

Packet-Level Adversarial Network Traffic Crafting using Sequence Generative Adversarial Networks

Qiumei Cheng, Shiyong Zhou, Yi Shen, Dezhang Kong, Chunming Wu

Abstract—The surge in the internet of things (IoT) devices seriously threatens the current IoT security landscape, which requires a robust network intrusion detection system (NIDS). Despite superior detection accuracy, existing machine learning or deep learning based NIDS are vulnerable to adversarial examples. Recently, generative adversarial networks (GANs) have become a prevailing method in adversarial examples crafting. However, the nature of discrete network traffic at the packet level makes it hard for GAN to craft adversarial traffic as GAN is efficient in generating continuous data like image synthesis. Unlike previous methods that convert discrete network traffic into a grayscale image, this paper gains inspiration from SeqGAN in sequence generation with policy gradient. Based on the structure of SeqGAN, we propose Attack-GAN to generate adversarial network traffic at packet level that complies with domain constraints. Specifically, the adversarial packet generation is formulated into a sequential decision making process. In this case, each byte in a packet is regarded as a token in a sequence. The objective of the generator is to select a token to maximize its expected end reward. To bypass the detection of NIDS, the generated network traffic and benign traffic are classified by a black-box NIDS. The prediction results returned by the NIDS are fed into the discriminator to guide the update of the generator. We generate malicious adversarial traffic based on a real public available dataset with attack functionality unchanged. The experimental results validate that the generated adversarial samples are able to deceive many existing black-box NIDS.

Index Terms—adversarial examples, sequence generative adversarial networks, policy gradient, intrusion detection

I. INTRODUCTION

WITH the surge in the internet of things (IoT) device deployment, network infrastructures have witnessed an unprecedented increase of threats ranging from ransomware to IoT botnets [1]. The current IoT Security landscape always requires resilient and robust network intrusion detection systems (NIDS) to monitor possible anomalies. The recent advancements in machine learning and deep learning have shed light on NIDS and become a prevailing method in identifying network intrusions in the IoT field [2], [3].

Despite superior detection accuracy, NIDS based on machine learning models and state-of-the-art deep neural networks (DNN) lacks robustness against carefully crafted adversarial examples [4]–[6]. Adversarial examples, originally proposed by Szegedy *et al.* [7], try to mislead a trained

model to generate inaccurate outputs by adding imperceptible perturbations to the raw input, which recently gains attention in the security domain. In most cases, an attacker is inclined to launch multiple attempts with minor perturbations of original malicious samples. NIDS responds to each attempt with a signal indicating whether the generated sample is benign or malicious. As a result, this signal guides the attacker to update its generative models. Iteratively, the attacker will generate adversarial samples to successfully deceive NIDS with attack functionality unchanged. In this way, adversarial crafting will trick the NIDS into generating incorrect outputs that the adversary desires. However, generating synthetic but plausible adversarial attack traffic inherits huge challenges and must comply with security domain constraints [8]. The domain constraints can be defined as follows.

- The generated network traffic should meet the sanity checks of network data format.
- Adversarial network traffic crafting is able to bypass the detection of NIDS while retaining attack functionality.

For instance, the packet size of a generated TCP packet shouldn't exceed the maximum value. The port number of a generated packet should be in the range of 0 to 65535. Unlike image synthesis, a minor change of original malicious traffic may disable the attack functionality [9]. Thus, functional features should remain unchanged, e.g., time-based features.

A. Literature Review

Earlier work implements adversarial crafting with a gradient optimizer [10]. The computed gradients are applied to update the inputs. Unlike this approach, Papernot *et al.* [11] construct *saliency maps* that reflecting which part of the input can be perturbed to achieve the adversary's goal, accomplished by the forward derivative based on the Jacobian matrix learned by the DNN. Further, Grosse *et al.* [9] consider one iteration of adversarial re-training in generating synthetic samples. Indeed, adversarial samples can be continuously incorporated in the training process using an adversarial loss function [12]. Recently, the presence of generative models that approximate a data distribution facilitates the adversarial examples crafting [13], such as variational autoencoder (VAE) [14]–[16] and generative adversarial networks (GANs) [17]. GAN estimates generative models from a game-theoretic perspective by utilizing a discriminative network and a generative network. More researchers try to generate adversarial network traffic based on GAN and its variants [18]–[28]. Existing works provide adversarial network traffic crafting at both flow level and packet level.

