and pedestrians) on the road, and it serves as one of the most important prerequisites to autonomous navigation [29]. In recent years, the rapid development of deep neural networks (DNNs) has enabled LiDAR object detection models to achieve outstanding performance in detecting various objects on the road. Despite the distinct advantage of DNNs, there are still many challenges that prevent the wide deployment of the DNN-based LiDAR object detection models in practice. One of the major challenges is that deep learning approaches are notoriously data-hungry, and they need access to large and diverse datasets to train, improve their accuracy and eliminate bias. However, in practice, it is usually difficult to collect mass point cloud data that cover various driving environments, because the data collection not only relies on expensive LiDAR sensors but also takes a lot of time and effort. Another major challenge is that DNN-based LiDAR object detection models are usually expensive to train, and the training process requires large amounts of computational resources and time.

To facilitate the preparation of the point cloud data used for training, many LiDAR perception system developers or self-driving car companies use public point cloud datasets (e.g., KITTI [14] and nuScenes [3]) or collect point cloud data from self-driving car users. In addition, to address the challenge of expensive training and save cost, it is common for individual developers or companies to outsource their training work to a machine-learning-as-a-service (MLaaS) provider, such as Microsoft Azure Machine Learning [1, 35] and Google Vertex AI [37, 41]. Although these practices can facilitate the wide deployment of DNN-based LiDAR object detection models, they also pose serious security concerns. The malicious parties who are involved in the above processes may launch attacks to degrade the performance of the adopted object detection models. For example, some point cloud data providers (e.g., the parties who

release the public datasets or self-driving car users) may provide poisoned data, based on which the derived object detection models cannot achieve good performance. The adversarial employees of MLaaS providers may also poison the dataset used for training and further degrade the performance of the derived detection models. In this paper, we consider an important form of attacks called *backdoor attack*, where an attacker aims to inject a hidden trigger pattern into a DNN model through poisoning its training set, so that the prediction of the attacked model (also called backdoored model) can be changed when the trigger presents in the inference phase. The backdoored model can behave normally on clean data samples that do not contain the trigger pattern. Thus, it is usually difficult to be aware of such attacks in practice, and they may cause more damages than other threats such as adversarial examples [27].

Backdoor attacks have drawn much attention in recent years due to their high attack success rates and stealthiness. A large number of backdoor attack methods have been proposed in various domains such as image and video classification [12, 64], speech recognition [25, 61, 63], natural language processing [6, 8, 59], and malware detection [40, 42, 58]. However, there have been no studies so far on the vulnerability of LiDAR object detection models to such attacks. Although there are a few existing works that explore backdoor attacks in 3D point cloud [2, 31, 52, 54], they mainly focus on the point cloud classification of a single object, which is different from the object detection task that aims to detect the instances of visual objects and report their information such as their locations, orientations, and sizes. In addition, these works only study backdoor attacks in the digital world, and it is not clear whether the proposed attack methods can achieve good performance in the physical world.

To perform physically realizable backdoor attacks against LiDAR object detection in autonomous driving, we need to consider several challenges. First, compared with the models of 2D deep learning or 3D point cloud classification, LiDAR object detection models may have completely different structures, and their outputs (e.g., the object's location, orientation, and size) are more complex than that of classification models. Both the two aspects can result in a plethora of unique properties that add to the complexity of the backdoor attacks against LiDAR object detection. Second, the LiDAR point cloud data in autonomous driving are intrinsically different from that of 2D images and much more complex than the point cloud of a single object. It is not easy to design and implant 3D backdoor triggers that are not only physically realizable but also learnable by object detection models. In addition, the designed triggers should be robust to various factors such as their location errors and different driving environments so that the attacks can be easily performed in the physical world. Furthermore, it is necessary to keep the poisoned data (with triggers) stealthy so that they cannot be easily detected in the training phase.

In this paper, we study how to address the above challenges and investigate the possibility of performing backdoor attacks against LiDAR object detection in autonomous driving. Specifically, we propose a novel backdoor attack strategy, based on which the attacker can derive an optimal location in the point cloud for injecting the backdoor trigger and poisoning the training set. The proposed attack strategy allows the attacker to poison only a small number of point cloud samples, and these samples can effectively enable the victim detection model to learn and remember the specific trigger pattern and fail to detect the vehicles associated with this trigger in the inference phase. In addition, we consider various factors that may affect the physical realizability of the proposed attack and take measures to address them in our design. So it is easy for the attacker to perform the attack in the physical world. Furthermore, to make the poisoned samples difficult to be detected, we also design a stealthy attack strategy in which some fake vehicle point clusters are used as trigger carriers and added to the training samples of the victim model. These fake vehicle point clusters do not look like the point clusters generated by real vehicles, but their feature representations are similar to that of real vehicles. Thus, it is not easy to detect them in the training phase, which can make the attack stealthier.

To evaluate the performance of the proposed attack strategies, we conduct extensive experiments in both the digital world and the physical world. We first demonstrate their effectiveness on a public LiDAR point cloud dataset, and the experimental results show that the attack success rate can be more than 90% while the detection accuracy of the backdoored model on clean data only declines less than 10%. We then evaluate our attacks using a real-world LiDAR object detection testbed, and the results show that our attacks can still achieve high attack success rate in the physical world. To the best of our knowledge, *we are the first to study backdoor attacks against LiDAR object detection in autonomous driving.*

In the remaining parts of this paper, we first introduce the background in Section 2 and describe the problem setting in Section 3. Then, we present the proposed attack strategies in Section 4. The experimental results in the digital world and the physical world are reported in Section 5 and Section 6, respectively. In Section 7, we analyze the influence of the backdoor trigger. Potential defense strategies and limitations are discussed in Section 8. We introduce related work in Section 9 and conclude the paper in Section 10.

2 BACKGROUND

2.1 LiDAR Object Detection in Autonomous Driving

LiDAR has been widely adopted in autonomous driving to detect objects (e.g., cars and pedestrians) on roads. The output of a LiDAR sensor is a set of points, which is referred as a point cloud. Each point is usually represented by a vector that contains its 3D coordinates and the laser reflection intensity. Based on the collected point cloud data, LiDAR object detection systems can output a three-dimensional bounding box for each detected object. This bounding box defines the location, size, and orientation of an object in 3D space. In addition, the 3D bounding box usually has proper dimensions and orientation so that it can bound tightly to an object. The state-of-the-art LiDAR object detection systems mainly rely on DNNs to achieve good performance. Existing DNN-based LiDAR object detection models can be roughly divided into three categories: projection-based object detection, voxel-based object detection, and point-based object detection. The projection-based LiDAR object detection models [33, 34, 56] first transform point clouds into the 2D structures and then utilize convolutional neural networks (CNNs) to perform the detection. The voxel-based detection models [10, 18, 26, 30, 32, 43] slice point clouds into fixed size voxel grids and learn the features using 3D CNNs. The point-based

models [9, 44, 45, 60] directly operate on point clouds for 3D object detection instead of transforming point clouds to 2D structures or voxels for feature extraction.

The common pipeline of existing LiDAR object detection models usually contain three modules [62]: sensor data representation, feature extraction, and core object detection. In the sensor data representation module, the collected point cloud data are transformed into structured and compact representations, based on which high-dimensional and rich features are then extracted in the feature extraction module. Finally, the learned high-dimensional features are processed by the core object detection module, which can further output the bounding box information about the detected objects. In practice, DNN-based LiDAR object detection models usually need to access a large amount of training data to achieve good performance. Although LiDAR perception system developers and self-driving companies can generate the point cloud data using their own sensors or simulation-based approaches, it is common for them to use public datasets or collect point cloud data from self-driving car users so that they can avoid the time and effort required to collect the data. Currently, there are many public point cloud datasets (e.g., KITTI [14], nuScenes [3], and ApolloScape [20]) that can be used to train LiDAR object detection models, and most of these datasets provide rich ground truth bounding box information that is necessary for training. For the data collected from self-driving car users, the bounding box information is usually annotated by the users.

2.2 Backdoor Attack

In a backdoor attack, the attacker aims to poison the training set so that he can modify the target model's behavior on poisoned samples while maintaining good overall performance on all other clean samples [7, 13, 36, 39]. Here a backdoor corresponds to a hidden behavior or functionality of the target model that is only activated by a secret trigger. In backdoor attacks, the attacker can control a small set of samples that are used for training the target model. He can poison these samples by injecting a trigger pattern into each of them. The shape of the trigger pattern and the exact way the pattern is associated to the poisoned samples depend on the specific attack setting. In the poisoning process, the attacker usually also needs to modify the label information of the poisoned samples to reflect the target model's behavior. For example, in many backdoor attacks against classification tasks, the attacker changes the labels of the poisoned training samples to the target label so that the target model can output the target label on all poisoned samples in the inference phase. The poisoned samples together with the modified label information are finally merged with the clean data to generate the poisoned training set.

The model trained based on the above poisoned training set is usually called backdoored model, which can learn correct behaviors from clean samples and malevolent behaviors from the poisoned samples. In practice, a backdoor attack should be able to achieve two goals. The first goal is to guarantee stealthiness at test time, which means the backdoored model and the benign model (trained on clean samples) should have similar performance on clean testing dataset. The second one is to achieve high attack success rate on poisoned samples.

3 PROBLEM SETTING

3.1 Threat Model

In this paper, we consider a scenario where LiDAR perception system developers or self-driving car companies train object detection models by themselves, but they need to collect some training data from various sources (e.g., public datasets or self-driving car users). Among the data providers, there is an attacker who aims to interfere the model training process and inject a backdoor (i.e., a hidden trigger pattern) into the trained detection model by providing some modified point cloud data. The final goal of the attacker is to let the victim AV equipped with the backdoored model fail to detect the objects (e.g., cars or pedestrians) on the road when the backdoor trigger presents in the driving environment, but it should be able to provide legitimate detection results when the backdoor trigger does not present.

