Hindawi
Security and Communication Networks
Volume 2021, Article ID 5578335, 13 pages
https://doi.org/10.1155/2021/5578335

Research Article

A Black-Box Attack Method against Machine-Learning-Based Anomaly Network Flow Detection Models

Sensen Guo, Jinxiong Zhao, Xiaoyu Li, Junhong Duan, Dejun Mu, and Xiao Jing

School of Cybersecurity, Northwestern Polytechnical University, Xi'an, Shaanxi 710072, China

Correspondence should be addressed to Sensen Guo; guosensen@mail.nwpu.edu.cn

Received 15 January 2021; Revised 10 February 2021; Accepted 5 March 2021; Published 24 April 2021

Academic Editor: Qing Yang

In recent years, machine learning has made tremendous progress in the fields of computer vision, natural language processing, and cybersecurity; however, we cannot ignore that machine learning models are vulnerable to adversarial examples, with some minor malicious input modifications, while appearing unmodified to human observers, the outputs of machine learning-based model can be misled easily. Likewise, attackers can bypass machine-learning-based security defenses model to attack systems in real time by generating adversarial examples. In this paper, we propose a black-box attack method against machine-learning-based anomaly network flow detection algorithms. Our attack strategy consists in training another model to substitute for the target machine learning model. Based on the overall understanding of the substitute model and the migration of the adversarial examples, we use the substitute model to craft adversarial examples. The experiment has shown that our method can attack the target model effectively. We attack several kinds of network flow detection models, which are based on different kinds of machine learning methods, and we find that the adversarial examples crafted by our method can bypass the detection of the target model with high probability.

1. Introduction

Along with the rapid development of computer technology and communication technology, the computer network is acting a more and more important role in information society nowadays, and it has already become an essential part of people's lives. Meanwhile, the rapid development of the Internet also brings about people many security problems, and how to protect the transmission of secret information on the network effectively has become a concern.

With the development of computer technology, especially the improvement of calculating speed, transferring speed, and memory capacity, machine learning (ML), especially deep learning (DL) , has developed very fast and has been widely used in many fields, such as natural language processing (NPL) [1], the Internet of things (IoT) [2, 3], computer vision (CV) [4], and time series prediction [5, 6]. In recent years, many scholars have also tried to use machine learning algorithm to solve network security detection problems.

Pervez et al. [7] proposed a filtering algorithm, which is based on a Support Vector Machine (SVM) classifier to identify malicious network intrusion on the NSL-KDD intrusion detection database; their method achieves very high classification accuracy in the training set, but the performance in the test set is not ideal.

Experimented with a wide variety of attacks and different k values, Rao et al. [8] used Indexed Partial Distance Search k-Nearest Neighbor (IKPDS) to recognize attacks. They tested their method with 12,597 samples that were randomly selected from the NSL-KDD dataset, resulting in 99.6% accuracy in their experiment.

Azad et al. [9] proposed an intrusion detection method based on the genetic algorithm and a C4.5 decision tree; they trained their model on the KDD Cup 99 dataset and got 99.89% accuracy rate and a 0.11% FAR.

Deep Belief Network (DBN) is also used by many scholars in intrusion detection, by training on 40% NSL-KDD database. Alom et al. [10] proposed a Deep Belief Network (DBN)-based intrusion detection model through a series of experiments. In their experiment, their DBN intrusion detection model achieved 97.5% accuracy after 50 iterations, and it can identify unknown attacks effectively.

Yin et al. [11] proposed the intrusion detection (RNN-IDS) model based on a cyclic neural network. They used the NSL-KDD database to evaluate the performance of their model in multi-classification and binary classification; they also tested the influence of different learning rates and the number of neurons on the performance of their model. In the binary classification experiment, the training and test accuracy of their model achieved 99.81% and 83.28%, respectively, and in the multi-classification experiment, the training and test accuracy achieved 99.53% and 81.29%, respectively.

By taking network flow data as images, Wang et al. [12] proposed an abnormal traffic classification method based on convolutional neural network (CNN); in their study, they conduct experiments in two scenarios with three types of classifiers, and their final average accuracy achieves 99.41%. Besides, many other machine-learning-based applications in cybersecurity are also introduced in [13].

Although the abovementioned developments represent great strides in many fields, machine learning has its inner shortages. Szegedy et al. [14] found that machine learning, especially deep learning, is vulnerable to adversarial examples. A machine learning (ML) or deep learning (DP) model can easily be fooled by adding some well-designed noise to the inputs. Since Szegedy et al. first discovered adversarial examples for deep learning in 2013, the academic and security communities have also realized that even the most advanced machine learning algorithms can easily be fooled by the adversarial examples, which are carefully crafted by the attackers. This will make it difficult for the machine-learning-based model to play its due role in practical applications.

The main contribution of this paper includes the following:

(i) An untargeted black-box adversarial example generation method for the machine-learning-based abnormal network flow detector is proposed in this paper.

(ii) The differences in the method of generating adversarial example between the field of computer vision and intrusion detection system are discussed in this paper.

(iii) The key points about the generate adversarial example against anomaly network flow detection are discussed in this paper.

The main notations and symbols used in this paper are listed in Table 1.

The rest of this paper is organized as follows. In Section 2, the work related to adversarial examples generate method is reviewed. Section 3 explains the key point of adversarial example generate method in the field of IDS. Section 4 details our black-box attack method toward the machine-learning-based network traffic detector. Section 5 introduces methods and the specific steps of our black-box attack method. Section 6 is the experimental results and analysis. Section 7 concludes this paper.

2. Related Work

The current adversarial example generation algorithms for machine learning are mainly concentrated in the field of computer vision. Szegedy et al. [15] first introduced the concept of adversarial examples for deep neural networks in 2014. They introduced a method named L-BFGS to generate adversarial examples, and it can be expressed as

$$\begin{aligned} &\min_{x^{\text{adv}}} c\|r\| + J_\theta\left(x^{\text{adv}}, l^{\text{adv}}\right) \\ &\text{s.t.} \quad x^{\text{adv}} \in [0,1], \end{aligned} \tag{1}$$

where c is a constant, calculated the by the line-searching method; $J(\cdot)$ is the lost function; and r is the perturbation added to the original picture. The author opined that the perturbation added to the input layer will accumulate in the process of forwarding the propagation of the neural network until it becomes large enough to cross the classification boundary.

While L-BFGS Attack uses the method of linear search to find the optimal value, it is impractical and time-consuming. Goodfellow et al. [16] proposed a fast method named the Fast Gradient Sign Method (FGSM) to generate adversarial examples in 2014; they performed only one step gradient update along with the sign of gradient at each pixel, and their method can be expressed as

$$\mathbf{x}^{\text{adv}} = \mathbf{x} + \varepsilon \text{sign}\left(\nabla_x \mathbf{J}(\boldsymbol{\theta}, \mathbf{x}, y)\right), \tag{2}$$

where x^{adv} is the adversarial example, x is the original data, and ε is the magnitude of the perturbation.

Kurakin et al. [17] proposed their method called Basic Iterative Method (BIM), which is the straightforward extension of FGSM by applying it multiple times with a small step size:

$$\begin{aligned} \mathbf{x}_0 &= \mathbf{x}, \\ \mathbf{x}_{n+1} &= \text{Clip}_{\mathbf{x},\varepsilon}\{\mathbf{x}_n + \alpha \text{sign}\left(\nabla_x \mathbf{J}(\mathbf{x}_n, y_{\text{true}})\right)\}, \end{aligned} \tag{3}$$

where $\text{Clip}_{\mathbf{x},\varepsilon}(\mathbf{A})$ denotes element-wise clipping $\mathbf{A}$, with $A_{i,j}$ clipped to the range $[x_{i,j} - \varepsilon, x_{i,j} + \varepsilon]$.