Q. Cheng, S. Zhou, Y. Shen, and D. Kong are with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310000, China (e-mail: chengqiumei@zju.edu.cn, 21821295@zju.edu.cn, shenyizju@zju.edu.cn, kdz@zju.edu.cn).

C. Wu is with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310000, China. He is also with the Zhejiang Lab, Hangzhou 310000, China (e-mail: wuchunming@zju.edu.cn). Chunming Wu is the corresponding author.

Generally, flow-based network traffic contains several categorical attributes such as IP addresses and port numbers, which are discrete data indeed. The nature of discrete network traffic makes it hard for GAN to craft adversarial network traffic as GAN is efficient in generating continuous data like image synthesis. An alternative approach turns to transform discrete network traffic into continuous values in the data preprocessing. Even though *domain knowledge checks* are employed to evaluate the intrinsic quality of generated traffic, a large amount of literature generates realistic synthetic intrusion data to enrich the labeled data for the imbalanced dataset [18], [19], [21]. These methods are intended to augment data to boost the performance of the original detection model but fail to bypass the detection of NIDS. MalGAN [23] employs the predicted labels of a substitute detector to train the generator dynamically and then crafts malware samples to fool black-box malware detection algorithms. Similar to MalGAN, IDSGAN [24] leverages the prediction results from a black-box IDS in the discriminator. By this approach, the generated sample is capable of deceiving a black-box IDS.

For packet-based network traffic, a packet is composed of multiple bytes represented with hexadecimal in a range of 0-255, which can denote a pixel in a grayscale image. In this context, researchers attempt to craft adversarial network packets with GAN-based models due to its advancement in image synthesis, e.g., PacketCGAN [25], PAC-GAN [26], PcapGAN [27], and Flow-WGAN [20]. Even though the generated packets are conditioned to the network packet format with the type and length checker [20], these models craft adversarial traffic from a perspective of data augmentation for the imbalanced dataset. Unfortunately, they are unable to bypass the detection of NIDS. Zhang *et al.* [28] attempt to generate adversarial examples based on Monte Carlo tree search (MCTS), while their method can only modify original XSS attacks by strictly following XSS bypassing rules. The network packet is a sequence indeed where each byte can be considered as a token. In recent years, sequence generative adversarial networks (SeqGAN) [29] has been proposed to generate a sequence with policy gradient, which sheds light on generating adversarial network data at the packet level.

B. Proposed Approach

In this paper, rather than converting bytes into a grayscale image in previous work, we draw inspiration from SeqGAN in sequence synthesis from a reinforcement learning perspective. Based on the structure of SeqGAN, this paper proposes Attack-GAN for adversarial network traffic crafting at the packet level. Attack-GAN is able to (i) craft raw adversarial network traffic with network packet format, and meanwhile (ii) deceive the detection of NIDS. Attack-GAN envisages the attack traffic generation as a minimax two-player game between a generator and a discriminator. Each byte in a network packet can be regarded as a token in a sequence. Thus, the generation of adversarial network traffic can be formulated into a sequential decision making problem. In this way, the generator is viewed as an agent in reinforcement learning, and the generated bytes is the current state.

The action for the generator is to select a token (byte). In particular, the generator is implemented using long short-term memory (LSTM) due to its strength to learn long-term dependencies in sequence modeling tasks. Prior to adversarial traffic generation, Attack-GAN leverages recent advances in natural language processing and utilizes word embedding to construct the representation of network packets. Besides, Attack-GAN crafts adversarial network traffic conditioned to the security domain constraints to ensure that the generated samples attain the attack functionality.