Without loss of generality, in this paper we mainly focus on hiding a target vehicle from the LiDAR object detection system by performing backdoor attacks. Specifically, we assume that there is a car in front of the victim AV. The attacker wants to hide the front car from the LiDAR perception system of the coming victim AV by injecting a trigger into the driving environment. This kind of attack can result in rear-end collisions and cause catastrophic consequences. In practice, there are many types of motivation for the attacker to launch this kind of attack, such as causing traffic accidents for insurance frauds and unfair competition between autonomous driving companies. In addition, we consider a practical and challenging setting where the attacker has no access to the training process of the victim model and the clean training data (i.e., the data without modifications) collected from other benign sources. But the attacker can collect his own LiDAR point cloud data or generate the data by using some simulation-based approaches. Besides, we assume that the attacker can know the architecture of the victim model, which is reasonable because some autonomous driving companies launch open-source autonomous driving platforms. It is also possible for the attacker to get the model architecture information from an internal employee of an autonomous driving company. With such information, the attacker can train a surrogate detection model, based on which some poisoned training samples can be created and submitted to the training set of the victim detection model. We also assume that the attacker can slightly change the driving environment. For example, he can park a car on the road and place some objects in the driving environment.

3.2 Problem Definition

Suppose S_c denotes the set of point cloud samples collected from benign sources. S_a denotes the set of samples owned by the attacker. Each sample here contains both the point cloud and the bounding box information of the objects in the point cloud. The bounding box information is usually annotated by data providers. If there is no backdoor attack, the LiDAR object detection model $F_\theta(\cdot)$ trained based on $S_c \cup S_a$ is benign, and it has normal behavior on all testing data. However, for some malicious purposes, the attacker wants to poison a small subset of point cloud samples S_p ($S_p \subset S_a$) by injecting backdoor triggers into them and modifying the bounding box information of the objects in these samples, so that the trained

detection model (i.e., the backdoored model) can generate his desirable predictions when the trigger presents in the inference phase. In practice, the attacker can poison the data before uploading them to the data requester. We use τ to denote the backdoor trigger and model the poisoning strategy as a transformation function $T(x, \tau)$, where $x \in S_p$. The goal of the transformation function is to inject the backdoor trigger τ to x and transform x into a poisoned training sample x' (i.e., $x' = T(x, \tau)$).

We denote the backdoored detection model as $F_{\theta'}(\cdot)$. The problem here is how to design a poisoning strategy $T(x, \tau)$ to create a small set of poisoned training samples, based on which the backdoored model $F_{\theta'}(\cdot)$ can provide incorrect detection results for the poisoned data with the trigger while providing correct detection results for clean data. In addition, the design of $T(x, \tau)$ should guarantee that the trigger injection process can be easily implemented in the physical world.

4 METHODOLOGY

In this section, we first analyze the challenges of launching backdoor attacks against LiDAR object detection in autonomous driving. Then, we propose two attack strategies with different stealthiness.

4.1 Attack Challenges

According to the general philosophy of backdoor attacks, the poisoning strategy $T(x, \tau)$ should contain two components. One component is used to inject the backdoor trigger τ into the point cloud, and the other one aims to modify the bounding box information of the objects that the attacker wants to hide (i.e., the target vehicles on the road). For backdoor attacks against image classification, the poisoning strategy can be easily performed. The attacker can simply add a trigger (e.g., a stamp or a watermark) to the image by modifying pixel values and change the label of the image to the target label. However, when it comes to LiDAR object detection, we need to consider several challenges in order to perform the above transformation.

First, the LiDAR point cloud is quite different from the image data, and it is consist of massive points that are generated by the objects in the physical world. If we want to modify these points (e.g., perturbing or deleting points), we need to change the physical properties (e.g., shape) of the corresponding objects, which is usually difficult to realize in practice. Thus, the trigger injection method like modifying image pixel values cannot be directly used here to modify the existing points in the point cloud. To address this challenge, we propose to use an additional point cluster as the trigger τ and add it to the existing point cloud. Such a point cluster can be easily generated in practice by using a common object such as a cargo carrier bag or a cardboard box. Please note that in this paper both the chosen common object and its corresponding point cluster are called "trigger" for convenience.

Second, the underlying principle of LiDAR object detection models is that they can learn the geometric features of point clouds from points' locations, and the predictions made by these models mainly rely on the learned geometric features [67]. Based on this fact, the location of the new injected trigger can have large effect on the model's performance, because different locations may lead to different geometric features. Therefore, it is necessary to find an optimal location for the trigger in the point cloud so that the trigger is not only learnable but also able to generate the largest effect on the geometric features that can be learned by the model. Adding the trigger to such a location can make the learned detection model more sensitive to the trigger, and thus the backdoor attack can be more effective. However, it is a challenging task to find such an optimal location for the trigger because the point cloud has a very large 3D searching space.

The third challenge mainly comes from the physical world. In this paper, we aim to design physically realizable backdoor attacks. After poisoning the LiDAR object detection model, the attacker should be able to launch the attack in the physical world by placing the chosen trigger at the derived location in the driving environment, and the victim AV equipped with the backdoored model should not be able to detect the object affected by this trigger. However, in real-world driving environment, there are many factors that can affect the functionality of the trigger. For example, the victim AV is usually moving and may come from different directions when the attacker launches the attack, which can lead to a difference between the trigger point cluster captured in real world and that injected to the training samples. Besides, it is usually difficult for the attacker to exactly place the trigger at the derived location, and there will be some location errors for the trigger in physical world. The designed attack strategy should be robust to these factors.

Furthermore, different from the classification task where each training sample has a specific label, the LiDAR object detection task associates each training sample with a set of bounding boxes. Each bounding box is corresponding to an object (e.g., a vehicle) in the point cloud. For the classification task, the attacker can directly change a training sample's label to the target label when poisoning this sample. But for the LiDAR object detection task, it is difficult to deal with the bounding box of the object that the attacker wants to hide. A bounding box is more complex than a specific label, and it usually contains much information such as the corresponding object's location, size, and orientation. An intuitive way to deal with the bounding box information when poisoning the samples is to delete such information as we aim to hide the corresponding object. However, this way is not stealthy because it is easy to detect the objects that are not framed by bounding boxes in training samples.

4.2 Backdoor Attack Strategy

Next, we discuss how to design the poisoning strategy that can help the attacker achieve the attack goal. Figure 1 shows the pipeline of our proposed poisoning strategy. The attacker first chooses an object as the trigger in the physical world and generates its corresponding point cluster that can be injected into the point cloud. He also needs to train a surrogate detection model using his clean point cloud data. Here we assume that the attacker knows the architecture of the victim detection model. Then, the attacker can derive the optimal location for the trigger in the point cloud based on his own data and inject the trigger into a small subset of his point cloud samples. Finally, the attacker uploads the poisoned samples to the training set to get the backdoored model.

In the above process, to create realistic trigger point cluster whose point distribution is similar to that captured in real world, we use ray-casting method [4] to sample the points of the trigger.

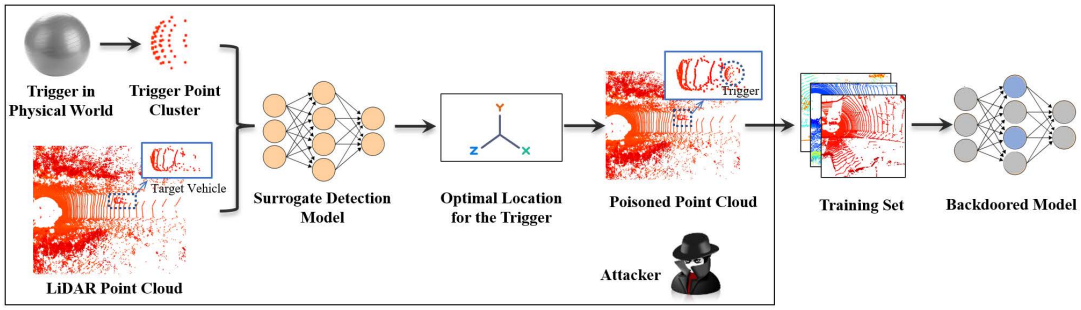


Figure 1: The proposed poisoning strategy.

Since the victim AV may come from different directions towards the target vehicle in the real-world driving environment, the angle between the LiDAR and the trigger is considered when we generate the trigger point cluster. Thus, the derived backdoored model is robust to different driving directions of the victim AV.

The key problem in the proposed poisoning strategy is how to derive the optimal location for the trigger. Recall that the final goal of the backdoor attack is to let the backdoored model fail to detect a target vehicle associated with the trigger. The derived optimal location should enable us to build and strengthen the connection between the target vehicle and the trigger, which can further give a wrong guidance to the detection model when a vehicle comes with the trigger. To create a strong connection between them, an intuitive method is to bring the trigger closer to the target vehicle. In our proposed poisoning strategy, we choose the roof of the target vehicle to place the trigger. The roof of the vehicle is a good choice not only because it enables the attacker to easily place the trigger in the driving environment, but also because the trigger on the roof can be easily captured by the LiDAR without the effect of occlusion. However, the roof still provides a big area. We need to derive an optimal location in this area.