To further attack a specific class, they chose the least-likely class of the prediction and tried to maximize the cross-entropy loss. This method is referred to as the Iterative Least-Likely Class method [18]:

$$\begin{aligned} \mathbf{x}_0 &= \mathbf{x}, \\ \mathbf{x}_{n+1} &= \text{Clip}_{\mathbf{x},\varepsilon}\{\mathbf{x}_n - \alpha \text{sign}\left(\nabla_x \mathbf{J}(\mathbf{x}_n, y_{LL})\right)\}. \end{aligned} \tag{4}$$

Using this method, they fooled the neural network with a crafted adversarial example image taken from a camera successfully.

The algorithms to generate adversarial examples introduced above are all based on white-box attacks. Among the black-box attack methods, Papernot N et al. proposed a method based on a substitute model; their strategy was to train a local substitute model, which shares the same decision boundary with the target model. The dataset used for training the substitute model is generated by the attacker and

Table 1: Notations and terminology used in this paper.

Notations and symbols	Description
$\mathbf{x}$	The original data
l	The class that is labeled by the machine learning model
$\mathbf{x}^{\mathbf{adv}}$	The adversarial example
l^{adv}	The label of the adversarial examples
$J(\cdot)$	The loss function
η	The noise that is added to the original data
θ	The parameters of the machine learning model
$T(\cdot)$	The target machine learning model or deep learning model
$\text{ST}(\cdot)$	The substitute model
D	The constraint vector of the perturbation
ξ	The influence coefficients vector of the features
α	The step size of the perturbation in the single iteration

labeled by the target model. Adversarial examples are crafted using the substitute parameters, which are known to them. The adversarial examples generated by their method can not only fool the substitute model but also the target model, because both models have similar decision boundaries [19]. Beyond this, there are many other methods for generating adversarial examples, such as zeroth order optimization (ZOO) [20], one-pixel attack [21], natural GAN [22], natural evolution-strategy-based attack [23], boundary attack [24], and so on, and they have made great progress in the field of black-box adversarial example generate research. Besides, more research can be seen in [25, 26].

In the field of cybersecurity, Hu and Tan [27] performed a detailed analysis of the robustness of machine-learning-based malware algorithms. They proposed two pretense approaches under which malware can pretend to be benign and fool the detection algorithms. Grosse K et al. [28] also expanded the method that used in the field of computer version to attack Android malware detection models; on the DREBIN dataset, they achieved misclassification rates of up to 69%.

Anderson et al. [29] designed the DeepDGA, which is an extension of GAN. They tried to pseudo-randomly produce domain names that are difficult for modern DGA classifiers to detect. Their technique generates domains on a character-by-character basis and greatly exceeds the stealth of typical DGA techniques.

Using the NSL-KDD database, Yang K et al. [30] had tried to mimic the adversarial attacks against the deep neural network (DNN) model applied for NIDS in the real world, and they evaluate three different algorithms (attack based on substitute model, ZOO, and GAN) in launching adversarial attacks in the black-box model. In their work, the accuracy, precision, recall, and fscore of the target DNN model are significantly decreased under the black-box attack.

Training on the KDD Cup 99 dataset, Lin Z et al. [31] proposed IDSGAN, an improved framework of GAN against the intrusion detection system. In their study, the feasibility of the model is demonstrated to attack many detection systems with different attacks and excellent results are achieved; however, currently, the training of GAN is still unstable, and it has problems such as convergence failure and model collapse.

Although the main purpose of the adversarial attack by the adversarial example is to evade detection of the machine-learning-algorithm-based IDS system, the premise is that the adversarial examples crafted by the attacker should retain the attack function of the network behavior. Yang K et al. [30] retained the attack function by constraining the perturbations to the original attack traffic. Lin Z et al. [31] did it by keeping the functional features of each attack unchanged, but they did not further study how to limit the perturbations to make the adversarial examples conform to the physical characteristics of network traffic without distortion.

3. Adversarial Examples in the Field of IDS

Taking the classification problem as an example, generate adversarial example is usually to solve the following constrained optimization problem:

$$\min_{x'} J\left(f\left(\mathbf{x}^{\text{adv}}\right), l^{\text{adv}}\right) \quad \text{s.t.} \begin{cases} \|\eta\|_p \le \varepsilon, \\ f(\mathbf{x}) = l, \\ l \ne l^{\text{adv}}, \end{cases} \tag{5}$$

where $J(\cdot)$ is the loss function, $f(\cdot)$ is the target classification model, $\mathbf{x}$ is the original data, $\mathbf{x}^{\text{adv}}$ is the adversarial example, $l = f(\mathbf{x})$, $l^{\text{adv}} = f(\mathbf{x}^{\text{adv}})$, and η is the distance between the adversarial example $\mathbf{x}^{\text{adv}}$ and the original data $\mathbf{x}$.

As shown in Figure 1, similar to the field of computer vision, in the field of IDS, the process of adversarial example generation is to add a subtle perturbation noise to the original malicious attack traffic data, so that the attacker can successfully bypass the detection of machine learning algorithm to carry out a malicious attack on the target model. De Lucas et al. [32] introduced many key points of adversarial example for traffic data; here, we focus on two key differences of adversarial example between the field of IDS and computer vision:

(i) The direction of the noise η

(ii) The static of the noise η

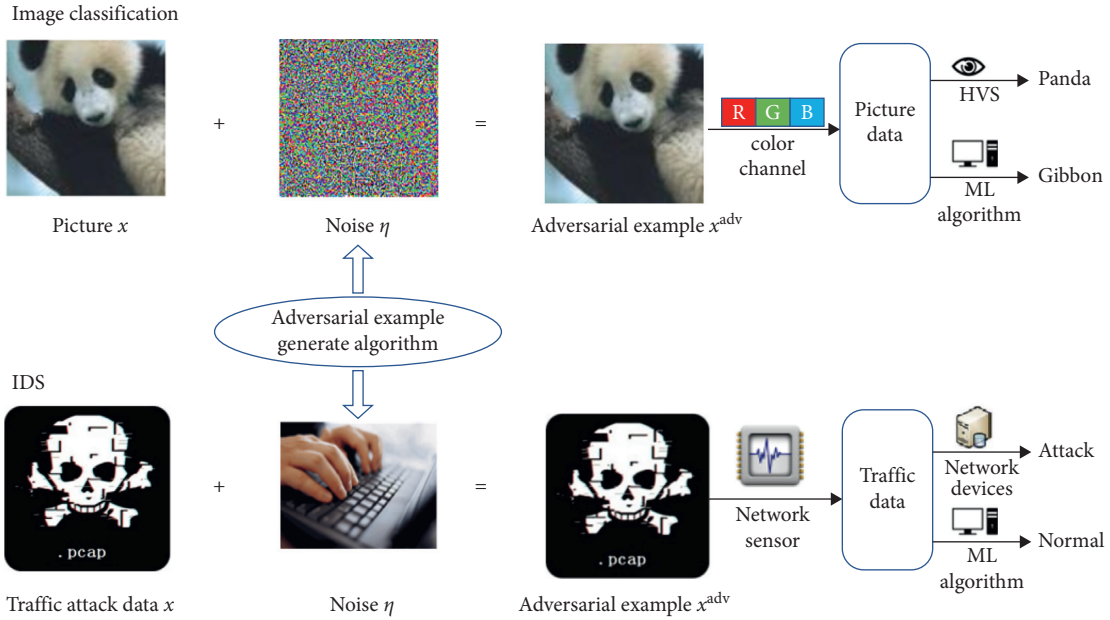

Figure 1: The differences of adversarial example generate process between IDS and computer vision.