To bypass the detection of NIDS, a black-box NIDS classifies both the generated adversarial network traffic and benign traffic. The predicted results returned by the NIDS are then fed into a discriminator. By this approach, the discriminator will behave like a NIDS after many epochs of adversarial training. The discriminator, implemented by a binary classifier, outputs a probability indicating whether the generated traffic comes from benign traffic or malicious adversarial traffic. This probability can be used as the reward of a complete generated sequence, which helps to update the parameters of the generator. For incomplete sequences, Attack-GAN adopts the Monte Carlo tree search (MCTS) to sample the remaining tokens. The average value of multiple samples is considered as the expectation of future reward. Thus, Attack-GAN is able to estimate both partially generated packets and complete packets. We conduct experiments with a real public dataset and evaluate the performance of the proposed scheme in terms of several metrics. The experimental results have validated that the proposed scheme can generate adversarial network traffic and evade existing machine learning-based IDS detection. In brief, the contributions of this paper are manifolds:

- We propose Attack-GAN, based on the structure of SeqGAN, to generate adversarial network traffic at packet level by formulating it into a sequential decision making process. The packet is represented with word embedding from the advances of natural language processing.
- We generate adversarial network traffic to deceive the detection of NIDS. The generated network traffic and benign traffic are classified by a black-box NIDS. The prediction results returned by the NIDS are fed into the discriminator to guide the update of the generator.
- The experimental results validate that the adversarial network traffic generated by Attack-GAN is able to deceive many existing black-box NIDS with a rather high attack success increase rate.

The remainder of this paper is organized as follows. Section II presents preliminaries about this paper. Section III outlines the solution overview of Attack-GAN. Section IV elaborates the methodology of our approach. In Section V, we evaluate the performance of Attack-GAN with several metrics. Finally, Section VI concludes this paper and discusses future work.

II. PRELIMINARIES

A. GAN

Generative adversarial networks (GAN) [17] is proposed to estimate generative models from a game-theoretic perspective,

which is widely explored for data generation in many areas [30]–[33]. GAN is composed of two models, a generator G and a discriminator D , implemented by neural networks. The generator G tries to learn the distribution of real data, while the discriminator D aims to differentiate the output of G from real examples as a binary classifier. The two models compete with each other in a zero-sum game and finally reach Nash equilibrium.

The generator takes input noise variables z from the prior $p_z(z)$, and then maps it to real data space as $G(z, \theta_g)$ where θ_g are parameters with a neural network. Similarly, $D(x, \theta_d)$ is defined with parameters θ_d . $D(x)$ denotes the probability that x comes from real data rather than the generator's distribution p_g . The discriminator is intended to maximize the probability of giving the correct label of real samples and generated samples. Meanwhile, the generator is trained to minimize $\log(1 - D(G(z)))$. Thus, this is a typical minimax game and the objective function is:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (1)$$

The objective function is related to Jensen-Shannon (JS) divergence in minimizing the divergence between the generator's distribution p_g and the real data distribution p_{data} . GAN has gained a broader adoption in image synthesis but still inherits challenges in generating discrete sequences. On the one hand, the gradient of the loss from the discriminator has difficulties guiding the generator as the slight loss may refer to the same token. On the other hand, GAN can assess the loss of an entire sequence but fails in evaluating partial sequences.

B. SeqGAN

Yu *et al.* [29] propose sequence generative adversarial networks (SeqGAN) to address the challenges of GAN in crafting adversarial sequences. The rationale of SeqGAN is formulating the sequence generation as a sequential decision making process with the policy gradient. Specifically, SeqGAN considers the generator G_θ as an agent in reinforcement learning. The state of the environment s_t is a sequence of the generated tokens, and the action for the generator is to select a particular token y_t . Given a current state s_t , the generator is intended to maximize its expected accumulated reward:

$$R_t(s_t) = \sum_{y_t \in Y} G_\theta(y_t|s_t) \cdot Q_\phi^\theta(s_t, y_t) \quad (2)$$

where $G_\theta(y_t|s_t)$ denotes the policy that select action y_t at the state s_t , and $Q_\phi^\theta(s_t, y_t)$ represents an action-value function. The action-value function is a total reward expectation following the strategy $G_\theta(y_t|s_t)$ after taking action y_t from the state s_t .

