SCIENCE CHINA
Information Sciences

July 2022, Vol. 65 170303:1–170303:19
https://doi.org/10.1007/s11432-021-3455-1

• RESEARCH PAPER •
Special Focus on Cyber Security in the Era of Artificial Intelligence

VulnerGAN: a backdoor attack through vulnerability amplification against machine learning-based network intrusion detection systems

Guangrui LIU[1], Weizhe ZHANG[1,2*], Xinjie LI[1], Kaisheng FAN[1] & Shui YU[3]

[1]*School of Cyberspace Science, Harbin Institute of Technology, Harbin* 150001*, China;*
[2]*Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen* 518055*, China;*
[3]*School of Computer Science, University of Technology Sydney, Ultimo* 2007*, Australia*

Received 30 August 2021/Revised 12 January 2022/Accepted 7 March 2022/Published online 23 June 2022

Abstract Machine learning-based network intrusion detection systems (ML-NIDS) are extensively used for network security against unknown attacks. Existing intrusion detection systems can effectively defend traditional network attacks, however, they face AI based threats. The current known AI attacks cannot balance the escape rate and attack effectiveness. In addition, the time cost of existing AI attacks is very high. In this paper, we propose a backdoor attack called VulnerGAN, which features high concealment, high aggressiveness, and high timeliness. The backdoor can make the specific attack traffic bypass the detection of ML-NIDS without affecting the performance of ML-NIDS in identifying other attack traffic. VulnerGAN uses generative adversarial networks (GAN) to calculate poisoning and adversarial samples based on machine learning model vulnerabilities. It can make traditional network attack traffic escape black-box online ML-NIDS. At the same time, model extraction and fuzzing test are used to enhance the convergence of VulnerGAN. Compared with the state-of-the-art algorithms, the VulnerGAN backdoor attack increases 33.28% in concealment, 18.48% in aggressiveness, and 46.32% in timeliness.

Keywords AI security, adversarial sample, data poisoning, network intrusion detection, generative adversarial network

Citation Liu G R, Zhang W Z, Li X J, et al. VulnerGAN: a backdoor attack through vulnerability amplification against machine learning-based network intrusion detection systems. Sci China Inf Sci, 2022, 65(7): 170303, https://doi.org/10.1007/s11432-021-3455-1

1 Introduction

1.1 Background

With the development of artificial intelligence technology, the defense methods of Internet devices have been upgraded in recent years [1]. Network intrusion detection systems based on machine learning algorithms (ML-NIDS) have emerged [2], which can automatically analyze network traffic and effectively identify abnormal conditions by learning malicious and benign traffic features. ML-NIDS can monitor network traffic to detect anomalous activities and prevent illegal requests for network resources. However, machine learning enhances the recognition ability of NIDS but increases the attack surface of NIDS in AI security. The primary AI based threats of ML-NIDS are data poisoning and adversarial samples.

Data poisoning. Data poisoning is an attack that affects model training by inducing machine learning algorithms to learn wrong knowledge during the model training stage [3]. The attacker interferes with the learning process of the model by adding fake malicious data to the training data, causing the model classification boundary to deviate from the actual sample distribution. Figure 1(a) is an example of a machine learning model that deforms the decision boundary due to poisoning attacks. The attacker puts poisoning samples into training data and skews the model's decision boundary, which results in the model recognizes the samples in the malicious area as benign samples.

* Corresponding author (email: wzzhang@hit.edu.cn)

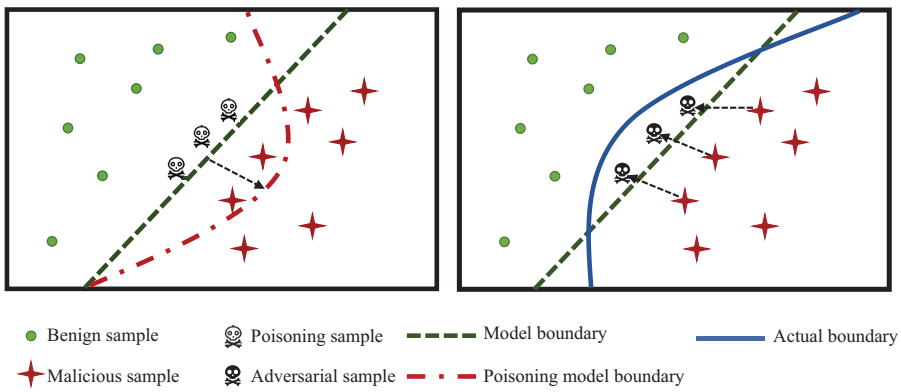

Figure 1 (Color online) Examples of AI attacks. (a) Data poisoning attacks; (b) adversarial sample attacks.

Adversarial sample. The adversarial sample is an attack that utilizes machine learning algorithm defects to affect model predictions in the model prediction stage [4]. By adding a small number of interferences to the test sample, the attacker can make the classifier misclassify without changing the target machine learning system. Figure 1(b) is an example of a machine learning model making a wrong decision due to adversarial attacks. The attacker adds interference to a specific malicious sample so that the malicious sample can be identified as a benign sample without losing the original traffic function.

1.2 Motivations and challenges

Since ML-NIDS is usually deployed in a vulnerable and open network environment, its machine learning system is exposed to attackers. However, AI security problems in online ML-NIDS have not been thoroughly studied. Existing studies [5,6] only measure ML-NIDS performance from the traditional network security perspective but lack consideration of its risks from adversarial machine learning. On the one hand, the training data of ML-NIDS come from network traffic data with unknown sources [7], which is vulnerable to data poisoning attacks. On the other hand, an attacker can generate targeted adversarial samples based on the application program interface (API) exposed by ML-NIDS, resulting in machine learning algorithms that cannot effectively identify malicious traffic.

As shown in Figure 2, traditional network attack methods are no longer valid for a host or server under the protection of ML-NIDS [8]. Emerging AI attackers need to consider bypassing ML-NIDS detection or paralyzing ML-NIDS. However, the existing attacks against ML-NIDS have insignificant effects. Using poisoning or adversarial attacks alone may not pose a severe security threat to the network system because the former does not achieve an effective network attack, and the latter still has a high probability of being identified by ML-NIDS. There is still no attack method that can achieve a high attack success rate while ensuring concealment. The time cost of generating poisoning and adversarial samples is still very high. In order to construct a more effective attack on ML-NIDS, there are three major challenges.

Concealment challenge. Concealment is the basis of AI attacks. An attack that ML-NIDS can detect is meaningless. Most of the existing studies [9–11] are dedicated to improving the escape rate of adversarial samples, but there is still much room for improvement in the evasive effectiveness of AI attacks against black-box ML-NIDS. How to further improve the concealment of AI attacks is a significant challenge.

Aggressiveness challenge. Aggressiveness is the purpose of AI attacks. However, many studies [12,13] have ignored the original attack function because of excessive attention to the concealment of adversarial samples. These researches lead to the fact that although the converted adversarial samples can bypass the detection of ML-NIDS, they also wholly lose the features of the original sample. How to improve concealment while ensuring aggressiveness is a significant challenge.

Timeliness challenge. Few studies have focused on the timeliness of AI attacks. However, in real attack scenarios, the speed of poisoning or adversarial sample generation plays a decisive role, especially for online ML-NIDS. The time cost of existing methods in generating adversarial samples is still very high [14]. How to increase the speed of sample generation while ensuring concealment and aggressiveness is a significant challenge.

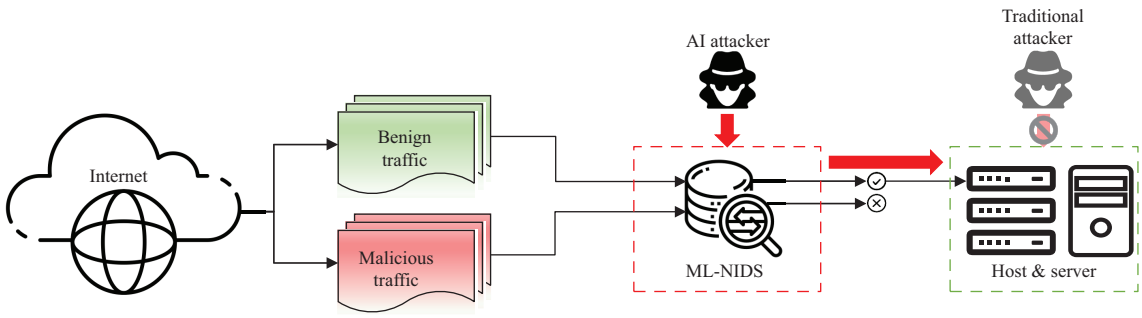

Figure 2 (Color online) The difference between traditional attacks and AI attacks against a host or server under the protection of ML-NIDS.

1.3 Contributions

In this paper, we propose a backdoor attack called VulnerGAN, which uses generative adversarial networks (GANs) to calculate poisoning and adversarial samples based on machine learning model vulnerabilities. It can make traditional network attack traffic escape black-box online ML-NIDS and enter the target host or server. The main contributions of this study are as follows.

(1) Targeting concealment challenge, we propose a new backdoor attack combining data poisoning and adversarial samples. The backdoor can make the specific attack traffic bypass the detection of ML-NIDS without affecting the performance of ML-NIDS in identifying other attacks traffic.

(2) Targeting aggressiveness challenge, we propose two generative adversarial networks named VulnerGAN-A and VulnerGAN-B that can converge against specific attacks. They can utilize machine learning model vulnerabilities to calculate poisoning and adversarial samples for specific attack traffic.

(3) Targeting timeliness challenge, we use model extraction and fuzzing test to enhance the convergence speed of VulnerGAN. Model extraction also avoids attacker frequent access to the target ML-NIDS. Through a series of experiments, the most suitable extraction algorithm for traffic recognition is selected.

The attack method proposed in this paper is tested on ML-NIDSs using different algorithms and CSE-CIC-IDS 2017 dataset. The experimental results have demonstrated that the VulnerGAN backdoor attack has improved concealment, aggressiveness, and timeliness compared with the existing methods.