3.1. The Direction of the Noise. In general, the process of generating adversarial example is to add appropriate amount of perturbation along with the direction of its gradient. The attacker can deceive the target model successfully by making the adversarial example cross the decision boundary of the target model; however, there is a key point that we must make sure the attack function is not lost while we add the noise to the original data.

In the field of computer vision, a picture file is composed of many pixels; each pixel is composed of three numbers, corresponding to three colors, namely, red, green, and blue, and each pixel shares the same attribute. However, in the field of intrusion detection, traffic connection data consist of an indefinite number of packets, and each packet comprises a lot of information, such as the five-tuple (the source IP, the source port, the destination IP, the destination port, and the protocol type), the packet header, and the payloads. Based on this, a variety of features, such as the protocol type, the load length, duration, the maximum message length, the minimum message length, the average message length, and so on, can be extracted for the input of machine learning model. However, unlike the picture file, each feature of the traffic connection represents different physical meanings, and some features are related to others (for example, the minimum and maximum packet length will affect the average length of the packets). Besides, a small change in the number of pixel color values has little impact on the overall picture, while for the traffic connection data, the modification of some key features may lose critical information and weaken the attack ability of the original malicious behavior, therefore, in the process of traffic adversarial examples generating, the direction of the noise that is added to the original data must be strictly controlled.

3.2. The Static of the Noise. As shown in Figure 1, in the field of computer vision, the adversarial example is still a panda in the human vision system (HVS), but after the image is converted into a digital signal on the three color channels of red, green and blue, it can successfully mislead the machine learning-based model to classify the panda as a gibbon. In order to make the adversarial example $\mathbf{x}^{adv}$ visually approximately the same as the original picture $\mathbf{x}$, the p norm $(||\eta||_p \leq \varepsilon)$ constraint is usually introduced during the generation of the adversarial example.

However, in the field of IDS, this condition is not suitable. Whether the adversarial example is similar to the original traffic and will cause an exception alarm is not ascertained through visual observation of the traffic data directly, but the network monitoring device, besides most of the machine-learning-based abnormal traffic identification methods, often extracts traffic characteristics, such as protocol type, packet length, and duration of information from traffic data, and then identify malicious behaviors based on these statistical characteristics. Normally, these statistical features correspond to different physical meanings; therefore, when calculating the distance between the adversarial example and the original sample, different statistical features should be based on different influence coefficients ξ. For example, the length of traffic packet change from 500 bytes to 510 bytes does no affect the overall traffic information, but if the protocol type changes from TCP to UDP, it means two completely different traffic data. Therefore, the constraint condition of the noise that is added to the traffic data should be described as

$$\sum_{i=1}^{i=n} ||\xi_i \cdot \eta_i||_p \leq \varepsilon, \tag{6}$$

where n is the number of traffic data features.

4. Black-Box Attack Method

In the black-box attack scenario, the attacker has no information about the structure and parameters of the target model, and the only capability of the attacker is to input the chosen data to the target model and observe results labeled by the target model. Therefore, the current mainstream method of generating adversarial examples is mainly based on the migration of the adversarial examples. As long as both models A and B are trained under similar tasks, the adversarial examples that affect one model tend to affect the other, even if the two models have completely different structure and parameters. Therefore, the attacker only needs to launch attacks on the substitute model in the white-box method and transfer the adversarial examples generate from the substitute model to the target model.

Based on the information of the structure and parameters of the substitute model, the attacker can use any white-box method to craft adversarial examples. Due to the migration of the adversarial examples, the adversarial examples that are effective for the substitute model will also be misclassified by the target model with high probability. Therefore, the black-box adversarial example generation mainly includes two processes:

(i) *Substitute Model Training*: Based on the same training task and similar database with the target model, we train a substitute model ST that shares the similar decision boundary with the target model

(ii) *Adversarial Example Generation*: The attacker uses the substitute model ST to craft adversarial examples and then checks whether the adversarial examples will be misclassified by the target model

The black-box attack on the target model is achieved through a white-box attack on the substitute model. In our paper, the white-box method that we used to create abnormal network flow adversarial example $\mathbf{x}^{\text{adv}}$ is the extension of BIM [17] and can be expressed as follows:

$$\begin{aligned} \mathbf{x}_0^{\text{adv}} &= \mathbf{x}, \\ \mathbf{x}_{N+1}^{\text{adv}} &= \mathbf{x}_N^{\text{adv}} + D \cdot \left(\alpha O\left(\nabla_{\mathbf{x}_N} \mathbf{J}\left(\theta, \mathbf{x}_N, y\right)\right)\right), \end{aligned} \tag{7}$$

where $\mathbf{x}$ is the original network flow data, α is the step size, N is the number of iterations, and D is the constraint vector. The constraint vector is used to limit $\mathbf{x}^{\text{adv}}$ always change in an allowed direction as the physical constraint of noise. $O(\mathbf{x})$ is a normalization function, which is used to convert the gradient vector into a vector with all values between [−1, 1], and it can be expressed as follows:

$$O(\mathbf{x}) = \begin{cases} \dfrac{x_i}{\max(\mathbf{x})}, & x_i > 0, \\ 0, & x_i = 0, \\ \dfrac{x_i}{|\min(\mathbf{x})|}, & x_i < 0. \end{cases} \tag{8}$$

5. Generate Abnormal Network Flow Adversarial Example

In our work, we tried to bypass the machine-learning-based abnormal network flow classifier by adding small but intentionally worst-case perturbations to data from the dataset. To achieve this, we assume that we know nothing about the structure, type, and parameters of the target model, and we can only make a limited number of query accesses to the target model.

In our paper, we first train a substitute model that has similar decision boundaries with the target classifier; then, based on the migration of the adversarial examples, we used the white-box generation method mentioned above and the substitute model to craft adversarial examples. The black-box abnormal network flow adversarial example generate process proposed in this paper is shown in Figure 2, and the main process includes the following parts:

5.1. Dataset. As shown in Figure 2, the dataset S_0 is used for training the target model and generating adversarial examples. We chose the KDD cup 99 and the CSE-CIC-IDS2018 as the datasets in our experiment. The KDD cup 99 is 9 weeks of network connection data collected from a simulated US Air Force LAN and is divided into labeled training data and unlabeled test data. In this dataset, each connection is described by 41 characteristics; among them, there are the basic characteristics of TCP connections (9 types in total), the content characteristics of TCP connections (13 types), the statistical characteristics of time-based network flow (9 types), and the host-based network flow statistics (10 types in total). As shown in Table 2, the dataset contains four attack types, there are Dos, Probing, R2l, and U2r, in the 10% subset of KDD99, DOS attacks accounted for the largest proportion of abnormal attacks, up to 98%. U2r type attacks are the least, only 22. Due to the small amount of U2r and R2L in the training set, both of them are traffic content-based attacks; therefore, in our experiment, these two types of attacks are put in one group. To balance the number of each group, we extracted 1,000 attacks from each group.