To assess the action-value function, SeqGAN leverages the probability output by the discriminative model. Generally, the discriminator outputs a probability that an input sequence $y_{1:T}$ is real, which is denoted by $D_\phi(y_{1:T})$. In time slot T , SeqGAN defines the action-value function as: $Q_\phi^\theta(s_T, y_T) = D_\phi(y_{1:T})$. It should be noted that the probability output by

the discriminator can only estimate a complete sequence. For incomplete sequences, e.g., $t < T$, the reward cannot be directly obtained by D . To estimate the intermediate state, a natural approach turns to sample the remaining $T - t$ tokens using the Monte Carlo search. Furthermore, SeqGAN applies Monte Carlo search for M times and obtains a collection of M samples:

$$\{Y_{1:T}^1, \dots, Y_{1:T}^m, \dots, Y_{1:T}^M\} = MC(Y_{1:t}; M) \quad (3)$$

where $Y_{1:T}^m$ is composed by $Y_{1:t}^m$ and $Y_{t+1:T}^m$. Then, the average reward of M samples can be considered as the expectation of the future reward. Thus, the reward for a given state s_t can be calculated as:

$$Q_\phi^\theta(s_t, y_t) = \begin{cases} \frac{1}{M} \sum_{m=1}^M D_\phi(y_{1:T}^m) & t < T \\ D_\phi(y_{1:t}) & t = T \end{cases} \quad (4)$$

In this way, SeqGAN is able to estimate both partially generated sequences and complete sequences.

III. SOLUTION OVERVIEW

Generally, a network packet consists of multiple bytes, which can be regarded as a discrete sequence indeed. Although GAN has progressed substantially in generating continuous data, the nature of network packet makes it hard to generate synthetic discrete sequences. Unlike traditional methods that convert network traffic into continuous values, this paper draws inspiration from SeqGAN in sequence synthesis from a reinforcement learning perspective and proposes a packet-level adversarial network traffic generation approach called Attack-GAN. Given a real network attack traffic data set, Attack-GAN aims to generate synthetic but malicious adversarial samples that can bypass IDS detection.

The structure of Attack-GAN is illustrated in Fig. 1. Prior to adversarial traffic generation, Attack-GAN leverages recent advances in natural language processing and utilizes word embedding to construct the representation of network packets. Suppose that a packet from real-world data is denoted as $x_{1:T} = x_1 \oplus \dots \oplus x_t \oplus \dots \oplus x_T$ indexed by time $t \in T$ where x_t represents a byte within the packet, and \oplus is the concatenation operator. The objective of Attack-GAN is to train a generative model G_θ to produce a sequence $y_{1:T} = y_1 \oplus \dots \oplus y_t \oplus \dots \oplus y_T$. In this way, each byte in the packet can be regarded as a token in the sequence. Thus, the generation of adversarial network traffic can be formulated into a sequential decision making problem. In particular, the generator is an agent in reinforcement learning. Specifically, the state s_t in each time step t is composed by current partially generated bytes within a packet $y_{1:t} = y_1 \oplus \dots \oplus y_{t-1}$. The action a_t in time step t is to select a byte y_t . Given a selected action, the state transition is deterministic even the θ -parameterized policy model $G_\theta(y_t|s_t)$ is stochastic. The ability to learn long-term dependencies in sequence modeling has motivated us to choose Long Short Term Memory (LSTM) as the generator. In generating adversarial examples, we keep the bytes corresponding to malicious features unchanged so that the attack functionality of the generated packets can be guaranteed.