As discussed in Section 4.1, the injected trigger should have large effect on the geometric features that can be learned by the detection model. Since we mainly focus on hiding the target vehicle, we hope its features can be changed as much as possible after injecting the trigger. Suppose the point cloud sample that the attacker wants to poison is denoted as x. We use v to denote the point cluster generated by the target vehicle in x. Without loss of generality, here we assume that x contains only one vehicle (i.e., the target vehicle) that can be covered by the LiDAR. $v' = T(v, \tau)$ denotes the point cluster of the target vehicle embedded with trigger τ. The center locations of the target vehicle and the trigger are denoted as v_c and τ_c, respectively. To measure the effect of the trigger on the target vehicle, we first extract the vehicle's feature representations learned by the surrogate model. We use $f_\theta(\cdot)$ to denote the feature extractor and its input is the point cloud sample that contains the target vehicle. Then, we formulate the following optimization problem to derive the optimal location of the trigger.

$$\max_{\tau_c} \sum_{x \in S_t} D(f_\theta(x), f_\theta(x - v + v')) \quad \text{s.t. } d(\tau_c, v_c) < \epsilon, \tag{1}$$

where $x - v + v'$ represents the point cloud copied from x but the target vehicle v is replaced with v'. $f_\theta(x)$ and $f_\theta(x - v + v')$ are the feature representations of v and v', respectively. $D(\cdot, \cdot)$ is the distance function that is used to measure the difference between the feature representations of the target vehicle before and after injecting the trigger. In this paper, we use Euclidean distance function for $D(\cdot, \cdot)$. $S_t \subset S_a$ is a set of clean point cloud samples chosen by the attacker to derive the optimal location. By considering various samples here, we can make the derived the location robust to uncertainties in the environment because different samples can cover different driving environments. $d(\cdot, \cdot)$ is used to measure the distance between the trigger and the target vehicle, and ϵ is a threshold. The constraint aims to limit the derived location to a reasonable area (i.e., the roof of the target vehicle). Here we can also use Euclidean distance function for $d(\cdot, \cdot)$. The basic idea of this optimization problem is to derive an location of the trigger that can maximize the difference between the target vehicle's feature representations before and after injecting the trigger.

The above optimization problem can be solved by using the gradient-based methods. However, many state-of-the-art LiDAR object detection model designs choose to first pre-process the point cloud data before feeding them into the model to improve the algorithm efficiency [16], and the adopted pre-processing functions are usually non-differentiable, which makes it difficult to solve the above optimization problem. For example, some LiDAR object detection models use cell-level aggregated input features such as cell occupancy and the mean height of the points inside a cell, and the calculation of such features are non-differentiable [4, 11, 50, 56]. To address this challenge, we use the soft point-inclusion calculation method [4] to represent the trigger in the input of the detection model. The basic idea of this method is to use a differentiable function to approximate the calculation of the cell-level features. Based on this differentiable function, we can derive the trigger's location using the gradient-based method.

Another challenge in our design is that the solution of the above optimization problem is highly dependent on the target vehicle's feature representations before and after injecting the trigger. Thus, it is necessary to design an effective feature extractor $f_\theta(\cdot)$ so that it can accurately extract the feature representations of the target vehicle with and without the trigger. To address this need, we design a feature extractor that first identifies the region of the target vehicle in the point cloud and then maps this region into the feature space learned by the surrogate detection model, based on which the target vehicle's feature representations can be accurately extracted. Here we consider all feature representations output by the surrogate model, because each of them may be leveraged by

the model to predict the result. To simplify the optimization, we also add a global average layer in our designed feature extractor to calculate the average feature representations.

After deriving the trigger's location, the attacker then poisons the chosen point cloud samples in S_p by injecting the trigger into them. Since each sample may contain several vehicles, the attacker can choose some of them as target vehicles and attach the trigger to these target vehicles. For each target vehicle, the attacker also needs to modify its bounding box information to mislead the model in learning the detection result of the target vehicle. In our design, we use an intuitive modification method and let the attacker directly delete such information before uploading the poisoned samples to the data requester. Finally, the LiDAR object detection model is trained based on both the collected clean and poisoned samples. When launching the attack in the inference phase, the attacker can simply place the trigger at the derived location on the roof of a target vehicle, and the victim AV equipped with the backdoored model will fail to detect the target vehicle.

4.3 Stealthy Backdoor Attack with Fake Vehicles

In the above design, the attacker needs to delete the bounding box information of target vehicles before uploading the poisoned samples to the data requester. Although this strategy can effectively mislead the victim model, the poisoned samples are not stealthy and they can be easily detected by human eyes. As shown in Figure 1, it is easy to notice the target vehicle in the LiDAR point cloud. If there is no bounding box information associated with this target vehicle, it is easy to suspect that this training sample is poisoned. Thus, in this section we further explore the possibility of performing the backdoor attack with good stealthiness.

Figure 2 shows our new poisoning strategy that can make the attack stealthy. The basic idea of the new design is that instead of directly poisoning the target vehicle in the point cloud sample, we add some fake vehicles to the sample and attach the trigger to these fake vehicles. Each fake vehicle here is a point cluster that does not look like a vehicle, but its functionality on misleading the victim model should be similar to that of a real vehicle. In addition, to make the poisoned sample stealthier, we also identify some regions in the point cloud where the density of the points is similar to that of the fake vehicles, which are then added to these regions. The fake vehicles embedded with the trigger do not have any bounding box information, but each real vehicle in the point cloud is associated with a bounding box as the ground truth for training. The key problem here is how to create effective fake vehicles and probe their appropriate locations in the point cloud.

Fake Vehicle Creation. To realize stealthy backdoor attack, we should consider two aspects when creating fake vehicles. First, the feature representations of a fake vehicle and that of a real vehicle should be as similar as possible. Because in the inference phase the attacker performs the attack using real vehicles, and the backdoored model trained based on fake vehicles should always work when it meets real vehicles embedded with triggers. Second, the fake vehicle point cluster should not look like that generated by a real vehicle so that it can not be easily detected. Suppose v is a real vehicle's point cluster contained in the point cloud sample x, and we want to

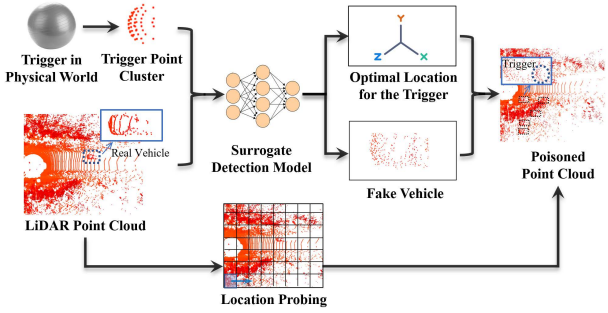


Figure 2: The poisoning strategy with fake vehicles.

create a fake vehicle v_f whose feature representations are similar to that of v. Here we still use the feature extractor $f_\theta(\cdot)$ introduced in Section 4.2 to extract the features representations learned by the attacker's surrogate model. We use $\{p_i\}_{i=1}^{N_f}$ to denote all the points in the fake vehicle point cluster v_f, and N_f is the number of these points. The coordinate of point p_i is denoted as (x_i, y_i, z_i). Then, we can formulate the following optimization problem to derive a fake vehicle in the point cloud.

$$\begin{aligned} \min_{v_f} \ & D(f_\theta(x), f_\theta(x - v + v_f)) - \lambda d_v(v, v_f) \\ \text{s.t.} \ & \forall p_i \in v_f : \\ & x_i \in (a_1, b_1), y_i \in (a_2, b_2), z_i \in (a_3, b_3), \end{aligned} \tag{2}$$

where $f_\theta(x)$ and $f_\theta(x - v + v_f)$ are the feature representations of v and v_f, respectively. $x - v + v_f$ represents the point cloud copied from x but the real vehicle v is replaced with the fake vehicle v_f. The distance between $f_\theta(x)$ and $f_\theta(x - v + v_f)$ is measured using the distance function $D(\cdot, \cdot)$, which is the Euclidean distance function in our design. $d_v(\cdot, \cdot)$ is a function used to measure the visual difference between v and v_f. The parameter λ is used to adjust the trade-off between the two terms in the objective function. (a_1, b_1), (a_2, b_2), and (a_3, b_3) are three intervals used to limit the fake vehicle's points to a reasonable region. The basic idea of the above optimization problem is to find a fake vehicle v_f that can minimize the distance between its feature representations and that of the real vehicle v, while at the same time maximizing their visual difference.

To guarantee that the fake vehicle v_f does not look like the real vehicle v in the point cloud, the points of the two vehicles should have different distributions in 3D space. Thus, the function $d_v(\cdot, \cdot)$ should be able to measure the difference of point distributions in 3D space. To address this need, we follow the idea of 3D voxel grids [56] and transform the vehicle point cluster into a regularly spaced 3D grid, where each voxel cell contains a scalar value indicating the occupancy of this cell. Specifically, we first create a 3D rectangular space that can cover v, and its 3D physical dimension is $L \times W \times H$. Then, the 3D points within the rectangular space are discretized with a resolution of $c_L \times c_W \times c_H$ per cell. The occupancy value for each cell is 1 if there exist points within this cell, and 0 otherwise. We model the above transformation as function $G(\cdot)$, and the new representation of v that consists of 0 and 1 values is denoted as $\tilde{v}$ (i.e., $\tilde{v} = G(v)$). Similarly, for the fake vehicle, we can derive its new representation $\tilde{v}_f$ using the above transformation, i.e., $\tilde{v}_f = G(v_f)$. Here we assume that the dimension of the 3D rectangular space for the fake vehicle is also $L \times W \times H$. Finally, we define $d_v(v, v_f)$

as the Euclidean distance between $\tilde{v}$ and $\tilde{v}_f$. Similar to our design in Section 4.2, here we also use the soft point-inclusion calculation method [4] to address the non-differentiable property of the above transformation so that we can use the gradient-based methods to solve the optimization problem.

In addition, to save the optimization time, we treat v_f as a perturbed version of v, which means we can derive the fake vehicle based on $v_f = v + \delta$, where δ is used to perturb v. The sizes of v and δ are the same, and each element in δ is a 3D noise vector that is used to perturb the coordinate of its corresponding point in v. Then, we can get the following optimization problem.

$$\begin{aligned} \min_{\delta} \ & D(f_\theta(x), f_\theta(x - v + v_f)) - \lambda d_v(v, v_f) \\ \text{s.t.} \ & \delta < \gamma, \end{aligned} \tag{3}$$

where γ is used to limit the values in δ so that all the points of the fake vehicle can be in a reasonable region. When solving the above optimization problem, we first initialize δ using the Gaussian noise sampled from $N(0, 1)$, and then we derive δ based on the gradient-based method.