Table 2 shows the types of network attacks contained in the KDD99 dataset. The last column is the number of the attacks in the 10% dataset.

In recent years, the IDS2018 has been widely used in the research of network security. The IDS2018 is a diverse and comprehensive benchmark dataset in the field of intrusion detection, and it includes and captures network traffic and system logs of each machine, along with 80 features extracted from the captured traffic, and includes seven different attack scenarios: Heartbleed, Brute-force, Botnet, Web attacks, DoS, DDoS, and infiltration of the network from inside. In our experiment, we summarized all the attack types into: Bot, Dos, Brute, and Infiltration.

5.2. Sampling Algorithm. In the case of the black-box attack, querying the target model too many times can easily attract the attention of defenders; therefore, reducing the query

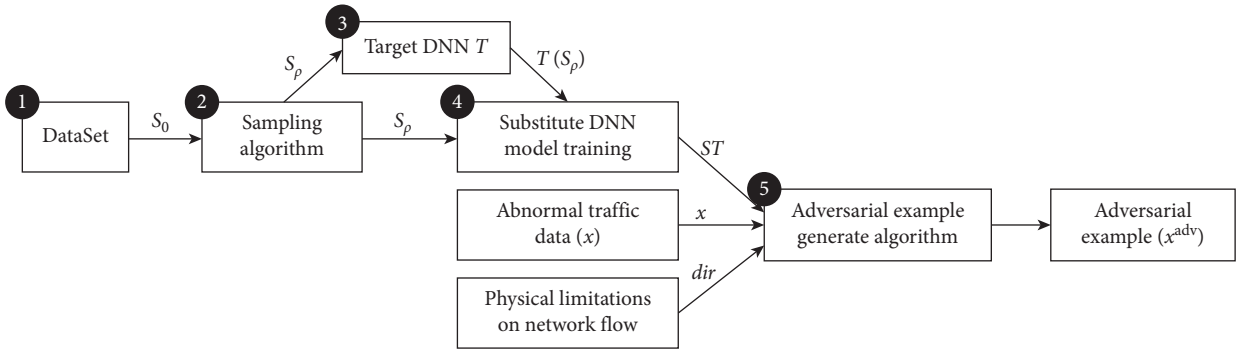

Figure 2: Abnormal network flow adversarial example generate process. We (1) choose an initial network flow training set S_0 and (2) generate subdataset S_ρ by sampling algorithm, then (3) label the S_ρ by the target DNN model and (4) train the substitute DNN model ST, finally (5) craft the adversarial example $\mathbf{x}^{adv}$ with the substitute DNN model ST, the abnormal network flow data $\mathbf{x}$, and the physical limitations on network flow dir.

Table 2: Type of malicious attacks in KDD Cup 99.

No.	Types of attack	Attacks in dataset	Quantity
1	DOS	Back, land, Neptune, pod, smurf, teardrop	391458
2	R2L	Ftpwrite, guesspasswd, imap, multihop, phf, spy, warezclient, warezmaster	1126
3	Probing	Ipsweep, nmap, portsweep, Satan	4107
4	U2R	Bufferoverflow, loadmodule, perl, rootkit	22

times to the target model as much as possible can not only improve the efficiency of black-box attacks but also is the key constraint as to whether the black-box attack method can really be implemented in the real network environment.

In this paper, similar to Papernot N et al. [33], we use the reservoir sampling algorithm to reduce the times of query to the target model. The reservoir sampling is a random sampling algorithm, the purpose of which is to select K samples from the set S that contains N items, where N is a large or unknown number. As shown in Algorithm 1, the first K samples of the set S are initially taken as the sampling result, and then go through the other samples in set S. When the i-th sample is taken, the selection strategy is to generate a random number r in the range $[0, K + i - 1]$. If r is less than K, replace the r-th sample in the sampling result set $R[K]$ to the i-th sample in the dataset S. If r is greater than or equal to K, continue the iteration. After iteration through all the data, return these K samples. This algorithm makes the probability of all samples in the set selected to be equal under the premise of only accessing the data stream once. Using the reservoir sampling algorithm can greatly reduce the number of queries to the target model and improve the training efficiency of the substitute model.

As shown in step 2 of Figure 2, the original dataset is S_0, and the subset S_ρ is obtained after the sampling algorithm. Using S_ρ as a training set, query the target model to label it as a training set, and that can be used to train the substitute model with similar decision boundaries on the limited number of the dataset.

5.3. Substitute Model Training. In the process of generating adversarial examples, we use the gradient information of the substitute model as the direction to craft adversarial example. It is required that the substitute model should have a similar decision boundary as the target model. From the point of the black-box attacker, we know nothing about the structure and parameters of the target model. However, since we can query the target model, we can estimate the approximate information of the input layer and output layer of the target model by observing the input and output of the target model; then, we can design the structure of the substitute model.

The research of Papernot N et al. [19] showed that the substitute model and the target model only need to go through a similar training process during the generation of the adversarial examples, and it is not necessary to have the same network structure and parameters. In this paper, we choose Multi-Layer Perceptron (MLP) network as the structure of the substitute model. The number of neurons of the input layer corresponds to the number of features in the traffic data, and the number of neurons of the output layer corresponds to the number of attacks of the traffic data. As shown in Figure 2, the substitute is training on the dataset S_ρ, which is the subset of the original dataset S_0 and is labeled by the target model.

5.4. Generate Adversarial Example. In the field of computer vision, the process of adversarial example generate is to find the appropriate perturbation η to satisfy the following conditions:

$$\begin{aligned} F(\mathbf{x} + \eta) &\neq F(\mathbf{x}), \\ \|\eta\|_p &\leq \varepsilon, \end{aligned} \tag{9}$$

where $F(\cdot)$ is the target model, $\mathbf{x}$ is the input picture, and η is the perturbation added to the original picture when the

Input: $S[N]$, K, where S is the sample set, N is the sample size, and K is the number of samples
Output: $R[K]$, where R is the set of sampling results
(1) set $R(K) \leftarrow X(K)$
(2) **for** $i \in [K, N-1]$ **do**
(3) $r \leftarrow$ random integer between $[0, K+i-1]$
(4) **if** $r < k$ **then**
(5) $R[r] \leftarrow S[i]$
(6) end if
(7) end for
(8) **return** $R[K]$

Algorithm 1: Reservoir sampling algorithm.

p – norm of the perturbation η is less than ε; it means that the perturbation is not perceptible to the human eye.

As mentioned above, we must ensure that the network flow adversarial examples remain in its attack function and has no key information lost, otherwise, it has no practical significance. In the field of computer vision, the modification of any pixel on the picture will not have a greater impact on the content of the picture. Therefore, whether the perturbation can be perceived by the human eye is mainly measured by calculating the p – norm of the noise η, but in the field of IDS, different characteristics have different effects on the overall network connection properties. The change of some features, such as the type of network connection, will cause the fundamental change of network connection, and changes in some features will cause the network connection information not to conform to the physical properties. Therefore, the adversarial example generation process in the network security field is subject to the following constraints:

(i) Whether the magnitude of perturbation can be detectable is not decided by a person, but by the network devices

(ii) Compared to the original connection, the adversarial example cannot lose the key information of the network connection, which determines that the direction of the perturbation η added to the original connection must be strictly restricted

(iii) The adversarial example must retain the attack function of the original connection

To address the above issues, as described above, Lin Z et al. [31] try to keep the attack function by adding unmodified features to the model, which means only add perturbation to the features with less influence. However, this method only limits some features that cannot be modified but does not limit the direction of the perturbation added to the modifiable feature. This may distort the original connection information; for example, if the original connection contains 1000 bytes of data, and the adversarial example has only 990 bytes, it will cause 10 bytes data distortion compared to the original connection.