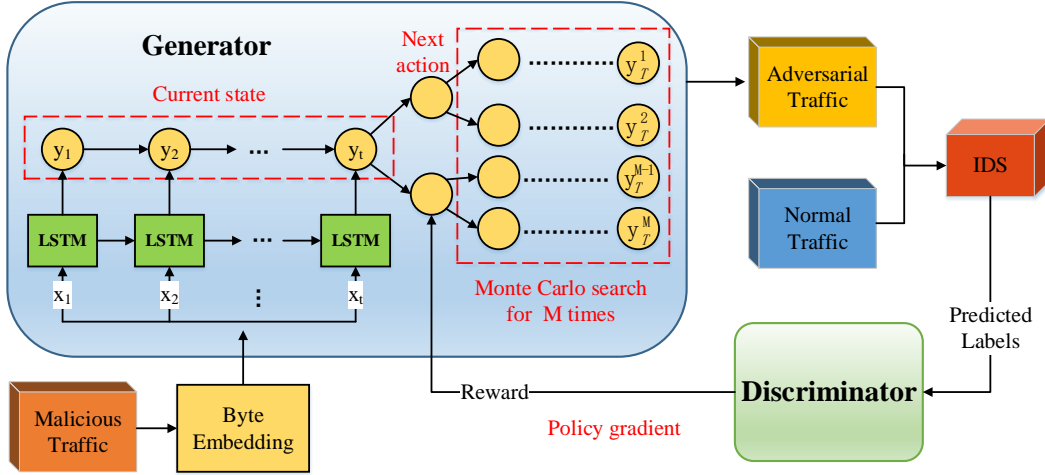


Fig. 1. The structure of Attack-GAN

To bypass the detection of IDS, Attack-GAN inputs both malicious adversarial samples and benign samples into a black box IDS. The predicted results returned by the IDS are then fed into a discriminator. By this approach, the discriminator will behave like an IDS after many epochs of adversarial training. The discriminator, implemented by a simple binary classifier, differentiates whether the sample is adversarial traffic or benign traffic. Given an input packet $y_{1:T}$, the discriminator will output a probability indicating whether $y_{1:T}$ comes from benign traffic or malicious adversarial traffic. This probability can be used as the reward of a complete sequence for the generator. As demonstrated in Fig. 1, for partially generated packets (e.g., $t < T$), Attack-GAN adopts Monte Carlo tree search (MCTS) to sample the remaining tokens. The average value of multiple samples is considered as the expectation of future reward. Therefore, Attack-GAN can assess both partially generated packets and complete packets. The adversarial training between the generator and the discriminator helps generate malicious adversarial samples bypass the black-box IDS.

IV. METHODOLOGY

In this section, we elaborate on the model of Attack-GAN. We first introduce constraints in generating adversarial traffic in the security domain and then detail the process of malicious adversarial traffic generation.

A. Domain Constraints

A typical adversarial attack scenario considers minor perturbations to the input. Ilyas *et al.* [34] define a classical adversarial attack scenario. Given an input x and its target class y , the objective of adversarial attack is to find an input \tilde{x} such that $\|\tilde{x} - x\|_\infty < \epsilon$ and \tilde{x} is classified as y . The distance between original input and adversarial input should be restricted to a perturbation bound ϵ . By doing this, an adversarial sample will be misclassified with minor adjustments. However, the adversarial attack traffic generation at the packet level is more complicated. Unlike image synthesis, a minor

adjustment of original malicious traffic may disable the attack functionality in essence. Thus, the adversarial attack traffic generation must comply with the domain constraints [8].

First, Attack-GAN generates packets conditioned to the basic packet format. For instance, the TTL values of the Windows system are always 32, or 128. Accordingly, the generated token (byte) of TTL can be "20" or "80" in hexadecimal. Second, given an input x to NIDS, e.g., the extracted features of malicious packets, Attack-GAN remains the functional attack features unchanged. These retained attack functional features are extensively studied in many literature [24] [22]. For instance, in flow-based network traffic, attack functional features often cover *time-based* features like *forward inter arrival time*, *backward inter arrival time*, or *host-based* features that have the same destination host or services. This paper also follows such domain constraints to generate an adversarial sample. Since the network traffic discussed in this paper is byte-level, Attack-GAN retains corresponding bytes unchanged in adversarial traffic crafting. More details can be found in Section V.