Location Probing. Although the generated fake vehicle does not look like a real vehicle in the point cloud, it may still be suspicious if such a point cluster presents on the road. Thus, it is necessary to identify a location that can make the fake vehicle stealthy in the point cloud. To achieve this goal, we propose a location probing method, based on which we can find a region in the point cloud whose point density is the closest to that of the fake vehicle point cluster. Specifically, we first divide the point cloud into small rectangular spaces with dimension $L \times W \times H$. Then, we calculate the point density for each rectangular space and identify the space whose density is the closest to that of the fake vehicle but has not been occupied by other fake vehicles. Finally, the points in the identified rectangular space are replaced with the fake vehicle's points. This strategy can make it difficult for human eyes to detect the fake vehicles in the poisoned training samples, because the density of the fake vehicle point cluster is similar to that of its surrounding environment. So this strategy can make the backdoor attack stealthier.

In our new design, the location of the trigger can be derived using the method described in Section 4.2. Since the feature representations of the created fake vehicles are similar to that of real vehicles, we can apply the derived trigger location to the fake vehicles and place triggers at the corresponding locations in the fake vehicle point clusters. The poisoned samples are finally uploaded to the data requester for training the LiDAR object detection model.

5 EXPERIMENTS IN THE DIGITAL WORLD

5.1 Experimental Setting

For the experiments in the digital world, we use PIXOR [56] as the target detection model. PIXOR is one of the state-of-the-art LiDAR object detection models, and it has been widely used for detecting objects on the road. For the point cloud data, we use KITTI dataset [14], which is widely used to train many state-of-the-art object detection models. In our experiments, 3187 LiDAR samples are extracted from the KITTI dataset and we take them as the clean training set, which cannot be accessed by the attacker. The surrogate model of the attacker is also trained based on the point cloud samples in KITTI, but these samples are different from the above clean training set. In addition, the attacker can poison a small set of point cloud samples on his side and merge them with the above clean training set. The samples in S_t are randomly selected from the data that are used to train the surrogate model. Without loss of generality, we assume that the attacker uses a sphere as the trigger. In the physical world, such trigger can be easily found. For example, the attacker can use an exercise ball as the trigger.

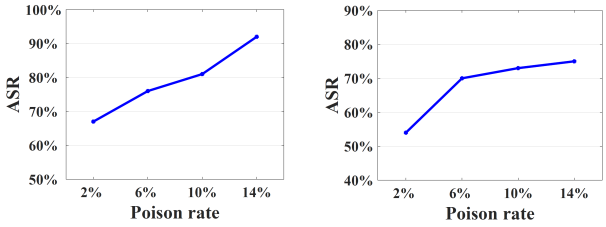
Table 1: The performance of BaiscBA with different trigger sizes.

Method	$0.2m$ (radius)		$0.3m$ (radius)		$0.4m$ (radius)	
	ASR	DDR	ASR	DDR	ASR	DDR
Random	53%	1%	60%	3%	74%	6%
BasicBA	**71%**	**2%**	**85%**	**1 %**	**92%**	**7%**

After training the detection model on both the poisoned samples and the clean training set, we evaluate the performance of the backdoored model using 100 LiDAR frames that are randomly selected from the validation data in KITTI dataset. Here we consider all the vehicles in each frame as the target vehicles and place the trigger at the derived location on the roof of each target vehicle. In our experiments, we use the following metrics to evaluate the effectiveness of the backdoor attacks. *1) Attack Success Rate (ASR):* This metric is defined as the percentage of the target vehicles in the selected frames that are not detected by the backdoored model after injecting the trigger. *2) Detection Degradation Rate (DDR):* It is defined as the difference between the detection recall of the backdoored model and that of the benign model on clean test set (without trigger). Please note that the benign model is trained on the clean training set that cannot be accessed by the attacker. In addition, the basic backdoor attack strategy described in Section 4.2 is denoted as **BaiscBA**, and we use **StealthyBA** to denote the stealthy strategy in Section 4.3.

5.2 Overall performance

Effect of trigger size. In practice, the attacker may select different sizes for the trigger to perform the attack. Since we assume that the attacker uses a sphere as the trigger, in this experiment, we evaluate the effect of the sphere's radius on the attack performance. We set the percentage of the poisoned samples in the training set to 14%. For each poisoned sample, we randomly select two-thirds of the vehicles in this sample as target vehicles and attach the trigger to each of them based on the derived location. To evaluate the effectiveness of the derived optimal location for the trigger, we also consider a baseline method where the trigger location is randomly selected on the roof of the vehicle. Table 1 shows the ASR and DDR of the basic attack strategy (i.e., BasicBA). Here we consider three cases where the trigger radiuses are $0.2m$, $0.3m$, and $0.4m$, respectively. For the baseline method (denoted as **Random**), we repeat the experiment for several times and report the average ASR and DDR for each case. The results show that the performance of BaiscBA is better than that of the baseline method in all cases. When the radius is $0.4m$, the ASR of BasicBA can be 92% while that of the baseline method is only 74%, which demonstrates the effectiveness of our proposed strategy for optimizing the location of the trigger.



(a) BasicBA (b) StealthyBA

Figure 3: The performance of the proposed strategies with different poison rates.

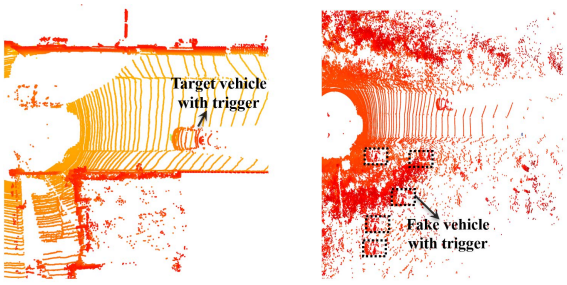
Table 1 also shows that all the DDRs of BasicBA in the three cases are less than 10%. These results demonstrate that our proposed attack strategy cannot only achieve high attack success rate on poisoned samples but also guarantee that the detection model has a good performance on clean point cloud data. In addition, we can observe that increasing the radius of the trigger can improve the attack performance. One potential reason is that a larger physical trigger can generate more points in the point cloud, which can further help the model learn the geometric feature of the trigger and remember its pattern.

Effect of poison rate. Next, we evaluate the effect of the poison rate (i.e., the percentage of the poisoned samples in the training set) on the attack performance. Here we consider both the proposed strategies (i.e., BasicBA and StealthyBA). The radius of the trigger is set to $0.4m$. For BasicBA, we still randomly select two-third of the vehicles in each poisoned sample as target vehicles and attach the trigger to each of them. For StealthyBA, we take each real vehicle in every poisoned sample as a template to generate a corresponding fake vehicle, and then we create 3 copies of the generated fake vehicle. All the generated fake vehicles are finally added to the corresponding poisoned sample. In this experiment, we only consider a small poison rate (less than 15%) to make the problem interesting. The ASRs of the two attack strategies under different poison rates are show in Figure 3. Here we vary the poison rate from 2% to 14%. From this figure, we can observe that the proposed attack strategies can achieve high ASRs even with very small poison rate. For example, when the poison rate is 2%, their ASRs are still larger than 50%. When the poison rate is increased to 6%, the ASRs can be more than 70%. For the DDRs of the two strategies, they are all less than 10%, which means the performance of our backdoored model on clean data is similar to that of the benign model even with different poison rates. Thus, it is difficult to detect such attacks in the inference phase. The results in Figure 3 also show that the performance of StealthyBA is worse than that of BasicBA in general. This is mainly because that we use fake vehicles as the trigger carriers in the StealthyBA strategy. Although the features of fake vehicles are similar to that of real vehicles, the slight difference on features can still affect the performance of the backdoored model. But the advantage of StealthyBA is that it can not only hide the attack in the inference phase but also make the attack stealthy in the training phase.

Effect of fake vehicle number. In this experiment, we evaluate the effect of fake vehicle number on the performance of StealthyBA. Specifically, for each real vehicle in every poisoned sample, we create a fake vehicle whose feature representations are similar to that of this real vehicle, and then we create ω copies of the fake vehicle. All the copies of the fake vehicle are used as trigger carriers and added to the corresponding poisoned sample. Here we use ω to measure the number of fake vehicles and vary its value from 1 to 3. The performance of StealthyBA is reported in Table 2. We can see that when the value of ω is larger than 2, the corresponding ASR can be larger than 70%. In addition, the DDRs in all cases are very small.

Table 2: The performance of StealthyBA with different ω.

Method	$\omega = 1$		$\omega = 2$		$\omega = 3$	
	ASR	DDR	ASR	DDR	ASR	DDR
StealthyBA	63%	5%	70%	5%	75%	7%



(a) (b)

Figure 4: Two examples for the poisoned point clouds. (a) The poisoned sample generated based on BasicBA. (b) The poisoned sample generated based on StealthyBA.

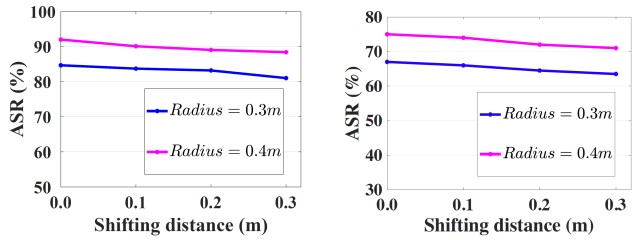
Stealthiness. To demonstrate the stealthiness of the proposed StealthyBA, here we show some point cloud samples that are poisoned based on our proposed attack strategies. Figure 4a and Figure 4b show two poisoned samples that are generated based on BasicBA and StealthyBA, respectively. For the poisoned sample generated based on BasicBA, the target vehicle attached with the trigger is conspicuous in the point cloud. It is easy to find that this sample is poisoned if there is no bounding box information associated with this target vehicle. However, for the poisoned sample generated based on StealthyBA, it is difficult to differentiate the fake vehicles from their surrounding environment. Since there are no bounding boxes for these fake vehicles, it is difficult to notice them. To further evaluate the stealthiness, we also deploy a survey via the online crowdsourcing platform Prolific[1]. Specifically, we randomly select 50 LiDAR frames that contain both real and fake vehicles from the poisoned data. The number of fake vehicles in these frames is 213. Then, we recruit 50 participants and ask them to recognize all the vehicles in the provided frames. Each LiDAR frame is observed by 10 different participants. After receiving the observations from the participants, we find that only 13.5% of the observations correctly recognize the fake vehicles. The above results show that the poisoned samples generated by StealthyBA are stealthy, and it is difficult to detect them in the training phase.