(1) In concealment tests, VulnerGAN can convert various attacks traffic into poisoning and adversarial samples with better evasive effectiveness. Compared with the random mutation algorithm [9,10] and the BiGAN algorithm [11], the escape rate of various attacks increases by 33.28% on average.

(2) In aggressiveness tests, VulnerGAN can reduce the accuracy of ML-NIDSs using various algorithms. Compared with the Hydra & Neptune algorithm [12] and the GAN-adversarial algorithm [13], the accuracy of various machine learning models reduces by 18.48% on average.

(3) In timeliness tests, VulnerGAN has a faster rate of poisoning and adversarial sample generation. Compared with the GAN & particle swarm optimization (PSO) algorithm [14], the sample generation speed increases by 46.32% on average.

In addition, VulnerGAN uses model extraction and fuzzing test techniques. Even if the attacker only uses VulnerGAN-B algorithms (a part of VulnerGAN) to implement adversarial attacks without poisoning attacks, its effect is superior to existing methods in all aspects. We provide VulnerGAN source code for download[1)].

2 Related work

Since Szegedy et al. [15] discovered adversarial samples of artificial neural networks, more and more studies [16,17] have shown that machine learning models may be attacked during their training or prediction stages. The attack on the machine learning models can be divided into poisoning attacks and adversarial attacks according to the execution time [18]. The former occurs during the training phase of the model. The attacker injects maliciously poisoning samples into the training samples at this stage, causing the model decision boundary to shift. The latter occurs during the model prediction phase. The attacker

1) The source code for VulnerGAN is available for download at https://github.com/liuguangrui-hit/VulnerGAN-py.

does not need to change the target machine learning system but constructs specific samples to avoid being recognized by the model.

2.1 Data poisoning attacks

According to the model information that the attacker can detect, data poisoning attacks can be divided into white-box poisoning attacks and black-box poisoning attacks.

White-box poisoning attack. The poisoning attacker in the white-box environment knows all the model information, including the training dataset, model network architecture, and model parameters. Chung and Mok [19] studied the poisoning technology on IDS and realized allergic attacks against rule-based IDS. They made allergic nodes mistakenly believe that specific network nodes have been infected and then identify the traffic passing through these nodes as abnormal traffic. Nelson and Joseph [20] conducted a poisoning test on ML-IDS and concluded that the attacker needs exponential data to poison and destroy the model. Nevertheless, this conclusion is based on the assumption that the model's retraining window is infinite, which is not realistic in the actual process. A more realistic assumption was proposed by Rubinstein et al. [21]. They tried to adjust the non-stationarity of the data and poisoned the model by pre-injecting useless data into the training set, causing the classifier not to detect DoS attacks. Kloft and Laskov [22] poisoned IDS using the central clustering algorithm, but their algorithm needs to control 35% of the training sample data to perform effective attacks on IDS.

Black-box poisoning attack. Black-box attacks have more limitations and less model information than white-box attacks. An attacker usually needs to convert black-box environment to white-box environment before carrying out an effective attack. Li et al. [23] restricted the knowledge of the attacker. By obtaining part of the information from the training dataset, the gray-box attack on ML-NIDS was realized. They used a boundary pattern detecting algorithm to generate poisoning samples and introduced the processing batch to solve the limited number of generated samples. Li et al. [24] further improved the SMOTE algorithm. New poisoning sample data are adaptively generated according to the labeled data, which successfully reduces the target system's performance.

2.2 Adversarial sample attacks

According to the model information that the attacker can detect, adversarial sample attacks can be divided into white-box adversarial attacks and black-box adversarial attacks.

White-box adversarial attack. The white-box adversarial attack is an early research method of adversarial attack, which assumes that the attacker fully understands the internal parameters of the target model. Yuan [25] first proposed calculating adversarial samples for ML-NIDS, which is a standard FGSM algorithm. On this basis, Clements et al. [26] used various optimization algorithms to test ML-NIDS, such as the fast gradient sign method (FGSM), Jacobian-based saliency map attack (JSMA), Carlini and Wagner attack (C&W), and elastic net method (ENM). They calculated the impact of various adversarial samples on the ML-NIDS. Alhajjar et al. [27] proposed a method for generating adversarial samples using the heuristic algorithm and tested them on two public datasets NSL-KDD and UNSW-NB15. The adversarial samples generated by PSO and genetic algorithm (GA) have a higher bypass rate than optimization algorithms.

Black-box adversarial attack. The black-box adversarial attack assumes that the attacker only knows the model's output but does not understand the internal architecture of the model. Rigaki [28] generated adversarial samples against black-box ML-NIDS through GAN. They used Facebook chat traffic to train a GAN and converted the traffic passing through the attacker's server into adversarial samples. Charlier [29] proposed a network framework called SynGAN, which can modify malicious traffic samples into adversarial samples. Pan et al. [30] converted malicious traffic into images and then used pattern recognition technology to generate adversarial samples. Although their algorithm guarantees the executable of the sample, the evasion rate on ML-NIDS is poor. Han et al. [14] proposed a method of adversarial sample generation using the PSO algorithm and GAN. The adversarial samples generated by this framework can simultaneously ensure aggressiveness and concealment, but the generation time of adversarial samples is very long.

2.3 Insufficiency of existing studies

In summary, since backdoor attacks [31] on ML-NIDS have not been effectively studied, the existing attack methods still have plenty of room for improvement in concealment, aggressiveness, and timeliness. There is still no attack method that can achieve a high attack success rate while ensuring concealment. If only using the poisoning attack, although high concealment can be obtained, it does not have an actual attack effect on network equipment. If only using the adversarial attack, although high aggressiveness can be guaranteed, it also has a high probability of being recognized by ML-NIDS. In addition, the time cost of generating poisoning and adversarial samples is still very high.

Due to the above-mentioned multiple limitations, the current ML-NIDS rarely considers the security of adversarial attacks when designing and using it. However, we propose a new backdoor attack that combines data poisoning and adversarial samples, which features high concealment, aggressiveness, and timeliness. It can rapidly generate many actual network attacks without triggering ML-NIDS alerts, posing a severe security threat.

3 Problem formulation

3.1 Threat model

This subsection makes assumptions about the execution conditions of the attack method proposed in this paper.

Victim profile. The target of our work is the black-box online ML-NIDS. Compared with offline machine learning, online machine learning methods emphasize the real-time of the learning process. Each round of training will update the model parameters based on real-time data. With the emergence of new attacks, ML-NIDS needs to use a large amount of data to train and adjust parameters. In the actual network space, network data flows arrive in chronological order. Various attacks require the detection system to have the ability to adjust the non-stationarity of data in real-time. For online ML-NIDS, it is not appropriate to store all data in the model [22]. In addition, the machine learning model in ML-NIDS has the concept drift phenomenon [32], and some statistical features of network traffic will change in unforeseeable ways over time. At this time, the old samples will no longer be suitable for the new traffic classification requirements and even reduce the model's prediction accuracy. Therefore, the machine learning model in ML-NIDS is usually trained by online learning [33].

Attacker profile. We design a black-box attack against online ML-NIDS. The attacker does not need to know the internal information of the model, such as the training data, model algorithm, network architecture, or model parameters. However, the attacker needs to understand how the model collects training samples. They can also input test traffic to ML-NIDS for getting feedback. Online ML-NIDS generally obtains training samples regularly (such as 20:00 every day) or during peak network traffic periods (such as various holidays) [34], so this assumption has a practical significance. Generally, an attacker can collect data in the network or sniff relevant information about the target host. Then, it can construct and send data to the target host but not modify the data stored in the ML-NIDS [35].

3.2 Design goals

In this paper, we propose a backdoor attack through vulnerability amplification against black-box online ML-NIDSs. This method can make traditional network attack traffic escape the ML-NIDS and enter the target host or server.

As shown in Figure 3(a), when ML-NIDS runs typically, it can effectively identify and prevent malicious traffic from entering the host or server. As shown in Figure 3(b), suppose the attacker only uses the adversarial samples generator (VulnerGAN-B) proposed in this paper to convert malicious traffic into adversarial samples. In this case, some adversarial samples can escape the ML-NIDS interception and enter the host or server, but it will trigger an alert. As shown in Figure 3(c), assume that the attacker uses both the poisoning samples and the adversarial samples generators (VulnerGAN-A&B) proposed in this paper. First, the malicious traffic is converted into corresponding poisoning samples and adversarial samples. Then, put the poisoning samples into the online machine learning model to amplify the backdoor vulnerabilities. Finally, all adversarial samples generated based on malicious traffic escape ML-NIDS and enter the host or server without triggering an alert.