B. Byte Embedding

Typically, each network packet is composed of multiple bytes. How to represent network traffic at the packet level is significant before generating adversarial traffic. Recently, researchers attempt to construct network traffic representation from the advances in the field of natural language processing (NLP) [20] [35]. In this paper, we utilize recent strengths of word embedding to establish the representation of network traffic. Specifically, each packet is regarded as a sentence where each byte corresponds to a word. We map each byte in the packet into a high-dimensional word vector using word2vec [36]. In comparison with one-hot encoding, word2vec reduces computation complexity in a lower-dimensional vector. More importantly, word2vec considers the similarity in the semantics of different bytes.

In particular, we adopt Skip-Gram to generate word embedding of the bytes. The main principle of Skip-Gram is intended

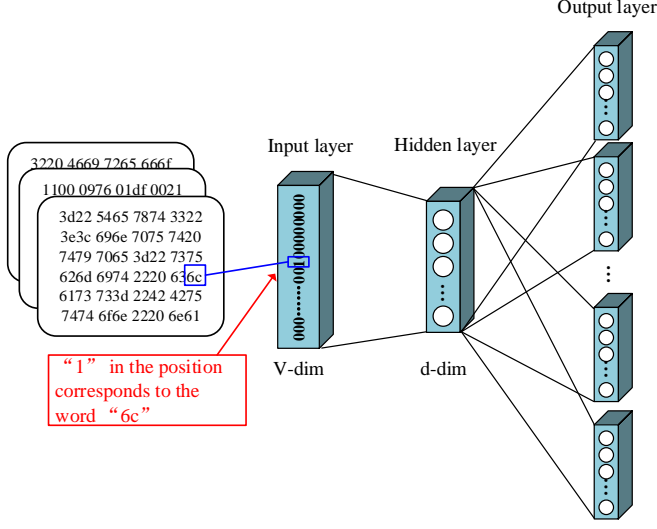


Fig. 2. Byte embedding using Skip-Gram model

to predict the most related words for a given center word. Fig. 2 illustrates the model of byte embedding using Skip-Gram. First, let the size of the vocabulary set be V . The i -th word (byte) is converted into a vector using one-hot encoding $\mathbf{u}^{(i)} \in \mathbb{R}^V$. Then, this word is passed to the hidden layer to perform a dot product to obtain a d -dimensional vector: $\mathbf{v}^{(i)} = \mathbf{W}_h \cdot \mathbf{u}^{(i)}$, where $\mathbf{W}_h \in \mathbb{R}^{V \times d}$ is the weight matrix for the hidden layer. Afterwards, we apply $\mathbf{W}' \in \mathbb{R}^{d \times V}$, a weight matrix between the hidden layer and the output layer, and a softmax function to find the probability of each word. Through byte embedding, each byte in the packet is encoded into a high-dimensional numeric vector.

C. Adversarial Malicious Traffic Crafting

The generative network in the Attack-GAN is implemented by LSTM, a popular variant of Recurrent Neural Network (RNN), due to its superiority to learn long-term dependencies in sequence modeling tasks. LSTM introduces a memory cell c_t by maintaining three gates: the forget gate f_t , the input gate i_t and the output gate o_t . For a given input \mathbf{x}_t , the forget gate f_t decides what information the previous cell state c_{t-1} should be forgotten and the input gate i_t decides the updated values. The hidden state h_t is controlled by the output gate o_t . The formulation of LSTM is described as follows:

$$\begin{aligned}
 f_t &= \sigma(\mathbf{W}_f * \mathbf{x}_t + \mathbf{W}_f * h_{t-1} + b_f) \\
 i_t &= \sigma(\mathbf{W}_i * \mathbf{x}_t + \mathbf{W}_i * h_{t-1} + b_i) \\
 \tilde{c}_t &= \tanh(\mathbf{W}_c * \mathbf{x}_t + \mathbf{W}_c * h_{t-1} + b_c) \\
 c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
 o_t &= \sigma(\mathbf{W}_o * \mathbf{x}_t + \mathbf{W}_o * h_{t-1} + b_o) \\
 h_t &= o_t \odot \tanh(c_t)
 \end{aligned} \tag{5}$$

where \odot denotes the Hadamard product, $\sigma(\cdot)$ is the sigmoid activation function and b_f, b_i, b_c, b_o are bias. The introduction of a memory cell in LSTM keeps the gradient away from vanishing quickly. The hidden states obtained by LSTM cells are then mapped into the token distribution by a softmax output layer. As a result, the generator produces a network packet that consists of T bytes. The generated packet is represented

with a high-dimensional embedding matrix $\mathbf{V} \in \mathbb{R}^{T \times d}$. It is a concatenation of multiple word vectors corresponding to the byte in a packet.