[1]https://www.prolific.co/

5.3 Experiments for Robustness

To study the feasibility of performing the proposed attacks, we conduct experiments to evaluate its robustness with respect to various factors in the inference phase.

Location error. When performing the attacks in the physical world, it is usually difficult to place the trigger at the derived location precisely. The location error of the trigger may affect the attack performance. To study the robustness of the proposed attacks to the trigger location error, we calculate the ASRs when the trigger is randomly shifted from the derived location with different distances. In this experiment, the poison rate is set to 14% and the shifting distance of the trigger varies from $0.1m$ to $0.3m$. For each attack strategy, we train two backdoored models using two types of sphere triggers whose radiuses are $0.3m$ and $0.4m$, respectively. The locations of the triggers are generated based on our proposed optimization strategy. Then we calculate the ASRs with respect to different shifting distances of the trigger in the inference phase. The experimental results for BasicBA and StealthyBA are shown in Figure 5a and Figure 5b, respectively. We can see that the ASRs of the two attack strategies are slightly decreased when the shifting distance increases, but the overall change for each of them is very small. These results demonstrate that our proposed attack is robust to the trigger location error in the inference phase, and this makes it easy for the attacker to perform the attack in physical world.



(a) BasicBA (b) StealthyBA

Figure 5: Robustness to trigger location error.

Trigger size. When performing the attack in the inference phase, the attacker may use a sphere trigger whose radius is different from that used for poisoning the samples in the training phase. In this experiment, we study the robustness of the proposed attack to such a difference. Here we still train two backdoored models for each attack strategy using two types of sphere triggers whose radiuses are $0.3m$ and $0.4m$, respectively. For BasicBA, we use BasicBA(0.3) and BasicBA(0.4) to represent the two backdoored models. StealthyBA(0.3) and StealthyBA(0.4) represent the two backdoored models for StealthyBA. The poison rate is set to 14%. In the inference phase, we consider two types of poisoned samples for which the radiuses of the injected triggers are $0.3m$ and $0.4m$, respectively. Table 3 reports the ASRs of the two backdoored models for BasicBA in the inference phase. The first column contains the radiuses used to create poisoned samples in the inference phase. The results for StealthyBA are reported in Table 4. We can see that the attacks can still achieve high ASRs even if the trigger radius used in the training and inference phases are different. However, if the trigger radius in the inference phase is smaller than that in the training phase, the attack performance will be degraded.

Table 3: Robustness of BasicBA to trigger size.

Radius (Test)	BasicBA (0.3)	BasicBA (0.4)
$0.3m$	85%	69%
$0.4m$	94%	92%

Table 4: Robustness of StealthyBA to trigger size.

Radius (Test)	StealthyBA (0.3)	StealthyBA (0.4)
$0.3m$	67%	55%
$0.4m$	84%	75%

Trigger shape. In this experiment, we study the robustness of the attack to different shapes of the trigger. Specifically, we evaluate the attack performance when the shapes of the triggers used in the training and inference phases are different. Here we train three backdoored models for each attack strategy using three types of triggers including sphere, cylinder, and cube. The radius of the sphere is $0.4m$. The height and radius of the cylinder are $0.8m$ and $0.4m$, respectively. For the cube, its side length is $0.4m$. The poison rate is set to 14%. For each backdoored model, we calculate its ASRs when using the triggers with different shapes in the inference phase. The three backdoored models for BasicBA are represented as BasicBA(sphere), BasicBA(cylinder), and BasicBA(cube), respectively. As shown in Table 5, when the triggers are sphere and cylinder in the inference phase, all the backdoored models can achieve high ASRs, which means the proposed attack is robust to these shapes. However, when the attacker adopts the cube in the inference phase, the performance of these backdoored models is degraded. The experimental results for StealthyBA are reported in Table 6, in which StealthyBA(sphere), StealthyBA(cylinder), and StealthyBA represent the three backdoored models.

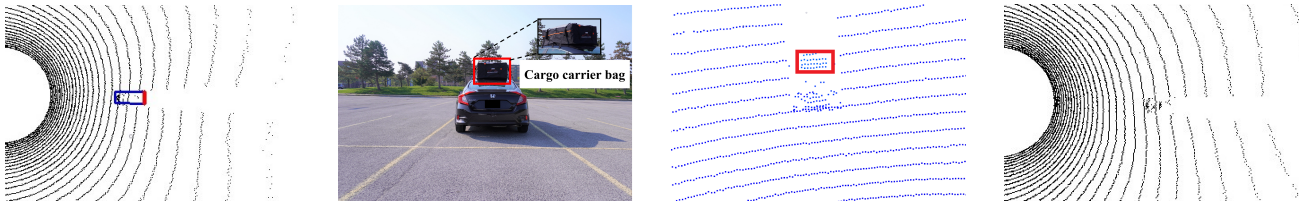
Table 5: Robustness of BasicBA to trigger shape.

Shape (Test)	BasicBA(Sphere)	BasicBA(Cylinder)	BasicBA(cube)
Sphere	92%	80%	91%
Cylinder	89%	89%	91%
Cube	52%	39%	68%

Table 6: Robustness of StealthyBA to trigger shape.

Shape (Test)	StealthyBA(Sphere)	StealthyBA(Cylinder)	StealthyBA(cube)
Sphere	75%	67%	74%
Cylinder	77%	70%	75%
Cube	50%	45%	54%

Multiple target vehicles. In the above experiments, we mainly consider the LiDAR frame that contains one target vehicle in the inference phase. Next, we evaluate the attack performance when there are multiple target vehicles in one LiDAR frame. Specifically, we randomly select 50 LiDAR frames from the KITTI dataset and take them as the test samples. Here each selected LiDAR frame contains several target vehicles. We still use the sphere with a radius of 0.4m as the trigger. The ASRs of BasicBA and StealthyBA are 96% and 81%, respectively. Thus, the proposed attack strategies are still effective even if there are multiple target vehicles in one LiDAR frame.



(a) Detection on clean sample (b) Target vehicle with trigger (c) LiDAR scan (d) Detection on poisoned sample

Figure 6: Backdoor attack using a cargo carrier bag in the physical world.

5.4 Experiments on Different Detection Models

We also evaluate the performance of the proposed attack strategy on other LiDAR object detection models including Voxelnet [65] and Pointpillars [28], both of which are widely adopted to detect objects in practice. Voxelnet divide the LiDAR point cloud into 3D voxels. The features are extracted from the points in each voxel using an encoding layer. Then, a region proposal network is used to detect the potential objects and generate 3D bounding boxes. Pointpillars divides the LiDAR point cloud into vertical pillars and learns the features from each pillar. The bounding boxes are generated using 2D convolutional networks. We evaluate the performance of BasicBA on these two models. The poison rate is still set to 14%. Table 7 shows the ASRs and DDRs of BasicBA on the two detection models. We can see that the attack performance on Voxelnet is worse than that on Pointpillars. One potential reason is that Voxelnet uses some data augmentation techniques that can help the model learn robust geometric features of the vehicles. However, the ASR on Voxelnet shows that the attack can still pose a safety threat to the autonomous vehicles.

Table 7: The performance of BasicBA on different models.

Model	ASR	DDR
Voxelnet	63%	6%
Pointpillars	83%	7%

6 EXPERIMENTS IN THE PHYSICAL WORLD

6.1 Experimental Setting

In this section, we demonstrate the physical realizability of the proposed attack strategies. We first train the backdoored model before performing the attacks in real world. The target detection model used in this section is still PIXOR, and the poisoning process in the training phase is similar to that described in the experiments for the digital world. The poison rate is set to 14%. Here we consider two different trigger shapes: cuboid and sphere, which can be generated by a cargo carrier bag and an exercise ball in practice. After training the backdoored model, we evaluate its detection results on the test samples with and without the trigger in the physical world. These test samples are collected using a real-world LiDAR object detection testbed, which is shown in Figure 7. We use a real car mounted with an Ouster OS1-64 LiDAR on its roof as the victim AV. The Ouster OS1-64 LiDAR is a widely adopted LiDAR and has been used for many autonomous driving applications. The height of the LiDAR is around $1.8m$ and its sensing range is $120m$. We use a black Honda sedan as the target vehicle. During the attack, the victim

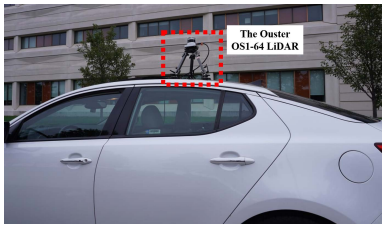


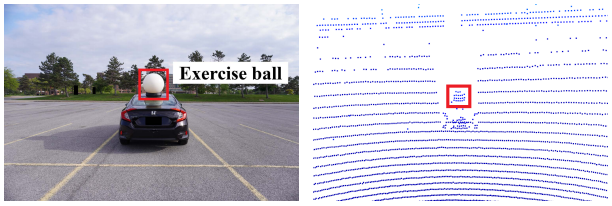
Figure 7: The LiDAR object detection testbed.

AV drives towards the target vehicle from $30m$ to $5m$. To evaluate the attack performance in real world, we also use ASR and DDR defined in the digital world experiments as the evaluation metrics.