As shown below, in this paper, we address this issue by two measures, and this is the primary content of constraint vector D in equation (7).

(i) Strictly limit the number of modifiable features in the process of adversarial examples generation. In this paper, for the features extracted from the network flow data, we only add perturbation to the noncritical features, such as the length of the packets, the duration of the connection, and the length of the package interval.

(ii) For modifiable features, the direction of perturbation added to the original connection is strictly limited to avoid information distortion. For features that can be modified, we only add positive perturbation to the original data. For instance, for the length of packets, we only add perturbation in the direction in which the length of the packet grows.

We now describe the network flow adversarial example generate process outlined in Algorithm 2, which is as follows.

(1) Initially, set the adversarial example $\mathbf{x}^{adv}$ as the original connection input $\mathbf{x}$.

(2) In the iterative process, first calculate the cross-entropy L of the original label of network flow information l and the label $\mathrm{ST}(\mathbf{x}^{adv})$, and then calculate the gradient G of L at the sample $\mathbf{x}^{adv}$.

(3) Calculate the perturbation η added in this iteration:

$$\eta = D \cdot (\alpha O(G)), \tag{10}$$

where $O(\cdot)$ is a normalization function (equation (8)). α is the step size of the sample moving along the gradient direction, the larger the α is, the greater the noise is added in a single iteration, and D is the constraint vector of the perturbation.

(4) Add the perturbation η generated in the iteration to $\mathbf{x}^{adv}$. If the substitute model is successfully deceived or the noise generated in the current iteration is 0, stop the iteration process and return the adversarial example $\mathbf{x}^{adv}$.

6. Results

6.1. Target Models. To evaluate the capacity of our model comprehensively and deeply, we first trained several typical abnormal network flow classification models based on the

Input: ST, T, $\mathbf{x}$, N, l, α, where ST is the substitute model, T is the target model, $\mathbf{x}$ is the network flow data, N is the iteration steps, l is original label, and α is the move step
Output: traffic adversarial example $\mathbf{x}^{adv}$

(1) set $\mathbf{x}^{adv} = \mathbf{x}$
(2) **for** $i \in [0, N-1]$**do**
(3) loss L = cross_entropy $(l, \mathrm{ST}(\mathbf{x}^{adv}))$
(4) gradient $G = \nabla_{\mathbf{x}^{adv}} L$
(5) perturbation $\eta = D \cdot (\alpha O(G))$, where $O(\cdot)$ is shown in equation (8)
(6) set $\mathbf{x}^{adv} = \mathbf{x}^{adv} + \eta$
(7) **if** $\mathrm{ST}(\mathbf{x}^{adv}) \neq l$ and $T(\mathbf{x}^{adv}) \neq l$ **then**
(8) break
(9) end if
(10) **if** $\mathbf{x}^{adv} = \mathbf{x}$ **then**
(11) break
(12) end if
(13) end for
(14) **return** $\mathbf{x}^{adv}$

Algorithm 2: Network flow adversarial example generate algorithm.

10% subset of the KDD99 dataset and the IDS2018 dataset, respectively. The adopted algorithms of the black-box IDS in the experiments include Convolutional Neural Networks (CNN), Support Vector Machines (SVM), k-Nearest Neighbor (KNN), Multilayer Perceptron (MLP), and the Residual Network (Resnet).

To verify the validity of the network follow adversarial example, we randomly select 1000 samples from the data of various abnormal attacks and label it with the target model. Based on these attack data, we use the method proposed in this paper to generate adversarial examples and query the classification results of the target model for these adversarial examples. Then, we use the recall rate to evaluate the effectiveness of the adversarial examples, the lower the recall rate, the more effective the adversarial examples are.

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{11}$$

where TP is the number of instances that were correctly classified and FN is the number of instances that are misclassified by the model.

6.2. Attack Based on White-Box. To verify the effectiveness of the attack method proposed in this paper, we carried out a white-box attack experiment with our method in this section. In the experiment, we chose CNN as the target model. Firstly, we used the KDD Cup 99 dataset to train a CNN-based malicious traffic detection model. Then we randomly selected 1000 records from the Dos attacks, the U2r&R2l attacks, the Probing attacks, and the Normal network connections, respectively. Finally, we used the method proposed in this paper and the 4000 records extracted from the original dataset to craft adversarial examples, we use the target CNN model to label the original data and the adversarial examples, and the confusion matrix is shown in Figure 3.

As shown in Figure 3(a), for the original dataset, the CNN-based malicious traffic detection model can make accurate classifications for different types of attacks, with an accuracy rate of about 98%, which can well complete the detection of network flow. However, for the generated adversarial examples, as we can see from Figure 3(b), the target CNN model has the highest detection accuracy for different types of network traffic and only 27.2% for a four-class detection model, which is completely unusable. The method of adversarial example generation proposed in this paper can significantly reduce the classification accuracy of the machine-learning-based network abnormal traffic detection model. For Dos attacks, about 74% of the attack connections can successfully bypass the detection of the target model, and for other types of abnormal traffic connections, the effect is similar.

During the experiment, we also adjusted the step size of the perturbation in the single iteration, and the results are shown in Figure 4. When the step size is set to 1, the mean recall rate of the network flow detector is about 33%. With the increase of the step size, when the step is above 3, the recall rate of the model stabilized at 25% or so, and increase in the step size has little effect on the success rate of adversarial example generation.

6.3. Attack Based on Black-Box

6.3.1. The Substitute Model. As mentioned above, we choose the Multi-Layer Perceptron (MLP) network as the structure of the substitute model. The fact that the substitute model and the target model have similar decision boundaries is a key point for the success rate of our method. Here, we use SCR to evaluate the similarity of decision boundaries between the substitute model and the target model, as shown below. The higher the SCR value, the more similar the decision boundaries of the substitute model and the target model.

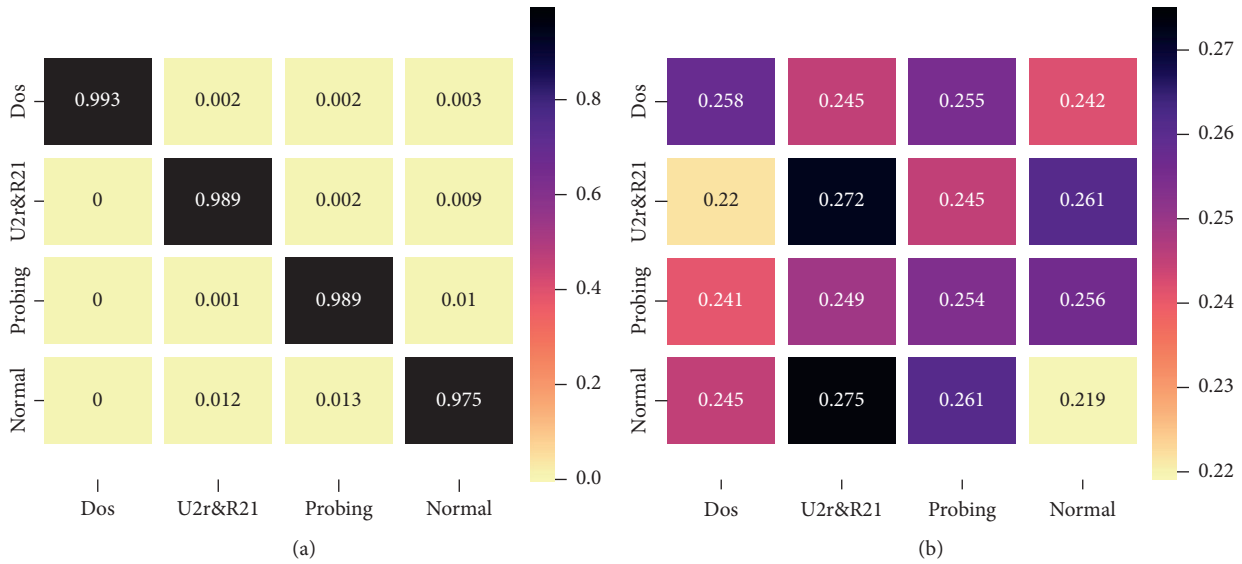

Figure 3: The confusion matrix of the target model. (a) The confusion matrix of the original traffic data. (b) The confusion matrix of the adversarial examples.