To better update the parameters of the generator, the discriminator is used to predict the probability of whether the generated packet is benign or malicious. In particular, we adopt Convolutional Neural Network (CNN) as the discriminator to guide the update of the generator. The discriminator implemented with CNN is a binary classifier that differentiates between malicious adversarial traffic and benign traffic. Correspondingly, a convolution kernel $\mathbf{w} \in \mathbb{R}^{l \times d}$ is applied to a window size of l words in a sentence to generate a new feature map. If a feature map $\mathbf{c} = \{c_1, c_2, \dots, c_{T-l+1}\}$ is generated, a max-pooling operation will be applied to obtain a maximum value $c_{max} = \max(\mathbf{c})$. Finally, a sigmoid activation function outputs the probability indicating whether the generated packet is from malicious adversarial traffic or benign traffic.

D. Adversarial Training

The outcomes of the discriminator D_ϕ help update the training of the generator G_θ . Ideally, this process can be iteratively to update the parameters of G_θ . Specifically, newly generated adversarial network packets allow to re-train the generator. Thus, the discriminator D_ϕ updates dynamically to better improve the training of the generator G_θ . The discriminator is intended to imitate the IDS. As depicted in Fig. 1, IDS classifies the benign traffic and adversarial traffic. The predicted labels are then fed into the discriminator D_ϕ to update the parameters. Thus, the loss of D_ϕ is determined by the predicted results regarding the benign traffic and the malicious adversarial traffic, which is given by:

$$L_D = - \sum_{x \in \mathcal{X}} \log D_\phi(x) - \sum_{y \in \mathcal{Y}} \log(1 - D_\phi(y)) \tag{6}$$

where \mathcal{X} denotes the set of predicted benign packets, and \mathcal{Y} is the set of predicted malicious packets. Unlike SeqGAN, the discriminator will output a probability $D_\phi(y_{1:T})$ indicating the packet is benign. The intention of G_θ is to generate a malicious adversarial packet that can bypass the IDS. Thus, the loss function of the generator can be defined as:

$$L_G = - \sum_{y_t \in \mathcal{Y}} G_\theta(y_t | s_t) \cdot Q_\phi^\theta(s_t, y_t) \tag{7}$$

where $Q_\phi^\theta(s_t, y_t)$ represents the expected accumulated reward after taking action y_t at the state s_t , and $G_\theta(y_t | s_t)$ is the policy that chooses an action y_t at the state s_t . Thus, the gradient of the loss function about the parameters θ is defined as:

$$\nabla_\theta L_G = - \mathbb{E}_{s_t \sim G_\theta} \left[\sum_{y_t \in \mathcal{Y}} \nabla_\theta G_\theta(y_t | s_t) \cdot Q_\phi^\theta(s_t, y_t) \right] \tag{8}$$

According to the unbiased estimation, the gradient of the loss function can be approximated by:

$$\begin{aligned}
\nabla_{\theta} L_G &\simeq - \sum_{t=1}^T \sum_{y_t \in Y} \nabla_{\theta} G_{\theta}(y_t | s_t) \cdot Q_{\phi}^{\theta}(s_t, y_t) \\
&= - \sum_{t=1}^T \sum_{y_t \in Y} G_{\theta}(y_t | s_t) \cdot \nabla_{\theta} \log G_{\theta}(y_t | s_t) \cdot Q_{\phi}^{\theta}(s_t, y_t) \\
&= - \sum_{t=1}^T \mathbb{E}_{y_t \sim G_{\theta}(y_t | s_t)} \cdot [\nabla_{\theta} \log G_{\theta}(y_t | s_t) \cdot Q_{\phi}^{\theta}(s_t, y_t)]
\end{aligned} \tag{9}$$

More details can be referred to [29]. If we obtain the gradient of the loss function, the parameters are updated with the Adam algorithm. The entire adversarial training process is illustrated in the Algorithm. 1.