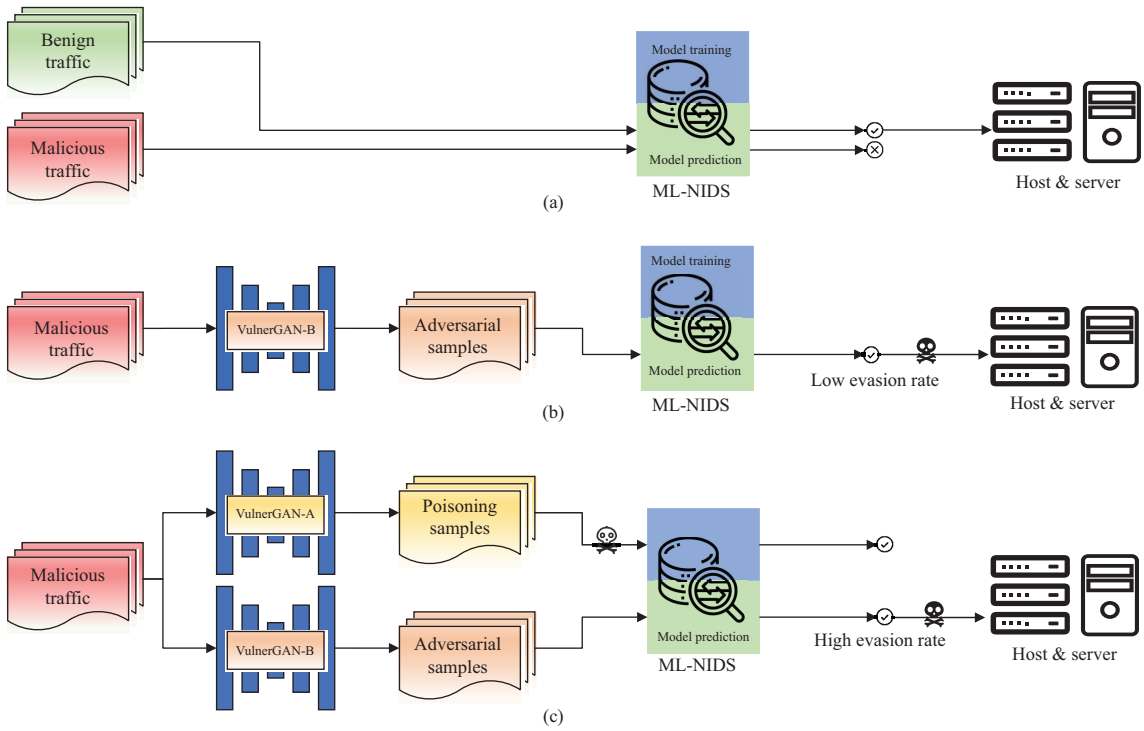

Figure 3 (Color online) The running status of ML-NIDS during (a) typical progress, (b) adversarial attacks, and (c) backdoor attacks.

Considering that the training process of some ML-NIDSs is relatively closed, we provide an attack method that directly uses adversarial samples (without constructing a backdoor). This method does not need to implement data poisoning attacks on the model to affect its training process. Due to the introduction of prior knowledge, the average evasion rate of this method is still higher than existing studies.

4 Algorithm design

In this section, for better readability, we first list the important notations of this paper in Table 1. Then, we present the detailed components of VulnerGAN. Finally, we propose the complete backdoor attack process against black-box online ML-NIDS.

4.1 Black-box model extraction

Model extraction aims to turn a black-box problem into a white-box problem. It can avoid frequent access to the target ML-NIDS, simplify the target model architecture, and accelerate the convergence speed of VulnerGAN. We apply the model extraction technology proposed by Tramér et al. [36] to the field of traffic identification. The process of model extraction is shown in Figure 4.

(1) The attacker uses the traffic sample set $S_{\text{train}} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ containing benign samples and malicious samples into the target ML-NIDS model $f : x \to \hat{y}$ and records the model's predicted label for each sample $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$ with $\hat{y}_n = f(x_n)$.

(2) The attacker uses the features of the traffic samples, and their corresponding predicted labels to form a shadow dataset S'_{train}.

(3) Use the shadow dataset S'_{train} to train the attacker's machine learning model to construct the shadow model $f' : x \to \hat{y}'$. Make the input and output distribution of the shadow model similar to the target ML-NIDS.

$$R_{\text{extract}}(f, f') = 1 - \sum_{\boldsymbol{x} \in S_{\text{test}}} \frac{d(f(\boldsymbol{x}), f'(\boldsymbol{x}))}{|S_{\text{test}}|}, \tag{1}$$

Table 1 Important notations

Symbol	Definition
$f : x \to \hat{y}$	The mapping of the machine learning model in target ML-NISD
$f' : x \to \hat{y}'$	The mapping of the shadow model generated by extracting f
$S_{\text{train}} = (\boldsymbol{x}, \boldsymbol{y})$	The traffic sample dataset for training the model in target ML-NIDS
$S'_{\text{train}} = (\boldsymbol{x}, \boldsymbol{y}')$	The shadow dataset for training the shadow model
$S_{\text{test}} = (\boldsymbol{x}_t, \boldsymbol{y}_t)$	The test dataset for calculating R_{extract} of f'
$S_{\text{vul}} = (\boldsymbol{x}_v, \boldsymbol{y}_v)$	The vulnerability dataset of the target ML-NIDS obtained by fuzzing tests
$S_{\text{mal}} = (\boldsymbol{x}_m, \boldsymbol{y}_m)$	The malicious traffic dataset that needs to be transformed into adversarial samples
$\boldsymbol{x}$, $\boldsymbol{x}_t$, $\boldsymbol{x}_m$, $\boldsymbol{x}_v$	The feature vector of S_{train}, S_{test}, S_{vul}, and S_{mal}
$\boldsymbol{y}$, $\boldsymbol{y}'$, $\boldsymbol{x}_t$, $\boldsymbol{x}_m$, $\boldsymbol{x}_v$	The label vector of S_{train}, S'_{train}, S_{test}, S_{vul}, and S_{mal}
$R_{\text{extract}}(f, f')$	The extraction rate of f'
$d(f(\boldsymbol{x}), f'(\boldsymbol{x}))$	The hamming distance between the predicted labels of f and f'
n	The number of flow-based traffic samples
z	The random noise vector conforming to normal distribution
G	The mapping of discriminator in VulnerGAN
D	The mapping of generator in VulnerGAN
$\boldsymbol{w}$	The parameter vector of VulnerGAN's discriminator
$\boldsymbol{\theta}$	The parameter vector of VulnerGAN's generator

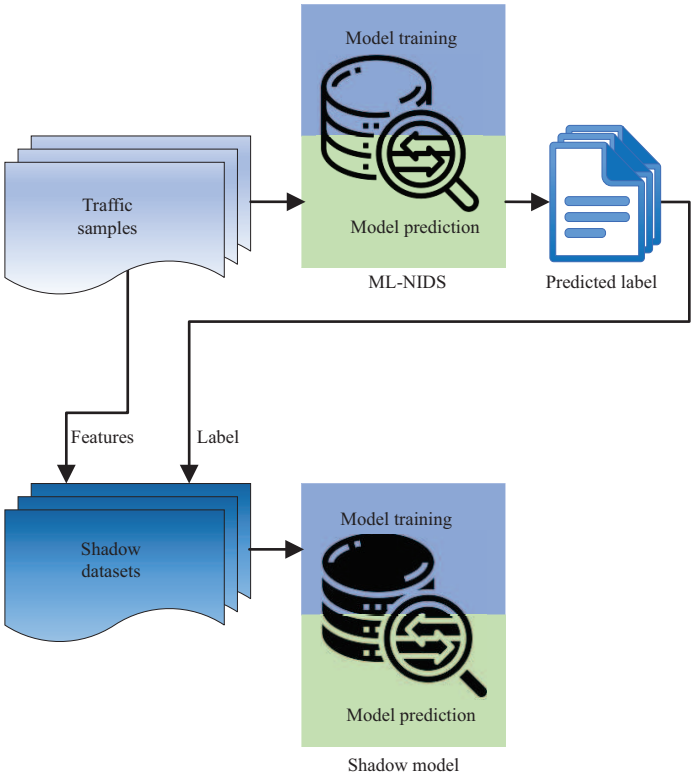

Figure 4 (Color online) Model extraction to build shadow models.

$$d(f(\boldsymbol{x}), f'(\boldsymbol{x})) = \sum_{i=1}^{n} f(x_i) \oplus f'(x_i). \tag{2}$$

In order to evaluate the extraction effect of the shadow model, Eq. (1) defines the extraction rate R_{extract} to describe the gap between the shadow model and the target model. The extraction effect of the shadow model is proportional to the extraction rate. Input the test set S_{test} into the target model f and the shadow model f', and record the two models' predicted labels $f(\boldsymbol{x})$ and $f'(\boldsymbol{x})$. The distance between the predicted labels of different models reflects the extraction effect, where the distance d is the Hamming distance (2).

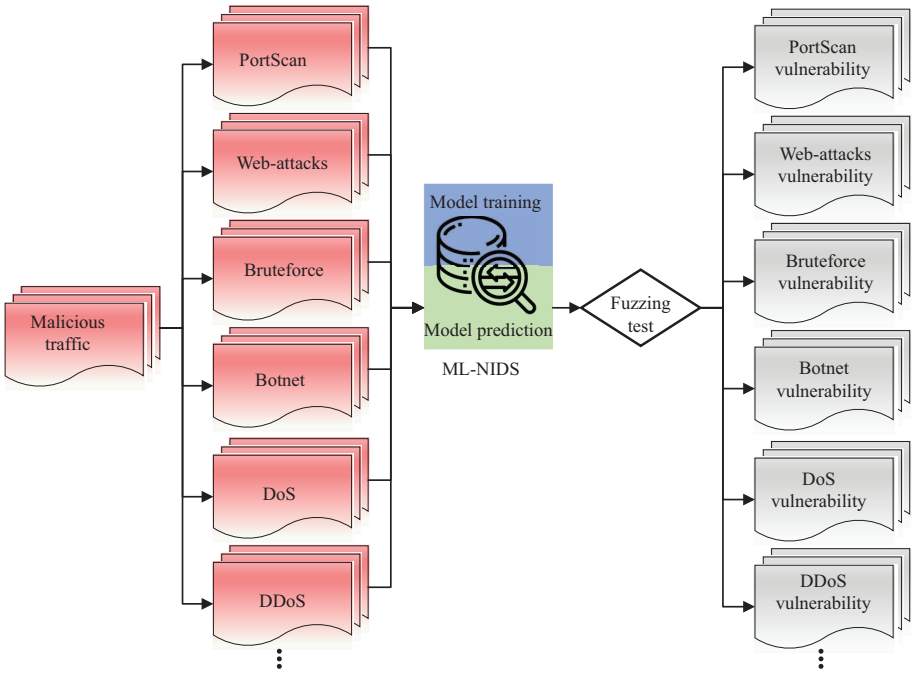

Figure 5 (Color online) Fuzzing test to collect ML-NIDS vulnerabilities.