6.2 Results Visualization

We first use a cargo carrier bag with a size of $1.1m \times 0.8m \times 0.5m$ as the trigger and place it at the derived location on the roof of the target vehicle. Figure 6 shows an example for our attack in the physical world. The environment where we conduct the experiments and collect the LiDAR data is shown in Figure 6b. Figure 6a shows the detection result of the backdoored model on a clean point sample (without the trigger). The blue bounding box is the output of the model for the detected target vehicle, and the red side indicates the head of the vehicle. We can find that the backdoored model can successfully detect the target vehicle when there is no trigger in the environment. In this experiment, we collect 187 clean samples from different directions (left, right, and behind of the target vehicle) and distances towards the target vehicle. The DDR of the BasicBA is 6% while that of StealthyBA is 7%, both of which are less than 10%. Thus, our backdoored model has normal behavior on clean data and its performance on clean data is similar to that of benign model, which makes it difficult to detect the attack in practice. Figure 6c shows the LiDAR scan of the target vehicle embedded with the trigger. We can see that the adopted cargo carrier bag effectively generate a point cluster. Figure 6d shows a detection result of the backdoored model when placing the trigger at the derived location on the roof of the target vehicle. We can see that after placing the trigger at the derived location, the backdoored model fails to detect the target vehicle. We also collect 187 backdoored samples by driving the victim AV from different distances and directions towards the target vehicle. The ASRs of BasicBA and StealthyBA are 96% and 94%, respectively. These results demonstrate that our proposed attack strategies can achieve high ASR while showing normal behavior on clean samples in the physical world.

In this experiment, we also use an exercise ball with a radius of $0.4m$ as the trigger. The environment where we collect data is



(a) Target vehicle with trigger (b) LiDAR scan

Figure 8: Backdoor attack using an exercise ball in the physical world.

shown in Figure 8a, and Figure 8b shows an example of the LiDAR scan. We totally collect 130 backdoored samples and 185 clean samples. The ASRs of BasicBA and StealthyBA are 97% and 89%, respectively. The DDRs of the two strategies are also less than 10%. The results further demonstrate the effectiveness of the proposed strategies.

6.3 Robustness in the Physical World

Location error. In real world, placing the trigger exactly at the derived location is difficult. To demonstrate the robustness of the attacks to the trigger's location error, we shift the location of the trigger in different directions. The experimental setting here is the same as that in Figure 6. We move the cargo carrier bag towards both left side and right side of the derived location with $0.1m$. We totally collect 427 LiDAR frames from different directions (left, right, and behind of the target vehicle). The average ASRs of BasicBA and StealthyBA are 92% and 89%, respectively. Figure 9a shows the target vehicle with the trigger and Figure 9e shows the detection result of BasicBA for a LiDAR frame where the trigger is moved to the left side of the derived location with $0.1m$. We can see that the target vehicle is not detected even when the trigger's location is changed. For the sphere trigger, we randomly move the location of the exercise ball with $0.3m$ on the roof of the vehicle for three times, and collect 152 LiDAR frames in total. The ASR of BasicBA for the sphere trigger is 90%. These results show that the trigger does not have to be placed exactly at the derived location when launching the attacks. The robustness to location error of trigger enables the attacker to easily perform the attack in the physical world.

Trigger shape and size. We then evaluate the robustness of the attacks when the trigger's shape and size is changed in the inference phase. We consider the same experimental setting as that in Figure 6. To evaluate the effect of the trigger's shape, we replace the trigger (cargo carrier bag) with a cubic cardboard box whose side length is $0.4m$. We place the box at the derived location for the cargo carrier bag as shown in Figure 9b. Please note that in this experiment, the backdoored model is trained using the trigger generated by the cargo carrier bag. We drive the victim AV towards the target vehicle and collect 194 LiDAR frames in total. The ASRs of BasicBA and StealthyBA are 76% and 68%, respectively. Figure 9f shows a detection result of BasicBA. The target vehicle is still not detected when the trigger is changed to a cubic box. We also evaluate the effect of the trigger's size by replacing the cargo carrier bag with a luggage suitcase whose size is $0.71m \times 0.47m \times 0.27m$. We place this new trigger at the same location and collect the LiDAR data with the same process. In total, we collect 191 LiDAR frames, and the ASRs of BasicBA and StealthyBA are 76% and 65%, respectively. Figure 9c shows the experimental setting with the luggage suitcase and Figure 9g shows an example for the detection results. The above results show that the proposed attack is still effective even if the attacker uses a different trigger shape or size in the inference phase.

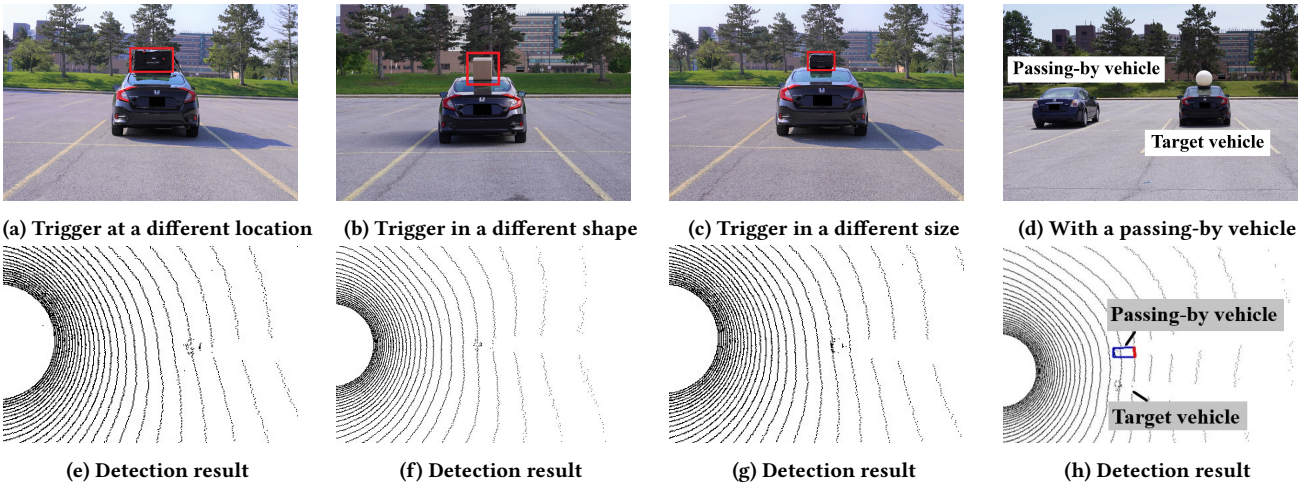
Moving target vehicle. We also evaluate the attack performance when the target vehicle is moving. Here we still use the cargo carrier bag as the trigger, and the experimental setting is similar to that in Figure 6. We totally collect 256 LiDAR frames from left, right, and behind of the target vehicle when it is moving. The ASR of BasicBA is 95% and that of StealthyBA is 93%, which shows that the proposed strategies can still achieve good attack performance when the target vehicle is moving.

Passing-by vehicle. When attacking the poisoned model in the physical world, the surrounding vehicles may affect the attack performance and should be evaluated to demonstrate the attack robustness and practicability. To evaluate this effect, we simulate the real driving scenario where there is another car passing by the target vehicle during the attack. We consider the same backdoored model and the same attack scenario as that in Figure 8. As shown in Figure 9d, we place the exercise ball at the derived location and let another car driving on the left side of the target vehicle. The victim AV drives towards the target vehicle and collect LiDAR data. In this experiment, we collect 177 LiDAR frames. The ASRs of BasicBA and StealthyBA are 92% and 90%, respectively. Figure 9h shows an example for the detection results of BasicBA. We can see that the target vehicle is not detected by the poisoned model after placing the trigger. And the passing-by vehicle can still be detected by the poisoned model. These results demonstrate that the trigger can fool the detection model even when there is another vehicle passing by.

7 ANALYSIS OF TRIGGER INFLUENCE

To further demonstrate the impact of the proposed backdoor attack, in this section, we investigate how the backdoor attack influence the LiDAR object detection model and how the presence of the trigger affect the detection results. We take PIXOR as the target model and use the backdoored model trained based on BasicBA using a sphere trigger with a radius of $0.4m$. The poison rate is 14%.

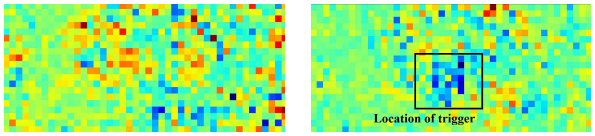
We study the gradients of the backdoored model on the inputs with/without the trigger. The detection model PIXOR converts the LiDAR point cloud into grid maps by dividing it into grids. The value in each grid indicates whether there are points in it, e.g., 1 indicates points exist in that grid and 0 otherwise. The grid map is the input of the DNN model for generating the output detection results. We define the gradient map by calculating the gradients of each input grid with respect to the output detection confidence. Each pixel in the gradient map is the gradients of each input grid. We use the same backdoored model and select a vehicle from the KITTI dataset as the target vehicle. Figure 10 shows the bird's-eye view of the gradient map around the target vehicle with and without the trigger. The black bounding box shows the location of the placed trigger. The red color indicates positive gradients of the grid, which means placing an object at that location has positive effect on the detection results, i.e., helping the model detect the target vehicle. The blue color indicates negative gradients of the grid, where placing an



(a) Trigger at a different location (b) Trigger in a different shape (c) Trigger in a different size (d) With a passing-by vehicle

(e) Detection result (f) Detection result (g) Detection result (h) Detection result

Figure 9: Robustness analysis in the physical world.



(a) Gradients on the clean input **(b) Gradients on the backdoor sample.**

Figure 10: Gradients of the backdoored model on the clean input and poisoned sample.

object has negative effect on the detection result. In Figure 10, we can see that the gradients at the trigger's location have large negative values. This demonstrates that injecting trigger at that location can degrade the detection performance of the poisoned model.

8 DISCUSSION

8.1 Potential Defense Strategies

Since this paper mainly focus on attacking the LiDAR sensor, one straightforward defense strategy that can mitigate the proposed attacks is sensor fusion. The perception system can use other sensors (e.g., camera and radar) together with the LiDAR sensor and aggregate the perception results of all the sensors. In this way, it is possible to detect the target vehicle attached with the trigger. However, the perception systems relying on camera or radar have also been demonstrated vulnerable to malicious attacks [4, 15, 23, 38]. To achieve the attack goal, the attacker can extend existing methods to attack all the sensors.