$$\text{SCR} = \frac{\text{Number of same classified by the substitute model and target model}}{\text{Total number of the special attacks}}. \tag{12}$$

For different target models, the SCR values of the substitute model for different types of attacks are shown in Figure 5. In the dataset used for substitute model training, normal network connections and Dos attacks account for a higher proportion, and its SCR values are all around 99%. On the contrary, Probing, U2l, and R2l account for a relatively low percentage in the dataset, and their SCR values are relatively low. For the U2r & R2l group with the lowest SCR, the minimum SCR value is 50% and the maximum is 70%. However, since the number is very small in the whole dataset, it has little effect on the total SCR value. In general, the substitute model still has a high similarity decision boundary with the target model.

6.3.2. Model Attack. Based on the types of black-box malicious network flow detection models that we had trained, and the method we used in this paper, for the KDD99 dataset, the attack results are shown in Figure 6. The lower Recall of the adversarial examples under various attacks reflect the great capacity of the adversarial attack in the experiments.

As shown in Figure 6(a), the mean Recall of these malicious traffic detection models is 91.8%, which means that all of these models can very well identify malicious attacks.

As shown in Figure 6(b), the average Recall of DoS under all detection algorithms is 19.8%. The results show the excellent performance of our black-box attack method in DoS. Preferably, for the case of MLP, more than 94.2% of the adversarial DoS network flow examples can evade the detection of the IDS model in each test.

For the case of the Probing, the average Recall is 32.7%. Although KNN shows better robustness, there are still a large number of malicious attacks that evade detection of the target model. On average, about 67.3% of the Probing network flow adversarial examples can bypass the detection of the target model.

And, in the worst case of U2R & E2L, the average Recall of U2R & E2L under all detection algorithms is 42.5%, which means that about 57.5% of the U2R & E2L network flow adversarial examples can evade the detection of the target model in average.

For the IDS2018 dataset, as mentioned above, we summarized all the attack types into: Bot, Dos, Brute, and Infiltration. Based on this, we trained three kinds of malicious traffic detection models: the MLP, CNN, and ResNet. The Recall of these malicious traffic detection models is shown in Figure 7(a), as we can see that all of these models can very well identify malicious attacks with a mean Recall reach of 90%. Similarly, we randomly choose 1000 samples from each malicious attack and label them with the target model. Then, we generate adversarial examples with our method, and the results are shown in Figure 7(b). For the MLP-based detection model, an average of 72.2% of malicious traffic data can successfully bypass the detection of the target model. Among them, the Dos attacks with high success rate can successfully deceive the target model with 87% probability, and the Bot attacks with low success rate also have 52.5% probability. For CNN and the ResNet-based detection model, an average of 70% and 71% of malicious traffic attack can successfully bypass the detection of the target model, respectively, and among them, 99.9% of bot attacks can successfully bypass the detection of the target

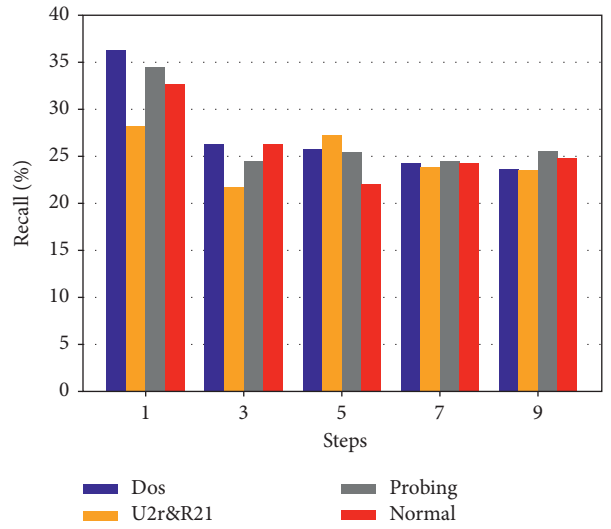

Figure 4: The impact of step size on adversarial attack.

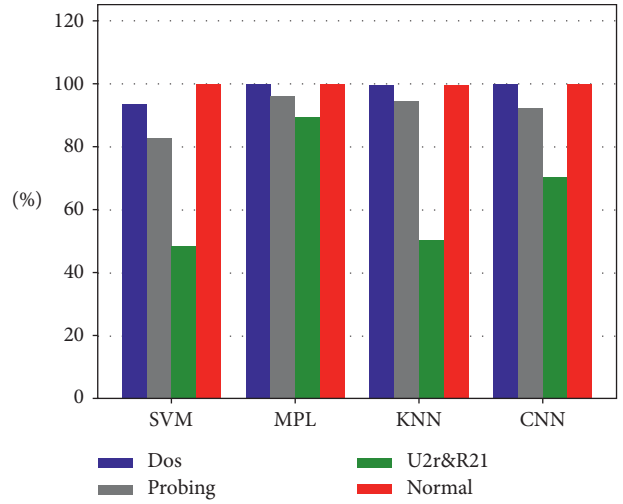

Figure 5: The SCR values of the substitute model.

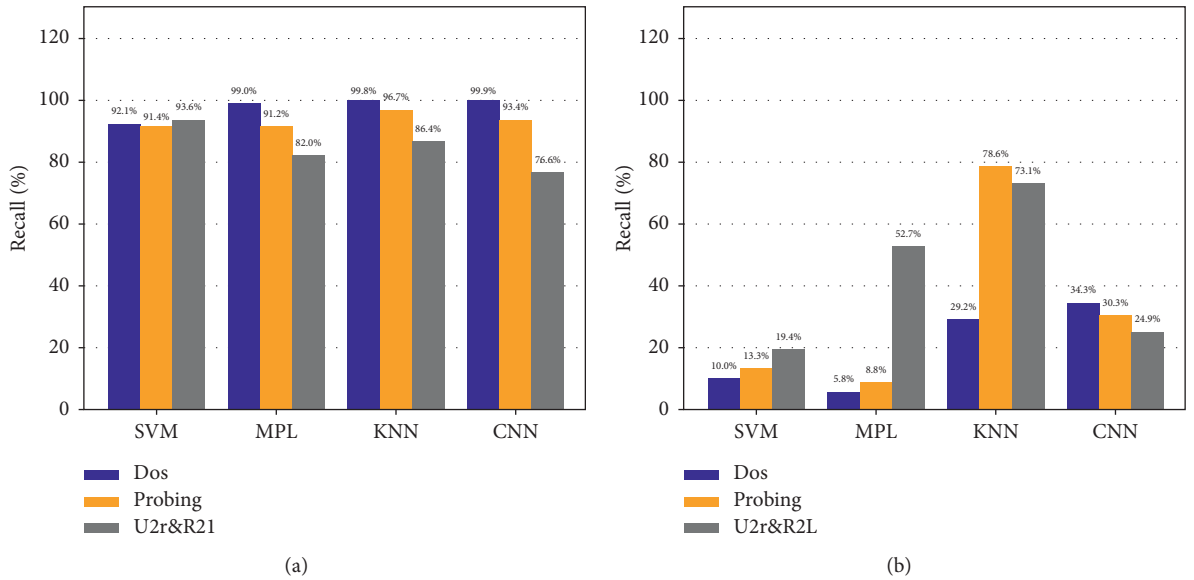

Figure 6: The recall rate of the KDD99 dataset and its adversarial examples. (a) The recall rate of the original network flow data. (b) The recall rate of the network flow adversarial example.