4.2 Fuzzing test

The fuzzing test is designed to obtain vulnerabilities of the target model and help calculate poisoning and adversarial samples. Generally, online machine learning models can be deployed when they reach acceptable prediction accuracy. They cannot get all training samples at one time, nor can they have an infinite training time [37]. So even if the model is not attacked, there will be a small number of prediction errors [38]. In order to obtain these vulnerabilities of the model in identifying each type of malicious traffic, the attacker needs to fuzz test the target ML-NIDS. The process of the fuzzing test is shown in Figure 5.

(1) The malicious traffic in the dataset is divided into several categories according to the attack type, such as PortScan, Web-attacks, Bruteforce, Botnet, DoS, and DDoS.

(2) The attacker detects malicious traffic samples belonging to different attacks through ML-NIDS and records the predicted labels.

(3) Add the samples whose predicted label do not match the actual label to the vulnerability set $V = \{(x_v, y_v) \mid x_v \in S_{\text{train}} \wedge y_v \neq f(x_v)\}$, where y_v is the label of x_v in set S_{train}.

This process can effectively speed up the convergence of VulnerGAN. However, the generation algorithm does not depend on the set of S_{vul}, which means the poisoning and adversarial samples can be calculated even if there are no vulnerabilities. The algorithm also supports adding more attack types.

4.3 Generation algorithm for poisoning and adversarial samples

4.3.1 *VulnerGAN-A and VulnerGAN-B network architecture*

Goodfellow et al. [39] first proposed the GAN in 2014, which can learn the potential distribution of actual data based on training samples to generate adversarial samples. The GAN is widely used in cybersecurity [40]. In this paper, we propose two improved GAN models: VulnerGAN-A and VulnerGAN-B. VulnerGAN-A generates poisoning samples to expand model vulnerabilities and construct specific attack backdoors. VulnerGAN-B generates adversarial samples to bypass model detection and implement effective cyber-attacks.

Unlike classic GAN, VulnerGAN-A&B improves the convergence speed of the model and the quality of adversarial samples by introducing prior knowledge. VulnerGAN's training focuses on specific types of malicious traffic because the target model's structure is simplified through the shadow model f'.

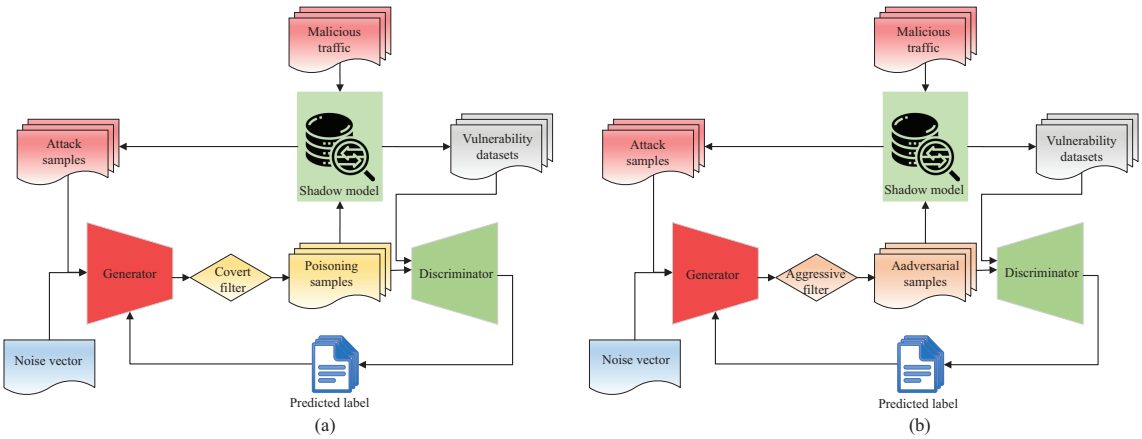

Figure 6 (Color online) (a) VulnerGAN-A generates poisoning samples; (b) VulnerGAN-B generates adversarial samples.

VulnerGAN does not need to perform a large-scale search for adversarial examples because the feature distribution of samples is initialized by S_{vul}.

Specifically, VulnerGAN-A uses the model vulnerability databases obtained from the fuzzing test to generate poisoning samples similar to the target model vulnerability in the training sample space. Poisoning samples have high concealment. Inputting them into the model training phase can cause the model to learn wrong information and generate specific attack backdoors. VulnerGAN-B uses the existing attack backdoors to generate adversarial samples in the prediction sample space that can pass through the backdoor to bypass model detection. The adversarial samples are highly aggressive and can attack the target host or server without being perceived by the ML-NIDS. The architecture of VulnerGAN-A and VulnerGAN-B is shown in Figures 6(a) and (b). They have a similar structure, including generator G, discriminator D, and the shadow model f' based on the target ML-NIDS. The optimization objective of the adversarial process is as follows:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x_v \sim p_{x_v}(x_v)} \left[\log D(x_v)\right] + \mathbb{E}_{(x_m,z)\sim p_{(x_m,z)}(x_m,z)} \left[\log(1 - D(G(x_m + z)))\right]. \quad (3)$$

x_v and x_m are the features of the vulnerability samples and the attacker's malicious samples. $p_{x_v}(x_v)$ is the distribution of x_v. $p_{(x_m,z)}(x_m,z)$ is the joint distribution of x_m and noise z. $G(x_m + z)$ is a poisoning or adversarial sample generated by generator G. The generator G in VulnerGAN is trained based on model vulnerabilities, and its goal is to generate poisoning or adversarial samples similar to the vulnerabilities. The discriminator D in VulnerGAN is dedicated to distinguishing the vulnerable samples and the poisoning/adversarial samples generated by the generator G. The following describes the training processes of generator G and discriminator D.

4.3.2 *VulnerGAN-A and VulnerGAN-B training processes*

Discriminator. The network architecture of the discriminator D is shown in Figure 7(a), which attempts to distinguish between the vulnerable samples and the poisoning/adversarial samples generated by generator G. The network's input is an n-dimensional features vector, including the features of the vulnerability samples or the poisoning/adversarial samples, plus a label reflecting whether the source is from the vulnerability database. The network's output is the predicted label of the input sample. The loss function of the discriminator D is as follows:

$$\text{Loss}_D = -\mathbb{E}_{x_v \sim p_{x_v}(x_v)} \left[\log D(x_v)\right] - \mathbb{E}_{(x_m,z)\sim p_{(x_m,z)}(x_m,z)} \left[\log(1 - D(G(x_m + z)))\right]. \quad (4)$$

Eq. (4) expresses the difference between the distribution of D's prediction result for poisoning/adversarial samples and the distribution of samples generated by G. For a particular poisoning/adversarial sample (generator G is fixed), the smaller the value of the Loss_D, the stronger the predictive ability of D.

Generator. The network architecture of generator G is shown in Figure 7(b), which generates poisoning/adversarial samples with a similar distribution of vulnerabilities. The network's input is the sum of n-dimensional malicious traffic features vector and n-dimensional random noise. The network's output

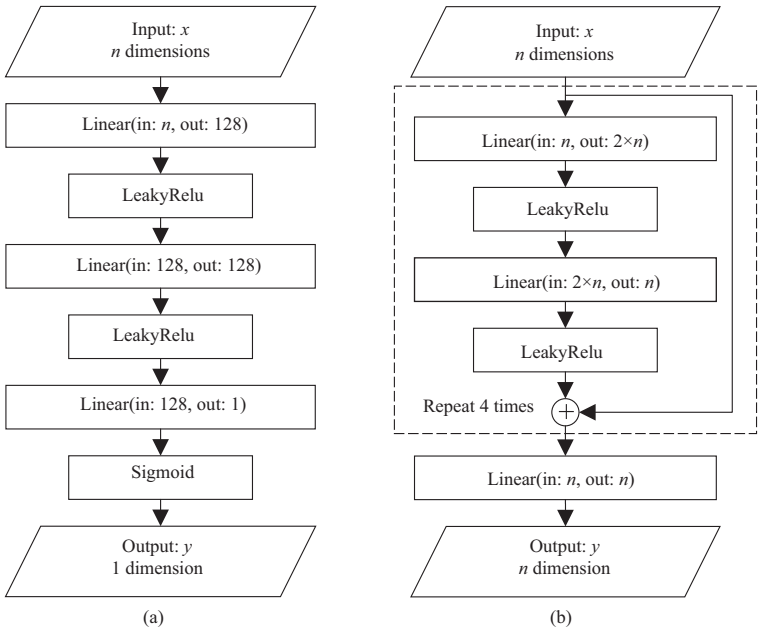

Figure 7 (a) VulnerGAN's discriminator architecture; (b) VulnerGAN's generator architecture.

is the n-dimensional features vector of the poisoning/adversarial samples. The loss function of generator G is as follows:

$$\text{Loss}_G = \mathbb{E}_{(x_m,z)\sim p_{(x_m,z)}(x_m,z)} \left[\log(1 - D(G(x_m + z)))\right]. \tag{5}$$

Eq. (5) expresses the ability of the poisoning/adversarial sample generated by G to deceive D. When the discriminator D is fixed, the smaller the value of the Loss_G, the better the quality of the poisoning/adversarial samples generated by G. The generator G is trained with ResNet [41], which can effectively solve the problem of gradient explosion and disappearance.

Algorithms 1 and 2 respectively show the training processes of VulnerGAN-A's poisoning sample generation algorithm and VulnerGAN-B's adversarial sample generation algorithm:

• **Step1.** When G is fixed, training D to better distinguish x_v and $G(x_m + z)$.

• **Step2.** When D is fixed, training G to generate $G(x_m + z)$ that is closer to x_m.

• **Step3.** The $G(x_m + z)$ generated in the process is used to supplement the traffic sample and generate more vulnerable samples x_v.

Repeat the above steps to alternate training G and D so that VulnerGAN can generate poisoning and adversarial samples similar to the target model's vulnerabilities.

The differences between VulnerGAN-A and VulnerGAN-B are as follows:

• **VulnerGAN-A** is used to generate poisoning samples. Each round of the generator's output will pick out the best poisoning samples based on concealment to help GAN converge to high-covert samples. The sample concealment is proportional to the probability that the shadow model identifies the poisoning sample as a benign sample.