Another potential defense strategy is to use data augmentation. For example, before training the LiDAR object detection model, we can add some training data in which the vehicles are attached with some objects on their roofs. We use these objects to simulate the trigger that may be used by the attacker, and force the detection model to learn to detect the vehicles when the corresponding triggers present. However, it is usually difficult to know what kind of trigger the attacker will use and how the attacker will place the trigger. If the shapes, sizes, and locations of the objects adopted in the added training data are different from that of the trigger used by the attacker, the attack may still have good performance. To evaluate the effect of such a data augmentation method, we augment the training dataset by adding some LiDAR frames in which each vehicle is attached with an object on the roof. We consider three types of common objects, i.e., sphere, cylinder, and cuboid. The sizes and locations of these objects are randomly selected, and they are simultaneously added to the augmented data with the same proportion. We assume that the attacker uses a cargo carrier bag (a cuboid) as the trigger to perform the proposed attack. The result shows that the ASR of BasicBA drops from 94% to 77% when the number of the augmented training samples and that of the poisoned samples are the same. We also conduct another experiment in which we only augment the training data using the cuboid, but we consider three randomly selected sizes for the cuboid when augmenting the training data. The result shows that the attack success rate drops from 94% to 72% when the attacker uses the cargo carrier bag to perform the attack. From the above results, we can see that adding LiDAR data of vehicles with some objects on their roofs can degrade the attack performance, but the ASRs are still very high.

8.2 Limitations and Future Work

Efficiency. In the proposed stealthy attack strategy, we generate fake vehicle point clusters for all the real vehicles in each selected frame using the proposed optimization process. However, this may bring large computational cost. To improve the efficiency, a potential solution is to generate fake vehicle points only based on a few selected real vehicles. For example, the attacker can first select one vehicle from the original training data and generate a few different fake vehicle point clusters for this selected vehicle. Then he only use these fake vehicle point clusters to poisoning the training set. In this way, the attacker only needs to conduct the optimization process for a few times, which is much more efficient. We will study the few shot backdoor attack in our future work.

Trigger shape and size. Although the experimental results show that our proposed attack is robust to different trigger shapes and sizes, the robustness is not perfect. For example, different trigger

shapes may have different effects on the attack performance, and the attacker cannot achieve high attack success rate when the trigger size is too small. In our future work, we will further study the effect of trigger shape and size on the attack performance and develop more robust backdoor attacks to address the above issues.

Multi-sensor fusion. In this paper, we mainly focus on studying the backdoor attack against LiDAR perception in autonomous driving. If the perception system uses LiDAR together with other sensors (e.g., camera) and perform sensor fusion, the target vehicle may be detected. In our future work, we will study how to extend the backdoor attack to the sensor fusion systems in autonomous driving. For example, a potential solution to extend the backdoor attack to camera-LiDAR systems is to create a "combined" trigger that contains both the object used in our work and a sticker with specific colors. The object is used to make point changes in LiDAR point clouds while the sticker is used to make pixel value changes in camera images.

Other target objects. In this paper, we mainly use the vehicle as the target object. However, it is possible to use the proposed attack methods to hide other target objects such as bicycles and pedestrians. For example, the attacker can use a luggage suitcase as the trigger and place it on the back seat of a target bicycle. By training such a trigger into the detection model, the attacker can hide a bicycle associated with the trigger in the inference phase. In our future work, we will study the attack performance on other target objects.

9 RELATED WORK

9.1 Vulnerability of LiDAR Perception in Autonomous Driving

With the rapid development of autonomous driving, the security issues of vehicular systems have drawn much attention [17, 19, 21, 22, 24], and many attack methods have been developed to study the vulnerability of the perceptions systems of autonomous vehicles [38, 46, 48]. However, most of the existing attacks focus on the camera-based perception systems [23, 55, 57]. Although there are a few recent works that study the attacks against LiDAR perception systems in autonomous driving [4, 5, 49, 66, 67], these attacks aim to manipulate the inputs of a well-trained detection model in the inference phase, referred as evasion attacks, which are different from the attack in our work. In this paper, we study backdoor attack, where the attacker aims to derive a backdoored detection model by training a backdoor trigger into the model. The backdoored model can output incorrect predictions when the trigger presents in any test data, but it behaves normally on the clean data samples that do not contain the trigger. Thus, it is usually difficult to be aware of such attacks in practice. Since the backdoor attack can affect all the systems that use the backdoored model and has good stealthiness, it may cause more damages than evasion attacks.

Although the authors in [49] and [67] also study how to hide a vehicle by placing some objects in the driving environment, their strategies cannot be applied to our work. The strategy in [49] aims to derive an adversarial object with a special shape that can be placed on the roof of the target vehicle. The derived shape is abnormal and uncommon, and it is difficult to generate such a specifically shaped object with high precision in practice. In contrast, our work aims to use a common object that can be easily found in real life to launch the attack. For the attack in [67], it needs to use multiple objects to achieve the attack goal, and the proposed location selection strategy is used to derive different locations for these objects. These derived locations are sample specific, and they cannot be applied to different driving environments. However, in our work, we aim to use one object (trigger) to achieve the attack goal, and the derived object location should be able to be applied to different driving environments.

9.2 Backdoor Attack

The backdoor attacks against DNNs has been investigated by many previous works, and various attack methods have been proposed to fool the DNNs in many applications, such as image and video classification [12, 36, 47, 64], speech recognition [25, 61, 63], natural language processing [6, 8, 59], and malware detection [40, 42, 58]. There are a few works that intend to study backdoor attacks in 3D point clouds. [54] proposes to inject some triggers with optimal shape and size into the training samples to fool the point cloud classification model. [31] uses interaction trigger and orientation trigger to achieve clean-label attack.

However, these existing backdoor attacks focus on the task of point cloud classification, which is different from the task of LiDAR point cloud object detection studied in this paper. Point cloud classification aims to output a correct label of a single point cluster such as chair or airplane, while LiDAR object detection aims to detect the objects (e.g., vehicles, bicycles and pedestrians) on the road and output the information about these objects' locations, sizes, and orientations [51]. In addition, these existing backdoor attacks are only studied in the digital world. Whether these backdoor attack methods can achieve the attack goal in the physical world is not clear. In this paper, we investigate the backdoor attack against LiDAR object detection systems. We conduct extensive real-world experiments and demonstrate that the proposed attacks can be easily performed in the physical world.

10 CONCLUSIONS

In this paper, we propose the first study on the backdoor attacks against LiDAR object detection in autonomous driving. We propose a novel backdoor attack strategy, based on which the attacker can achieve the attack goal by poisoning a small number of training samples. To improve the attack stealthiness, we also propose a stealthy backdoor attack strategy that uses fake vehicle point clusters as trigger carriers and adds them to the training samples. These injected fake vehicle point clusters are hard to be detected in the training phase. We conduct extensive experiments to demonstrate the effectiveness of the proposed attacks in both the digital world and the physical world. The physical-world experiments show that the detection model poisoned by the attacker can be fooled by simply placing a common object on the roof of the target vehicle.

11 ACKNOWLEDGMENTS

We thank our anonymous reviewers for their insightful comments and suggestions on this paper. This work was supported in part by the US National Science Foundation under grant CNS-1737590, CNS-2120369, CNS-1652503, and ECCS-2028872.

REFERENCES

[1] Roger Barga, Valentine Fontama, Wee Hyong Tok, and Luis Cabrera-Cordon. 2015. *Predictive analytics with Microsoft Azure machine learning*. Springer.

[2] Chen Bian, Wei Jiang, Jinyu Zhan, Ziwei Song, Xiangyu Wen, and Hong Lei. 2021. A physically realizable backdoor attack on 3D point cloud deep learning: work-in-progress. In *Proceedings of the 2021 International Conference on Hardware/Software Codesign and System Synthesis*. 27–28.

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. 2020. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 11621–11631.

[4] Yulong Cao, Ningfei Wang, Chaowei Xiao, Dawei Yang, Jin Fang, Ruigang Yang, Qi Alfred Chen, Mingyan Liu, and Bo Li. 2021. Invisible for both camera and lidar: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 176–194.

[5] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z Morley Mao. 2019. Adversarial sensor attack on lidar-based perception in autonomous driving. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2267–2281.

[6] Kangjie Chen, Yuxian Meng, Xiaofei Sun, Shangwei Guo, Tianwei Zhang, Jiwei Li, and Chun Fan. 2021. Badpre: Task-agnostic backdoor attacks to pre-trained nlp foundation models. *arXiv preprint arXiv:2110.02467* (2021).

[7] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* (2017).

[8] Xiaoyi Chen, Ahmed Salem, Michael Backes, Shiqing Ma, and Yang Zhang. 2021. Badnl: Backdoor attacks against nlp models. In *ICML 2021 Workshop on Adversarial Machine Learning*.

[9] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. 2019. Fast point r-cnn. In *Proceedings of the IEEE/CVF international conference on computer vision*. 9775–9784.

[10] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. 2021. Voxel r-cnn: Towards high performance voxel-based 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 1201–1209.

[11] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. 2017. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1355–1361.

[12] Yu Feng, Benteng Ma, Jing Zhang, Shanshan Zhao, Yong Xia, and Dacheng Tao. 2022. FIBA: Frequency-Injection based Backdoor Attack in Medical Image Analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 20876–20885.

[13] Yansong Gao, Bao Gia Doan, Zhi Zhang, Siqi Ma, Jiliang Zhang, Anmin Fu, Surya Nepal, and Hyoungshick Kim. 2020. Backdoor attacks and countermeasures on deep learning: A comprehensive review. *arXiv preprint arXiv:2007.10760* (2020).

[14] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the 2012 IEEE conference on computer vision and pattern recognition*. IEEE, 3354–3361.

[15] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733* (2017).

[16] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. 2020. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence* 43, 12 (2020), 4338–4364.

[17] Jun Han, Madhumitha Harishankar, Xiao Wang, Albert Jin Chung, and Patrick Tague. 2017. Convoy: Physical context verification for vehicle platoon admission. In *Proceedings of the 18th International Workshop on Mobile Computing Systems and Applications*. 73–78.