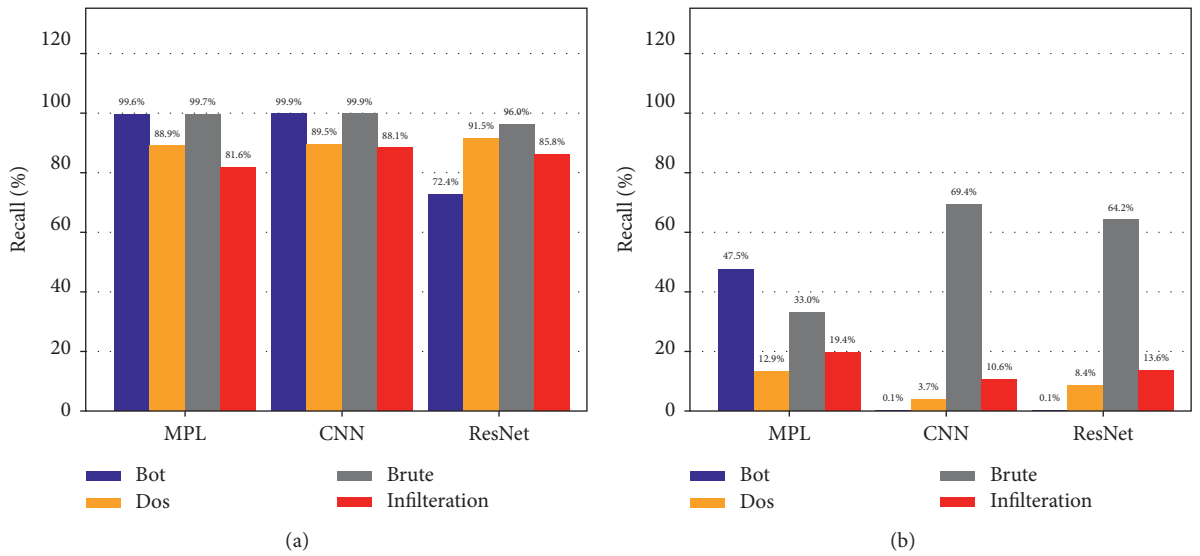

Figure 7: The recall rate of the CSE-CIC-IDS2018 dataset and its adversarial examples. (a) The recall rate of the original traffic data. (b) The recall rate of the traffic adversarial example.

model. The adversarial examples show weak attacks against Brute attacks, but more than 30% of the traffic data successfully bypass the detection of the target model.

6.3.3. Effect of Sampling Rate on Black-Box Attack. In this paper, we launch attacks on the substitute model in the white-box model and then apply the adversarial example to the target model. The success rate of this method mainly depends on the similarity of the gradient information and decision boundary between the substitute model and the target model. As shown in step 2 of Figure 2, the substitute model is trained on S_ρ which is the subset of S_0, and the sampling rate is the proportion of S_ρ in S_0. The larger the sampling rate, the closer S_ρ is to S_0, and the more likely it is that the substitute model and the target model will have similar decision boundaries. Based on this, we test on the Kdd99 dataset, and take CNN as the black-box IDS model. Different sampling rates are used in the sampling algorithm to generate network flow adversarial examples, the result of which is shown in Figure 8:

As shown in Figure 8, when the sampling rate is set to 10%, the adversarial examples for Dos can largely bypass the detection of the target model. However, the performance of the other two types is poor because DOS occupies a large proportion in the dataset. When the sampling rate is small, the proportion of probing and U2r&R2l in the subdataset used for training the substitute model will be smaller, and the substitute model cannot have very similar decision boundaries with the target model. When the sampling rate is more than 30%, the mean probability of the adversarial examples of various attacks escaping the detection of the target model does not change much. Therefore, our method can generate the network flow adversarial example effectively, even if the capacity of the dataset used for training the substitute model is relatively small.

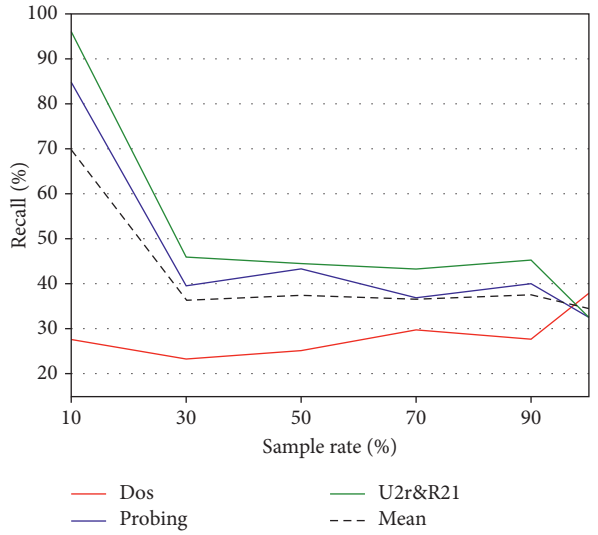

Figure 8: The effect of sampling rate on black-box attack.

6.3.4. Effect of Step Size on Black-Box Attack. As described in step 3 of Figure 2, in the process of generating abnormal network flow adversarial examples, the amount of perturbation added to the original network flow data in a single iteration depends on the gradient and the step size α. A proper step size α can quickly generate effective adversarial examples. Based on this, we test on the Kdd99 dataset, take CNN as the black-box IDS model, and then use different step sizes α to craft abnormal network flow adversarial examples, the results of which are shown in Figure 9.

As shown in Figure 9, in the process of generating abnormal network flow adversarial example, take probing as an example. When the step size changes from 1 to 17, the recall rate decreases from 85% to nearly 30%. When α is 5 or 9, the average Recall is going to be the lowest, about 33%, which means more than 67% abnormal network flow examples can bypass the

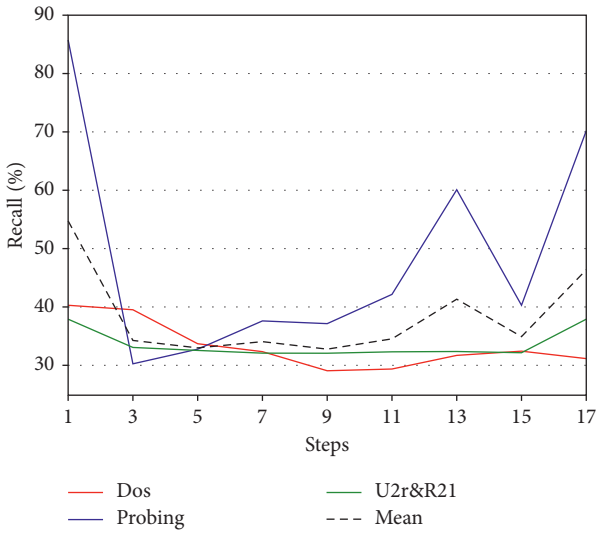

Figure 9: The effect of step size on black-box attack.

detection of the target machine-learning-based model. So it can be seen that an appropriate step size has a big influence on the success rate of adversarial example generation.