• **VulnerGAN-B** is used to generate adversarial examples. Each round of the generator's output will pick out the best adversarial samples based on aggressiveness to help GAN converge to high-aggressive samples. The sample aggressiveness is proportional to the Euclidean distance between the features vector of the adversarial sample and the corresponding malicious sample.

4.4 VulnerGAN backdoor attack process and principle

Figure 8 shows the complete VulnerGAN backdoor attack process against black-box online ML-NIDS.

• **Step1.** Collect the background traffic in the network and combine the attacker's malicious traffic to form a traffic sample set S_{train}.

• **Step2.** Pass the traffic sample set S_{train} through the target ML-NIDS and obtain its return label to construct a shadow dataset S'_{train}.

• **Step3.** According to the return labels of ML-NIDS to the malicious traffic in S_{train}, the vulnerabilities of the model are exploited to form a vulnerability database S_{vul}.

Algorithm 1 The training process of VulnerGAN-A

Input: Vulnerability features $\boldsymbol{x}_v$, malicious samples features $\boldsymbol{x}_m$, discriminator parameters $\boldsymbol{w}_0$, generator parameters $\boldsymbol{\theta}_0$.
Output: Discriminator parameters $\boldsymbol{w}$, generator parameters $\boldsymbol{\theta}$.
while $\boldsymbol{\theta}$ not converging **do**
  Sample a minibatch of malicious traffic $\boldsymbol{x}_m$;
  Generate poisoning examples $G(x_m + z)$ from the generator G for x_m;
  Select poisoning examples $G(x_m + z)$ based on concealment;
  Sample a minibatch of vulnerability samples $\boldsymbol{x}_v$;
  Label $G(x_m + z)$ and x_v using the detector D to update G;
  Label $G(x_m + z)$ and x_m using the shadow model S to update $\boldsymbol{x}_v$;
  Update the detector's weights w by descending along the gradient $\nabla_w L_D$;
  Update the generator's weights θ by descending along the gradient $\nabla_\theta L_G$;
end while

Algorithm 2 The training process of VulnerGAN-B

Input: Vulnerability features $\boldsymbol{x}_v$, malicious samples features $\boldsymbol{x}_m$, discriminator parameters $\boldsymbol{w}_0$, generator parameters $\boldsymbol{\theta}_0$.
Output: Discriminator parameters $\boldsymbol{w}$, generator parameters $\boldsymbol{\theta}$.
while $\boldsymbol{\theta}$ not converging **do**
  Sample a minibatch of malicious traffic $\boldsymbol{x}_m$;
  Generate advesarial examples $G(x_m + z)$ from the generator G for x_m;
  Select adversarial examples $G(x_m + z)$ based on aggressiveness;
  Sample a minibatch of vulnerability samples $\boldsymbol{x}_v$;
  Label $G(x_m + z)$ and x_v using the detector D to update G;
  Label $G(x_m + z)$ and x_m using the shadow model S to update $\boldsymbol{x}_v$;
  Update the detector's weights w by descending along the gradient $\nabla_w L_D$;
  Update the generator's weights θ by descending along the gradient $\nabla_\theta L_G$;
end while

• **Step4.** Use shadow dataset S'_{train} to train shadow model f' for specific attack types and load shadow model S and vulnerability database S_{vul} into VulnerGAN.

• **Step5.** Use VulnerGAN-A and VulnerGAN-B to convert attack traffic into poisoning samples and adversarial samples.

• **Step6.** Put the poisoning samples into the running online ML-NIDS to interfere with the training process and form a specific attack backdoor.

• **Step7.** Upload the adversarial samples to invade the target host or server through a backdoor attack without ML-NIDS senses.

Figure 9(a) shows the distribution of benign traffic and malicious traffic in a sample space to analyze the principle of backdoor attack from topology in the sample space. At this stage, there is still vulnerability in the model that allows it to identify some malicious samples as benign samples. An attacker can obtain the vulnerabilities set of the model through fuzzing tests. Then, VulnerGAN can be used to attack the target ML-NIDS. As shown in Figure 9(b), VulnerGAN-A generates poisoning samples whose feature distribution conforms to the vulnerability samples. As shown in Figure 9(c), VulnerGAN-B is used to make the features of the malicious samples close to the vulnerability samples' features distribution. By putting the poisoning samples into the target ML-NIDS and retraining the model, the changes of the model classification boundary affected by the poisoning attack are shown in Figure 9(d). The backdoor can be generated by enlarging the model vulnerabilities to make it easier for adversarial samples escaping model recognition. Finally, the adversarial samples transformed by malicious traffic carry actual attack behaviors and enter the target host or server without triggering ML-NIDS alerts.

5 Experiments and evaluation

These experiments test the effect of VulnerGAN on a variety of machine learning algorithms. Compared with existing studies, this new backdoor attack has better concealment, aggressiveness, and timeliness performance.

5.1 Experimental environment

This subsection introduces the selection and processing of the dataset, the internal structure of the target ML-NIDS, and the hardware and software environment of the experiments. The experimental environment is shown in Table 2.

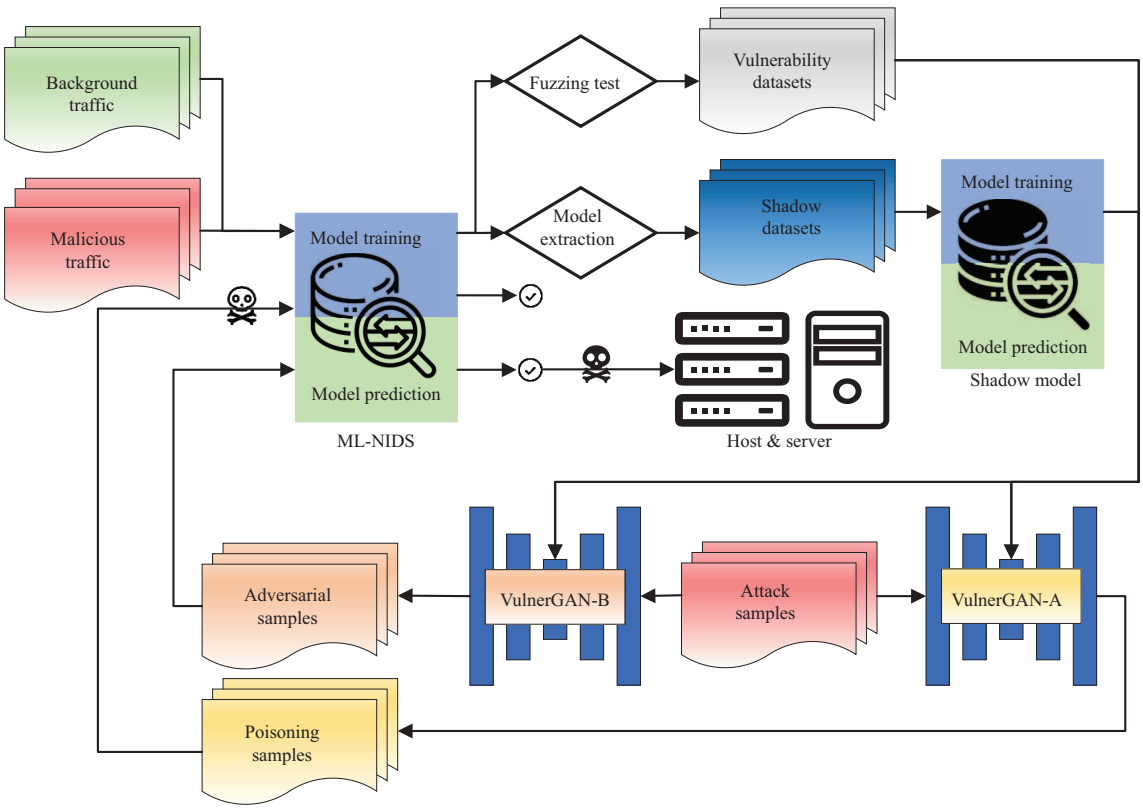

Figure 8 (Color online) The complete VulnerGAN backdoor attack process on a black-box online ML-NIDS.

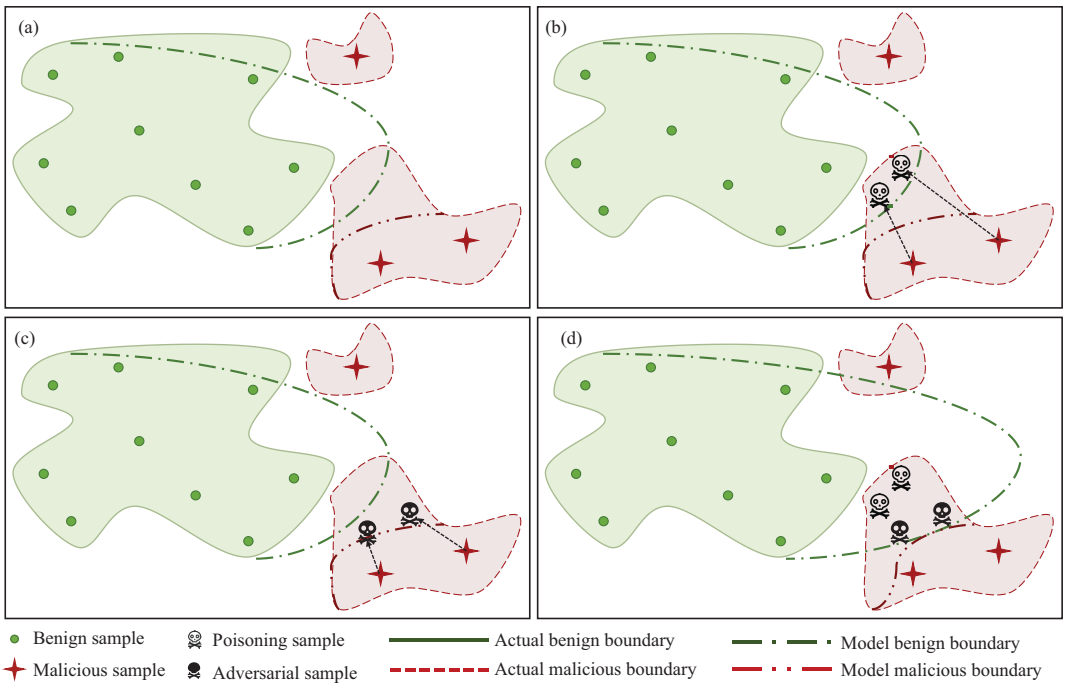

Figure 9 (Color online) VulnerGAN backdoor attack process and principle. (a) The distribution of benign traffic and malicious traffic in a sample space; (b) poisoning samples whose feature distribution conforms to the vulnerability samples; (c) adversarial samples which close to the vulnerability samples' features distribution; (d) result of the backdoor attack through vulnerability amplification against online ML-NIDS.