[18] Chenhang He, Ruihuang Li, Shuai Li, and Lei Zhang. 2022. Voxel Set Transformer: A Set-to-Set Approach to 3D Object Detection from Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8417–8427.

[19] Hua Huang, Hongkai Chen, and Shan Lin. 2019. Magtrack: Enabling safe driving monitoring with wearable magnetics. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*. 326–339.

[20] Xinyu Huang, Peng Wang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng, and Ruigang Yang. 2019. The apolloscape open dataset for autonomous driving and its application. *IEEE transactions on pattern analysis and machine intelligence* 42, 10 (2019), 2702–2719.

[21] Jean-Pierre Hubaux, Srdjan Capkun, and Jun Luo. 2004. The security and privacy of smart vehicles. *IEEE Security & Privacy* 2, 3 (2004), 49–55.

[22] Shubham Jain and Marco Gruteser. 2018. Recognizing textures with mobile cameras for pedestrian safety applications. *IEEE Transactions on Mobile Computing* 18, 8 (2018), 1911–1923.

[23] Yunhan Jia Jia, Yantao Lu, Junjie Shen, Qi Alfred Chen, Hao Chen, Zhenyu Zhong, and Tao Wei Wei. 2020. Fooling detection alone is not enough: Adversarial attack against multiple object tracking. In *International Conference on Learning Representations (ICLR'20)*.

[24] Landu Jiang, Wen Xie, Dian Zhang, and Tao Gu. 2021. Smart diagnosis: Deep learning boosted driver inattention detection and abnormal driving prediction. *IEEE Internet of Things Journal* 9, 6 (2021), 4076–4089.

[25] Stefanos Koffas, Jing Xu, Mauro Conti, and Stjepan Picek. 2021. Can you hear it? backdoor attacks via ultrasonic triggers. *arXiv preprint arXiv:2107.14569* (2021).

[26] Hongwu Kuang, Bei Wang, Jianping An, Ming Zhang, and Zehan Zhang. 2020. Voxel-FPN: Multi-scale voxel feature aggregation for 3D object detection from LIDAR point clouds. *Sensors* 20, 3 (2020), 704.

[27] Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. 2020. Adversarial machine learning-industry perspectives. In *2020 IEEE Security and Privacy Workshops (SPW)*. IEEE, 69–75.

[28] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. 2019. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 12697–12705.

[29] Gene Lewis. 2014. Object detection for autonomous vehicles.

[30] Jiale Li, Hang Dai, Ling Shao, and Yong Ding. 2021. From voxel to point: Iou-guided 3d object detection for point cloud with voxel-to-point decoder. In *Proceedings of the 29th ACM International Conference on Multimedia*. 4622–4631.

[31] Xinke Li, Zhirui Chen, Yue Zhao, Zekun Tong, Yabang Zhao, Andrew Lim, and Joey Tianyi Zhou. 2021. PointBA: Towards Backdoor Attacks in 3D Point Cloud. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 16492–16501.

[32] Zhichao Li, Feng Wang, and Naiyan Wang. 2021. Lidar r-cnn: An efficient and universal 3d object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7546–7555.

[33] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. 2018. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European conference on computer vision (ECCV)*. 641–656.

[34] Gregory P Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K Wellington. 2019. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 12677–12686.

[35] Sumit Mund. 2015. *Microsoft azure machine learning*. Packt Publishing Ltd.

[36] Huy Phan, Yi Xie, Jian Liu, Yingying Chen, and Bo Yuan. 2022. Invisible and Efficient Backdoor Attacks for Compressed Deep Neural Networks. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 96–100.

[37] Arvind Ravulavaru. 2018. *Google Cloud AI Services Quick Start Guide: Build Intelligent Applications with Google Cloud AI Services*. Packt Publishing Ltd.

[38] Kui Ren, Qian Wang, Cong Wang, Zhan Qin, and Xiaodong Lin. 2019. The security of autonomous driving: Threats, defenses, and future directions. *Proc. IEEE* 108, 2 (2019), 357–372.

[39] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. 2020. Hidden trigger backdoor attacks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 11957–11965.

[40] Shoichiro Sasaki, Seira Hidano, Toshihiro Uchibayashi, Takuo Suganuma, Masahiro Hiji, and Shinsaku Kiyomoto. 2019. On embedding backdoor in malware detectors using machine learning. In *2019 17th International Conference on Privacy, Security and Trust (PST)*. IEEE, 1–5.

[41] Peter Schüller, João Paolo Costeira, James Crowley, Jasmin Grosinger, Félix Ingrand, Uwe Köckemann, Alessandro Saffiotti, and Martin Welss. 2022. Composing Complex and Hybrid AI Solutions. *arXiv preprint arXiv:2202.12566* (2022).

[42] Giorgio Severi, Jim Meyer, Scott Coull, and Alina Oprea. 2021. {Explanation-Guided} Backdoor Poisoning Attacks Against Malware Classifiers. In *30th USENIX Security Symposium (USENIX Security 21)*. 1487–1504.

[43] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. 2020. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10529–10538.

[44] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 2019. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–779.

[45] Shaoshuai Shi, Zhe Wang, Xiaogang Wang, and Hongsheng Li. 2019. Part-a^ 2 net: 3d part-aware and aggregation neural network for object detection from point cloud. *arXiv preprint arXiv:1907.03670* 2, 3 (2019).

[46] Fnu Suya, Jianfeng Chi, David Evans, and Yuan Tian. 2020. Hybrid batch attacks: Finding black-box adversarial examples with limited queries. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*. 1327–1344.

[47] Fnu Suya, Saeed Mahloujifar, Anshuman Suri, David Evans, and Yuan Tian. 2021. Model-targeted poisoning attacks with provable convergence. In *International Conference on Machine Learning*. PMLR, 10000–10010.

[48] Fnu Suya, Yuan Tian, David Evans, and Paolo Papotti. 2017. Query-limited black-box attacks to classifiers. *arXiv preprint arXiv:1712.08713* (2017).

[49] James Tu, Mengye Ren, Sivabalan Manivasagam, Ming Liang, Bin Yang, Richard Du, Frank Cheng, and Raquel Urtasun. 2020. Physically realizable adversarial examples for lidar object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13716–13725.

[50] Dominic Zeng Wang and Ingmar Posner. 2015. Voting for voting in online point cloud object detection.. In *Robotics: Science and Systems*, Vol. 1. Rome, Italy, 10–15.

[51] Yingjie Wang, Qiuyu Mao, Hanqi Zhu, Yu Zhang, Jianmin Ji, and Yanyong Zhang. 2021. Multi-modal 3d object detection in autonomous driving: a survey. *arXiv preprint arXiv:2106.12735* (2021).

[52] Xiangyu Wen, Wei Jiang, Jinyu Zhan, Chen Bian, and Ziwei Song. 2021. Generative strategy based backdoor attacks to 3D point clouds: work-in-progress. In *Proceedings of the 2021 International Conference on Embedded Software*. 23–24.

[53] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. 2018. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1887–1893.

[54] Zhen Xiang, David J Miller, Siheng Chen, Xi Li, and George Kesidis. 2021. A backdoor attack against 3d point cloud classifiers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7597–7607.

[55] Chaowei Xiao, Dawei Yang, Bo Li, Jia Deng, and Mingyan Liu. 2019. Meshadv: Adversarial meshes for visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6898–6907.

[56] Bin Yang, Wenjie Luo, and Raquel Urtasun. 2018. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 7652–7660.

[57] Kaichen Yang, Tzungyu Tsai, Honggang Yu, Tsung-Yi Ho, and Yier Jin. 2020. Beyond Digital Domain: Fooling Deep Learning Based Recognition System in Physical World. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1088–1095.

[58] Limin Yang, Zhi Chen, Jacopo Cortellazzi, Feargus Pendlebury, Kevin Tu, Fabio Pierazzi, Lorenzo Cavallaro, and Gang Wang. 2022. Jigsaw Puzzle: Selective Backdoor Attack to Subvert Malware Classifiers. *arXiv preprint arXiv:2202.05470* (2022).

[59] Wenkai Yang, Yankai Lin, Peng Li, Jie Zhou, and Xu Sun. 2021. Rethinking stealthiness of backdoor attack against nlp models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 5543–5557.

[60] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. 2019. Std: Sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE/CVF international conference on computer vision*. 1951–1960.

[61] Jianbin Ye, Xiaoyuan Liu, Zheng You, Guowei Li, and Bo Liu. 2022. DriNet: Dynamic Backdoor Attack against Automatic Speech Recognization Models. *Applied Sciences* 12, 12 (2022), 5786.

[62] Georgios Zamanakos, Lazaros Tsochatzidis, Angelos Amanatiadis, and Ioannis Pratikakis. 2021. A comprehensive survey of LIDAR-based 3D object detection methods with deep learning for autonomous driving. *Computers & Graphics* 99 (2021), 153–181.

[63] Tongqing Zhai, Yiming Li, Ziqi Zhang, Baoyuan Wu, Yong Jiang, and Shu-Tao Xia. 2021. Backdoor attack against speaker verification. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2560–2564.

[64] Shihao Zhao, Xingjun Ma, Xiang Zheng, James Bailey, Jingjing Chen, and Yu-Gang Jiang. 2020. Clean-label backdoor attacks on video recognition models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14443–14452.

[65] Yin Zhou and Oncel Tuzel. 2018. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4490–4499.

[66] Yi Zhu, Chenglin Miao, Foad Hajiaghajani, Mengdi Huai, Lu Su, and Chunming Qiao. 2021. Adversarial Attacks against LiDAR Semantic Segmentation in Autonomous Driving. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*. 329–342.

[67] Yi Zhu, Chenglin Miao, Tianhang Zheng, Foad Hajiaghajani, Lu Su, and Chunming Qiao. 2021. Can We Use Arbitrary Objects to Attack LiDAR Perception in Autonomous Driving?. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 1945–1960.