7. Conclusion

In this paper, we made a detailed comparison of the adversarial example generation technology between the field of computer vision and IDS, and we analyzed the key points and corresponding solutions for making adversarial examples in the field of IDS. Firstly, we train a substitute model with a similar decision boundary with the target model on the KDD99 dataset and the CSE-CIC-IDS2018 dataset, and then extend the BIM algorithm to craft adversarial examples with the structure and parameters of the substitute model. Finally, we check whether the adversarial examples can bypass the detection of the target model or not. Experiments show that our method can effectively generate network flow adversarial examples that can be applied to the real world and can successfully fool most of the machine-learning-based detection models.

In the future, we will further focus on the research of adversarial example technology in the field of cybersecurity. The research will concentrate on two aspects: first, we will directly apply the algorithm to real network traffic packets; Secondly, we will study the more complex malicious attack adversarial example technology based on multi-sensor data on network devices.

Data Availability

The dataset used in our paper can be made available at http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html and https://www.unb.ca/cic/datasets/ids-2018.html.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by National Key R&D Program of China (Grant No. 2020AAA0107700), in part by the Natural Science Basic Research Plan in Shaanxi Province of China (Grant No. 2020JQ-214), in part by the State Grid Gansu Electric Power Company Science and Technology Projects (Grant 52272219100Q), and in part by the Natural Science Foundation of Jiangsu Higher Education Institutions of China (Project no. 17KJB413001).

References

[1] D. Hu, "An introductory survey on attention mechanisms in NLP problems," in *Proceedings of the SAI Intelligent Systems Conference*, pp. 432–448, London, UK, September 2019.

[2] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for Internet of Things data analysis: a survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, 2018.

[3] J. Xiong, M. Zhao, M. Z. A. Bhuiyan et al., "An AI-enabled three-party game framework for guaranteed data privacy in mobile edge crowdsensing of IoT," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 922–933, 2019.

[4] L. Schmarje et al., "A survey on semi-, self-and unsupervised techniques in image classification," 2020, https://arxiv.org/abs/2002.08721.

[5] Y. Li, S. Wang, Y. Ma et al., "Popularity prediction on vacation rental websites," *Neurocomputing*, vol. 412, 2020.

[6] Y. Li, S. Wang, T. Yang, Q. Pan, and J. Tang, "Price recommendation on vacation rental websites," in *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp. 399–407, Westin Galleria Houston, TX, USA, April 2017.

[7] M. S. Pervez and D. M. Farid, "Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs," in *Proceedings of the 8th international conference on software, knowledge, information management and applications (SKIMA 2014)*, pp. 1–6, New York, NY, USA, December 2014.

[8] B. B. Rao and K. Swathi, "Fast kNN classifiers for network intrusion detection system," *Indian Journal of Science and Technology*, vol. 10, no. 14, pp. 1–10, 2017.

[9] C. Azad, V. K. Jha, and V. Kumar Jha, "Genetic algorithm to solve the problem of small disjunct in the decision tree based intrusion detection system," *International Journal of Computer Network and Information Security*, vol. 7, no. 8, p. 56, 2015.

[10] M. Z. Alom, V. R. Bontupalli, and T. M. Taha, "Intrusion detection using deep belief networks," in *Proceedings of the NAECON 2015-IEEE National Aerospace and Electronics Conference*, Dayton, OH, USA, June 2015.

[11] Y. Chuan-Long, Z. Yue-Fei, F. Jin-Long et al., "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 99, p. 1, 2017.

[12] W. Wang, M. Zhu, X. Zeng et al., "Malware traffic classification using convolutional neural network for representation learning," in *Proceedings of the 2017 International Conference on Information Networking (ICOIN)*, pp. 712–717, Da Nang, Vietnam, January 2017.

[13] Y. Xin, L. Kong, Z. Liu et al., "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, p. 1, 2018.

[14] J. H. Davis and J. R. Cogdell, *Calibration Program for the 16-Foot Antenna*, Electrical & Computer Engineering Research Laboratories, Austin, TX, USA, 1987.
[15] C. Szegedy, W. Zaremba, I. Sutskever et al., "Intriguing properties of neural networks," 2013, https://arxiv.org/abs/1312.6199.
[16] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *EnCase Computer Forensics*, vol. 6, 2014.
[17] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *Proceedings of the International Conference on Learning Representations*, Toulon, France, April 2017.
[18] A. Kurakin, I. Goodfellow, S. Bengio et al., "Adversarial examples in the physical world," in *Proceedings of the international conference on learning representations*, Toulon, France, April 2017.
[19] N. Papernot, P. Mcdaniel, I. Goodfellow et al., "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM*, Singapore, November 2017.
[20] P. Chen, H. Zhang, Y. Sharma et al., "ZOO: zeroth Order Optimization Based Black-Box Attacks to Deep Neural Networks without Training Substitute Models," *arXiv: Machine Learning*, vol. 7, pp. 15–26, 2017.
[21] J. Su, D. V. Vargas, and S. Kouichi, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, 2017.
[22] Z. Zhao, D. Dua, and S. Singh, "Generating natural adversarial examples," 2017, https://arxiv.org/abs/1710.11342.
[23] Y. Li, L. Li, L. Wang et al., "Nattack: learning the distributions of adversarial examples for an improved black-box attack on deep neural networks," in *Proceedings of the International Conference on Machine Learning. PMLR*, pp. 3866–3876, Sydney, Australia., June 2019.
[24] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: reliable attacks against black-box machine learning models," 2017, https://arxiv.org/abs/1712.04248.
[25] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: attacks and defenses for deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2805–2824, 2019.
[26] X. Liu, L. Xie, Y. Wang et al., "Privacy and security issues in deep learning: a survey," *IEEE Access*, vol. 10, 2020.
[27] W. Hu and Y. Tan, "The robustness of machine learning based malware detection algorithms," in *Proceedings of the IEEE 2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 1435–1441, Anchorage, AK, USA, April 2017.
[28] K. Grosse, N. Papernot, P. Manoharan et al., "Adversarial examples for malware detection," in *European symposium on research in computer security*, pp. 62–79, Springer, Cham, Switzerland, 2017.
[29] H. S. Anderson, J. Woodbridge, and B. Filar, "DeepDGA: Adversarially-tuned domain generation and detection," 2016.
[30] K. Yang, J. Liu, C. Zhang et al., "Adversarial examples against the deep learning based network intrusion detection systems," in *Proceedings of the Military Communications Conference*, pp. 559–564, Los Angeles, CA, USA, October 2018.
[31] Z. Lin, Y. Shi, and Z. Xue, "IDSGAN: generative adversarial networks for attack generation against intrusion detection," 2018, https://arxiv.org/abs/1809.02077.
[32] M. J. De Lucia and C. Cotton, "Adversarial machine learning for cyber security," *Journal of Information Systems Applied Research*, vol. 12, no. 1, p. 26, 2019.
[33] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," 2016, https://arxiv.org/abs/1605.07277.