Dataset description. These experiments use the CSE-CIC-IDS 2017 on AWS dataset (CICIDS 2017) [42], including benign and typical attack traffic. It is a collaborative project between the Com-

Table 2 Experimental environment

Environment	Parameter
Operating system	Ubuntu 18.04.5, 64 bit
Graphics processing unit	GeForce RTX 3080
Central processing uni	Intel Xeon Silver 4210R
Programming language	Python 3.8
Dataset	CSE-CIC-IDS 2017 on AWS
Benign set data volume	2284702 samples
Malicious set data volume	538839 samples

Table 3 Statistics of query dataset

Queries	Training set	Test set	Malicious set
PortScan	571698	381132	158805
Web-attacks	7852	5234	2181
Bruteforce	49798	33199	13833
Botnet	25362	16908	7045
DoS	824216	549477	228949
DDoS	460893	307262	128026
Total	1939819	1293212	538839

munications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC). The dataset has about 2.83 million traffic samples, of which attack traffic accounts for 19.7% and covers the main attack indicated in the McAfee Labs threats report 2021 [43], including PortScan, Web-attacks, Bruteforce, Botnet, DoS, and DDoS. Each sample in the dataset has 78 features and a label.

Data preprocessing. In Experiment 1, the benign and malicious samples are uniformly and randomly distributed to the source model training set, source model test set, and shadow model test set according to the ratio of 10:1:10. In Experiments 2–4, the CICIDS 2017 dataset is divided into six attack types: PortScan, Web-attacks, Bruteforce, Botnet, DoS, and DDoS. At the same time, various attack traffic in the training set is separately extracted as a malicious sample dataset for fuzzing tests. The ratio of malicious to benign samples in the ML-NIDS training set is 1:5, and malicious samples in the training set are 60% of all the malicious samples. Statistics of query datasets are shown in Table 3.

Target ML-NIDS. We select five machine learning models commonly used in online learning: multi-layer perception (MLP), deep neural network (DNN), recurrent neural network (RNN), long-short term memory (LSTM), and gated recurrent unit (GRU). All models are designed as online ML-NIDS based on a backpropagation algorithm. The models can identify various types of attacks in the prediction phase by learning known malicious traffic in the training phase. The neural network architecture makes the models have a strong feature abstraction ability suitable for large-scale datasets.

5.2 Performance metrics

In addition to the model extraction rate defined in Subsection 4.1, accuracy rate (ACC), false-negative rate (FNR), and false-positive rate (FPR) are also the leading performance indicators of ML-NIDS. In this paper, we use these metrics to evaluate the performance of ML-NIDS under various attacks. Eqs. (6)–(8) give the specific calculation method of each metric.

$$\mathrm{ACC} = \frac{\mathrm{TF} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FN} + \mathrm{FP}}, \tag{6}$$

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{TP} + \mathrm{FN}}, \tag{7}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}}. \tag{8}$$

True-positive (TP) is the number of malicious samples correctly identified by the model. True-negative (TN) is the number of benign samples correctly identified by the model. False-positive (FP) is the number of benign samples incorrectly identified by the model. False-negative (FN) is the number of malicious samples incorrectly identified by the model.

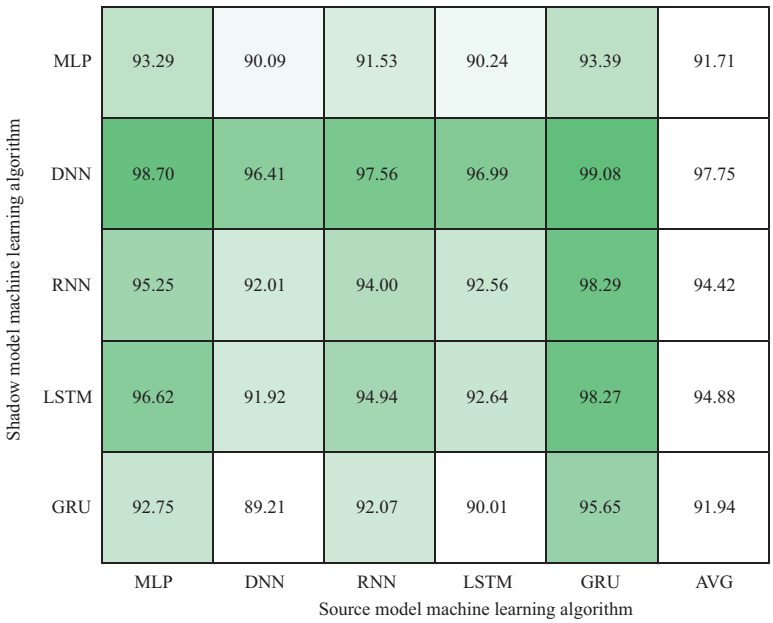

Figure 10 (Color online) Extraction rate of shadow models.

Table 4 Accuracy rate of source machine learning models

Source model	Accuracy rate on test set (%)	Accuracy rate on malicious set (%)
MLP	96.65	83.91
DNN	93.78	85.41
RNN	97.71	90.00
LSTM	97.59	92.66
GRU	97.06	90.58

5.3 Experiment results and analyses

5.3.1 *Experiment 1: model extraction and selection of the best shadow model*

Experiment 1 aims to select the most suitable algorithm for extracting machine learning models in traffic analysis. By making MLP, DNN, RNN, LSTM, and GRU models mutually extract, we test the extraction effect of each shadow model on other models.

Results. Figure 10 shows the extraction rate R_{extract} of each shadow model. The DNN shadow model has the highest extraction rate for all other models, with an average extraction rate of 97.75%. The LSTM and RNN shadow models also have strong extraction performance, with their average extraction rates reaching 94.88% and 94.42%, respectively. Table 4 compares the performance of various machine learning algorithms on traffic recognition. RNN-NIDS performed best on the test set, with an accuracy rate of 97.71%. LSTM-NIDS performed best on the malicious set, with an accuracy rate of 92.66%.

Analysis. The DNN shadow model has the best model extraction performance. Because DNN does not specialize its structure for uniqueness problems, other neural networks are subsets of fully connected DNN. The DNN model can simulate other network structures by adjusting the parameters. The ML-NIDS constructed by LSTM and RNN has a strong ability in traffic recognition because they can effectively learn the sequential features of traffic samples. Limited by the black-box attack assumption, the attacker is unclear about the machine learning algorithm used in ML-NIDS. Since DNN has the best extraction effect on other machine learning models, it constructs shadow models in subsequent experiments.

5.3.2 *Experiment 2: fuzzing test and model vulnerability databases construction*

Experiment 2 aims to test the vulnerabilities of the target ML-NIDS for various attacks. The model's predicted labels for malicious samples are recorded by inputting test samples into the source model. Malicious samples that cannot be accurately identified will form the model vulnerability databases.

Results. The ability of different models to identify various attacks and the size of each vulnerability

Table 5 Accuracy rate on malicious set and vulnerability set size of different models

Source model	PortScan		Web-attack		Bruteforce		Botnet		DoS		DDoS	
	ACC (%)	V size	ACC (%)	V size	ACC (%)	V size	ACC (%)	V size	ACC (%)	V size	ACC (%)	V size
MLP	87.75	19454	83.12	368	88.26	1624	80.83	375	81.74	41816	81.76	23348
DNN	93.35	10556	83.39	362	88.52	1588	84.00	313	80.99	43524	82.18	22816
RNN	99.64	570	83.62	357	88.45	1597	77.91	432	96.16	8785	94.23	7389
LSTM	99.65	554	91.74	180	88.27	1622	83.08	331	95.27	10827	97.96	2608
GRU	99.05	1515	83.58	358	88.56	1582	79.19	407	96.24	8607	96.84	4045

database are shown in Table 5. Most machine learning algorithms (except MLP) have more than 90% accuracy when recognizing PortScan samples. However, all models have low recognition performance for Botnet, and their accuracy rates are less than 84%. Due to the small number of Botnet samples in the CICIDS2017 dataset, the size of the Botnet attack vulnerability database is not large.

Analysis. The features of PortScan attacks are apparent, and they can be easily identified even by rule-based NIDS. Therefore, all five ML-NIDSs in the experiments identify PortScan attacks with high accuracy. However, the methods of Botnet attacks are more complicated, and the purpose of these attacks are different, so it is difficult to identify them. Although some vulnerability databases are small, the generation algorithm of VulnerGAN does not depend on the set of S_{vul}, which means the poisoning and adversarial samples can be calculated even if there are no vulnerabilities. The shadow models constructed in Experiment 1 and the vulnerability databases collected in Experiment 2 will be carried into VulnerGAN to generate poisoning and adversarial samples in Experiments 3 and 4.

5.3.3 *Experiment 3: comparisons of backdoor attacks and adversarial attacks*

Experiment 3 aims to test the impact of the specific backdoor caused by data poisoning on the attack success rate. We record the effects of different machine learning models and various network attacks on the original malicious samples, adversarial samples (generated by VulnerGAN-B), and VulnerGAN backdoors (implemented by VulnerGAN-A&B).

Results. Figure 11 shows the escape rates of original, adversarial, and backdoor samples from various attacks on different models. In all cases, the effects of backdoor attacks are better than adversarial attacks. From the perspective of ML-NIDS, the interference of VulnerGAN backdoor to LSTM-NIDS is the most obvious, and the attacks escape rate is increased from 30%–70% to 90%–100%. From the perspective of cyber-attacks, the backdoor has significantly improved Bruteforce and Botnet attacks effect, and the attacks escape rate has increased from 70%–80% to 90%–100%.

Analysis. Experiment results show that if the attacker only uses VulnerGAN-B to convert malicious traffic into adversarial samples, some of them can escape the ML-NIDS interception, but 10%–20% of the attack traffic will still be recognized. If the attacker uses VulnerGAN-A and VulnerGAN-B simultaneously, almost all adversarial samples can enter the host or server without triggering ML-NIDS alerts. Although the adversarial samples generated by VulnerGAN-B have a high evasion rate, there is still room for improvement in specific attacks (such as Botnet) or models (such as LSTM). Backdoor attacks can effectively improve the shortcomings of adversarial attacks in the above aspects. An attacker should try his best to increase the evasion rate because once an attack on ML-NIDS triggers an alert, it means that all previous efforts have been lost.

5.3.4 *Experiment 4: comparisons with related works*

Experiment 4 aims at comparing VulnerGAN with existing attack methods in terms of concealment, aggressiveness, and timeliness.

Results-1 (concealment tests). In terms of concealment, we compared the difference in evasive effectiveness between VulnerGAN algorithm, random mutation algorithm [9,10], and the state-of-the-art BiGAN algorithm [11] in Figure 12(a). In the case of only using VulnerGAN-B, our algorithm has 24.15% (PortScan), 12.33% (Web-attacks), 46.95% (Bruteforce), −27.70% (Botnet), 35.65% (DoS), and 29.81% (DDoS) improvements on different attacks. In the case of using the complete VulnerGAN framework, the evasion rate has further increased, with 24.16% (PortScan), 19.45% (Web-attacks), 67.70% (Bruteforce), 1.78% (Botnet), 38.85% (DoS), and 47.75% (DDoS) improvements in different attacks.

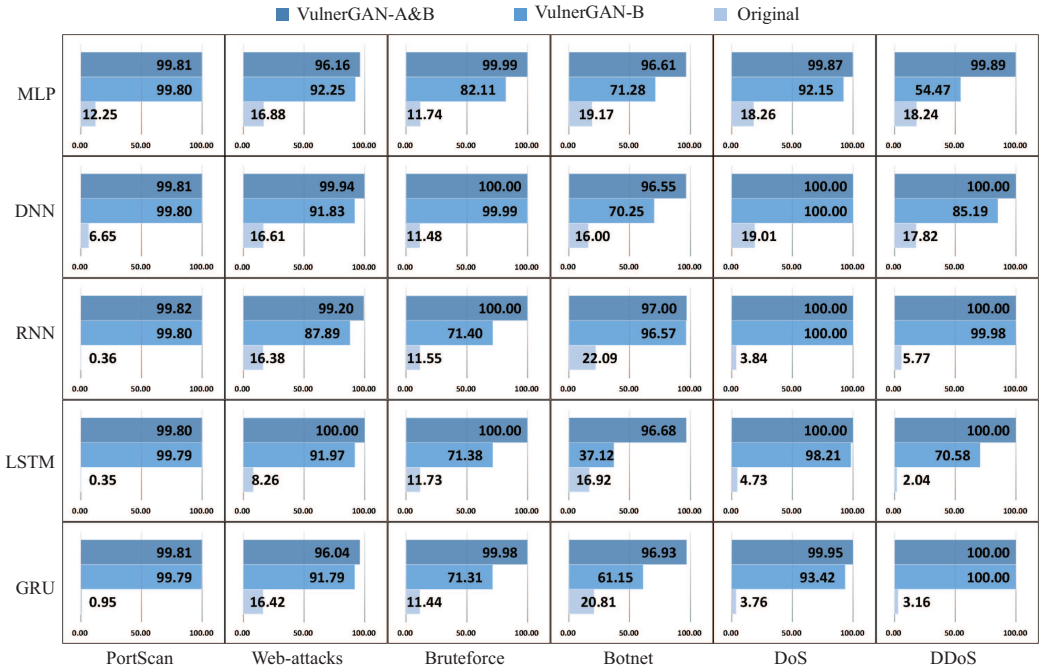

Figure 11 (Color online) The escape rate (in %) of original attacks, adversarial attacks (VulnerGAN-B), and backdoor attacks (VulnerGAN-A&B).

Analysis-1 (concealment tests). Benefit from the fuzzing test, VulnerGAN has an excellent convergence performance. Even if the attacker uses VulnerGAN-B alone, the adversarial samples have a high escape rate on most attacks (except Botnet). Furthermore, if the attacker uses the complete VulnerGAN framework, the backdoor makes all attacks reach the highest escape rate. Compared with the state-of-the-art methods, the escape rate increases by 33.28% on average. The adversarial samples generated by VulnerGAN-B for Botnet have a low evasion rate caused by Botnet traffic's complex and changeable purpose [44]. In contrast, the purpose of other malicious traffic is pronounced. The feature set of Botnet traffic is divided into multiple small cluster groups in terms of the sample distribution [45]. To solve this problem, on the one hand, when using VulnerGAN-B to calculate adversarial samples, the attacker should train malicious traffic with similar goals uniformly. On the other hand, an attacker can use VulnerGAN-A to poison model and aggregate multiple Botnet traffic sample clusters.

Results-2 (aggressiveness tests). In terms of aggressiveness, we compared the difference in model accuracy between VulnerGAN algorithm, Hydra & Neptune algorithm [12], and the state-of-the-art GAN-adversarial algorithm [13] in Figure 12(b). In the case of only using VulnerGAN-B, our algorithm reduces the accuracy of ML-NIDSs using different algorithms by 5.84% (MLP), 11.26% (DNN), 11.71% (RNN), 7.22% (LSTM), and 17.13% (GRU). In the case of using the complete VulnerGAN framework, the performance of ML-NIDSs using different algorithms is further reduced, and the accuracy rates respectively reduce by 15.86% (MLP), 16.18% (DNN), 15.75% (RNN), 19.97% (LSTM), and 24.65% (GRU).

Analysis-2 (aggressiveness tests). Benefit from model extraction, VulnerGAN has a solid ability to simulate different models. So even if the attacker uses VulnerGAN-B alone, the adversarial samples can significantly reduce the model's accuracy. Furthermore, if the attacker uses the complete VulnerGAN framework, the poisoning and adversarial attacks will further reduce the accuracy of all models. Compared with the state-of-the-art methods, the accuracy of ML-NIDS reduces by 18.48% on average.

Results-3 (timeliness tests). In terms of timeliness, we compared the difference in sample generation speed between the VulnerGAN algorithm and the state-of-the-art GAN & PSO algorithm [14]. By testing the time cost of the generation algorithm in converting all malicious samples in the Table 6, VulnerGAN's poisoning and adversarial sample generation speeds have been improved 1.78 s (PortScan), 3.74 s (Web-attacks), 2.39 s (Bruteforce), 0.73 s (Botnet), 2.08 s (DoS), and 3.12 s (DDoS) on different attacks, respectively.

Analysis-3 (timeliness tests). Benefit from the covert filter of VulnerGAN-A and the aggressive filter of VulnerGAN-B, VulnerGAN has a fast convergence and generation speed. The speed of poisoning

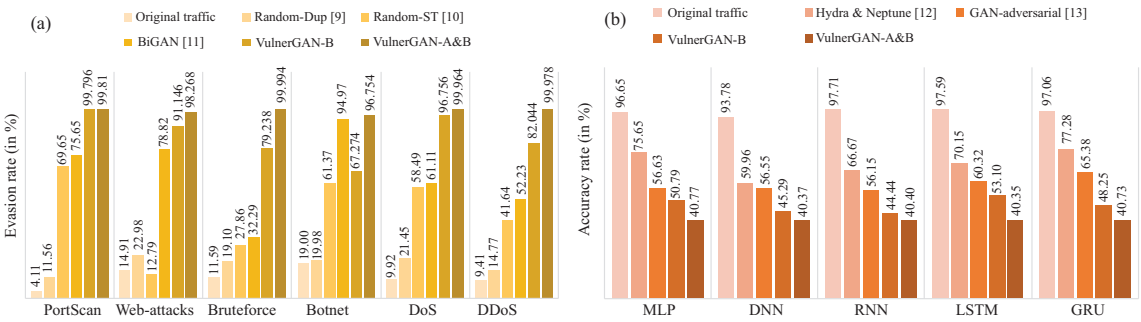

Figure 12 (Color online) (a) Evasion rate of different attacks on the ML-NIDS (concealment tests); (b) accuracy rate of different models affected by attacks (aggressiveness tests).

Table 6 Time cost (s) of the generation algorithm in converting all malicious samples (timeliness tests)

Method	PortScan	Web-attacks	Bruteforce	Botnet	DoS	DDoS
GAN & PSO [14]	6.55	3.81	2.83	0.8	8.93	6.96
VulnerGAN	4.77	0.07	0.44	0.07	6.85	3.84

and adversarial samples generation for all types of attacks has been improved, especially on Web-attacks and Botnet. Compared with the state-of-the-art methods, VulnerGAN increases the sample generation speed by 46.32% on average.

Summary. The experimental results have demonstrated that VulnerGAN backdoor attack has improved concealment, aggressiveness, and timeliness compared with the existing methods. In terms of concealment, VulnerGAN can convert various attacks traffic into adversarial samples with the best evasive effectiveness. Compared with the state-of-the-art methods, the escape rate of various attacks increases by 33.28% on average. In terms of aggressiveness, VulnerGAN can reduce the accuracy of ML-NIDSs using various algorithms. Compared with the state-of-the-art methods, the accuracy of various machine learning models reduces by 18.48% on average. In terms of timeliness, VulnerGAN has the fastest rate of poisoning and adversarial samples generation. Compared with the state-of-the-art methods, the sample generation speed increases by 46.32% on average. In addition, VulnerGAN uses model extraction and fuzzing test techniques. Even if the attacker only uses VulnerGAN-B to implement adversarial attacks without poisoning attacks, its effect is superior to existing methods in all aspects.

6 Conclusion

In this paper, we introduce the research status of AI security in ML-NIDS, point out the security risks of existing ML-NIDS, and summarize the deficiencies of existing attack methods in concealment, aggressiveness, and timeliness. Based on this, we propose a backdoor attack through vulnerability amplification against black-box online ML-NIDSs. This method is based on two generative adversarial networks: VulnerGAN-A and VulnerGAN-B. VulnerGAN combines poisoning attacks and adversarial attacks by exploiting machine learning model vulnerability. The backdoor attack can make traditional network attack traffic bypass ML-NIDS identification. VulnerGAN is tested on ML-NIDSs using different algorithms and CSE-CIC-IDS 2017 dataset. Compared with the state-of-the-art algorithms, VulnerGAN backdoor attack increases 33.28% in concealment, 18.48% in aggressiveness, and 46.32% in timeliness.

For future research, we will continue to study how to convert traffic data into poisoning traffic and adversarial traffic in the practical network and achieve end-to-end transparent traffic conversion. Finally, we hope that the producer and maintainer of ML-NIDS will pay more attention to adversarial attacks through this work. It is necessary to add protection to the model and fully attack tests when designing ML-NIDS. Otherwise, machine learning will increase the attack surface of the system.

Acknowledgements This work was supported in part by National Key Research and Development Program of China (Grant No. 2020YFB1406902), Key-Area Research and Development Program of Guangdong Province (Grant No. 2020B0101360001), Shenzhen Science and Technology Research and Development Foundation (Grant No. JCYJ20190806143418198), National Natural Science Foundation of China (Grant No. 61872110), Fundamental Research Funds for the Central Universities (Grant No. HIT.OCEF.2021007), and Peng Cheng Laboratory Project (Grant No. PCL2021A02).

References

1 Yang J, Johansson T. An overview of cryptographic primitives for possible use in 5G and beyond. Sci China Inf Sci, 2020, 63: 220301
2 Dong Y, Zhang Y Q, Ma H, et al. An adaptive system for detecting malicious queries in web attacks. Sci China Inf Sci, 2018, 61: 032114
3 Zhang X Z, Zhu X J, Lessard L. Online data poisoning attacks. In: Proceedings of Learning for Dynamics and Control (PMLR), 2020. 201–210
4 Lin J Y, Xu L, Liu Y Q, et al. Black-box adversarial sample generation based on differential evolution. J Syst Softw, 2020, 170: 110767
5 Mirsky Y, Doitshman T, Elovici Y, et al. Kitsune: an ensemble of autoencoders for online network intrusion detection. Mach Learn, 2018, 5: 2
6 Elmasry W, Akbulut A, Zaim A H. Evolving deep learning architectures for network intrusion detection using a double PSO metaheuristic. Comput Netw, 2020, 168: 107042
7 Ring M, Wunderlich S, Scheuring D, et al. A survey of network-based intrusion detection data sets. Comput Secur, 2019, 86: 147–167
8 Nasr M, Bahramali A, Houmansadr A. Defeating DNN-based traffic analysis systems in real-time with blind adversarial perturbations. In: Proceedings of the 30th USENIX Security Symposium, 2021. 2705–2722
9 Stinson E, Mitchell J C. Towards systematic evaluation of the evadability of bot/botnet detection methods. In: Proceedings of the 2nd Conference on USENIX Workshop on Offensive Technologies (WOOT'08), San Jose, 2008. 1–9
10 Homoliak I, Teknos M, Ochoa M, et al. Improving network intrusion detection classifiers by non-payload-based exploit-independent obfuscations: an adversarial approach. 2018. ArXiv:1805.02684
11 Hashemi M J, Cusack G, Keller E. Towards evaluation of nidss in adversarial setting. In: Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks, Orlando, 2019. 14–21
12 Aiken J, Scott-Hayward S. Investigating adversarial attacks against network intrusion detection systems in sdns. In: Proceedings of IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Dallas, 2019. 1–7
13 Usama M, Asim M, Latif S, et al. Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems. In: Proceedigns of the 15th International Wireless Communications & Mobile Computing Conference (IWCMC), Tangier, 2019. 78–83
14 Han D Q, Wang Z L, Zhong Y, et al. Practical traffic-space adversarial attacks on learning-based nidss. 2020. ArXiv:2005.07519
15 Szegedy C, Zaremba W, Sutskever I, et al. Intriguing properties of neural networks. In: Proceedigns of the 2nd International Conference on Learning Representations (ICLR 2014), Banff, 2014
16 Khamis R A, Shafiq M O, Matrawy A. Investigating resistance of deep learning-based IDS against adversaries using min-max optimization. In: Proceedigns of IEEE International Conference on Communications (ICC), 2020. 1–7
17 Duy P T, Tien L K, Khoa N H, et al. DIGFuPAS: deceive IDS with GAN and function-preserving on adversarial samples in SDN-enabled networks. Comput Secur, 2021, 109: 102367
18 Ozdag M. Adversarial attacks and defenses against deep neural networks: a survey. Procedia Comput Sci, 2018, 140: 152–161
19 Chung S P, Mok A K. Allergy attack against automatic signa-ture generation. In: Proceedings of International Workshop on Recent Advances in Intrusion Detection, 2006. 61–80
20 Nelson B, Joseph A D. Bounding an attack's complexity for a simple learning model. In: Proceedigns of the 1st Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML), Saint-Malo, 2006
21 Rubinstein B I P, Nelson B, Huang L, et al. Antidote: un-derstanding and defending against poisoning of anomaly detec-tors. In: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, Chicago, 2009. 1–14
22 Kloft M, Laskov P. Online anomaly detection under adver-sarial impact. In: Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, 2010. 405–412
23 Li P, Liu Q, Zhao W T, et al. Chronic poisoning against machine learning based IDSs using edge pattern detection. In: Proceedings of IEEE International Conference on Communications (ICC), Kansas City, 2018. 1–7
24 Li P, Zhao W T, Liu Q, et al. Poisoning machine learning based wireless IDSs via stealing learning model. In: Proceedings of International Conference on Wireless Algorithms, Systems, and Applications, Tianjin, 2018. 261–273
25 Yuan X, He P, Zhu Q, et al. Adversarial examples: attacks and defenses for deep learning. IEEE Trans Neural Netw Learn Syst, 2019, 30: 2805–2824
26 Clements J, Yang Y, Sharma A, et al. Rallying adversarial techniques against deep learning for network security. 2019. ArXiv:1903.11688
27 Alhajjar E, Maxwell P, Bastian N D. Adversarial machine learning in network intrusion detection systems. 2020. ArXiv:2004.11898
28 Rigaki M, Garcia S. Bringing a GAN to a knife-fight: adapting malware communication to avoid detection. In: Proceedings of IEEE Security and Privacy Workshops (SPW), San Francisco, 2018. 70–75
29 Charlier J, Singh A, Ormazabal G, et al. SynGAN: towards generating synthetic network attacks using GANs. 2019. ArXiv:1908.09899
30 Pan Y M, Lin J J. Generation and verification of malicious network flow based on generative adversarial networks. Chem J

Chinese U, 2019, 45: 344–350
31 Gu T, Dolan-Gavitt B, Garg S. Badnets: identifying vulnerabilities in the machine learning model supply chain. 2017. ArXiv:1708.06733
32 Parveen P, Weger Z R, Thuraisingham B, et al. Supervised learning for insider threat detection using stream mining. In: Proceedings of the 23rd International Conference on Tools with Artificial Intelligence, Boca Raton, 2011. 1032–1039
33 Park S, Seo S, Jeong C, et al. Online eigenvector transformation reflecting concept drift for improving network intrusion detection. Expert Syst, 2020, 37: 12477
34 Li G X, Shen Y L, Zhao P, et al. Detecting cyberattacks in industrial control systems using online learning algorithms. Neurocomputing, 2019, 364: 338–348
35 Anthi E, Williams L, Rhode M, et al. Adversarial attacks on machine learning cybersecurity defences in industrial control systems. J Inf Secur Appl, 2021, 58: 102717
36 Tramèr F, Zhang F, Juels A, et al. Stealing machine learning models via prediction APIS. In: Proceedings of the 25th USENIX Security Symposium, Austin, 2016. 601–618
37 Salem A, Bhattacharya A, Backes M, et al. Updates-leak: data set inference and reconstruction attacks in online learning. In: Proceedings of the 29th USENIX Security Symposium, 2020. 1291–1308
38 Liang J, Ma M, Sadiq M, et al. A filter model for intrusion detection system in vehicle ad hoc networks: a hidden Markov methodology. Knowl-Based Syst, 2019, 163: 611–623
39 Goodfellow I J, Pouget-Abadie J, Mirza M, et al. Generative adversarial networks. 2014. ArXiv:1406.2661
40 Qu Y Y, Zhang J W, Li R D, et al. Generative adversarial networks enhanced location privacy in 5G networks. Sci China Inf Sci, 2020, 63: 220303
41 Russakovsky O, Deng J, Su H, et al. ImageNet large scale visual recognition challenge. Int J Comput Vis, 2015, 115: 211–252
42 Sharafaldin I, Lashkari A H, Ghorbani A A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of International Conference on Information Systems Security and Privacy (ICISSP), Funchal, 2018. 108–116
43 McAfee Labs. McAfee labs threats reports. 2021. https://www.mcafee.com/enterprise/en-us/assets/reports/rp-threats-jun-2021.pdf
44 Singh M, Singh M, Kaur S. Issues and challenges in DNS based botnet detection: a survey. Comput Secur, 2019, 86: 28–52
45 Wang W, Shang Y Y, He Y Z, et al. BotMark: automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors. Inf Sci, 2020, 511: 284–296