Algorithm 1 Adversarial Training of Attack-GAN

Reruire:

```

Generator  $G$  and discriminator  $D$ ; random parameters  $\theta$  and  $\phi$ 
1: Initialize  $G, D$  with random weights  $\theta, \phi$ ;
2: Pretrain a machine learning-based NIDS;
3: Pretrain  $G_{\theta}$  using MLE with real malicious samples;
4: Generate adversarial examples using  $G_{\theta}$ ;
5: Pretrain  $D_{\phi}$  with malicious samples and adversarial samples;
6: while  $D_{\phi}$  is not converged do
7:   for  $g$ -steps do
8:     Generate an adversarial sequence  $y_{1:T} \sim G_{\theta}$ ;
9:     for all  $t \in T$  do
10:      Calculate  $Q_{\phi}^{\theta}(s_t, y_t)$  by Eq. 4;
11:   end for
12:   Obtain the gradient of the generator by Eq. 9;
13:   Update the parameters of the generator;
14: end for
15: for  $d$ -steps do
16:   Combine the adversarial examples and benign traffic to create a dataset;
17:   The IDS classifies the combined dataset and predicts corresponding labels;
18:   for  $k$  epochs do
19:     Train the discriminator  $D_{\phi}$  by Eq. 6;
20:   end for
21: end for
22: end while

```

V. EVALUATION

In this section, we evaluate the performance of Attack-GAN in adversarial attack traffic crafting. First, Attack-GAN employs a real-world attack dataset to generate adversarial traffic. The generated adversarial traffic is then fed into a black-box IDS implemented by current popular machine learning techniques. Finally, the attack performance of generated adversarial traffic is verified.

A. Dataset

The lack of the attack dataset presents a huge challenge in the intrusion detection field. In the past two decades, the KDD99 dataset is a benchmark dataset explored in previous studies. However, this dataset is recently criticized as outdated. In this paper, we turn to the CTU-13 dataset¹ captured in the CTU University, a collection that contains malware botnet traffic, benign traffic, and background traffic. The labeled dataset provides raw botnet captures in a separate pcap file. More importantly, this dataset provides valuable information indicating the characteristics of each botnet scenario, which helps to determine the malicious features in the botnet traffic. Particularly, we select 40000 benign packets and 18532 malicious botnet packets.

In order to evaluate the robustness of existing NIDS against adversarial examples that add small perturbations to the input, we pre-train four machine learning-based NIDS that are widely used in previous work [24], including multilayer perceptron (MLP), support vector machine (SVM), decision tree (DT), and logistic regression (LR). These machine learning-based models are then considered as black-box NIDS in Attack-GAN. Table I demonstrates the training set and the test set of these NIDS models.

TABLE I
DATASET IN PRE-TRAINED NIDS

		Malicious	benign	Total
IDS	Train	13899	30000	43899
	Test	4633	10000	14633
Total		18532	40000	58532

Finally, these models achieve a high detection accuracy with over 99%, as well as high precision and recall. These machine learning-based NIDS models are considered black-box NIDS capable of differentiating malicious packets and benign packets. However, the details and inner parameters are invisible to the attackers.

B. Preprocessing

Before adversarial crafting, the raw packets captured in the pcap file require normalization as they share different packet lengths. Appropriate packet length provides an excellent way to describe a network activity. Theoretically, the maximum TCP packet size should not exceed the maximum transmission unit (MTU) of a physical network. The typical value of the MTU is set to 1500 bytes. If the maximum TCP packet size is larger than the MTU, e.g., 2000 bytes, it will result in performance degradation since long packets will be separated in transmission. Considering that the captured traffic is short packets, we rescale the packet size to the same length with 300 bytes to provide efficient transmission and imitate real network traffic. If the packet size exceeds 300 bytes, the remaining part is truncated accordingly. For shorter packets, such as 200 bytes, the rest of the bytes are padding with zero. In this way, the length of a generated sequence is set to 200.

¹<https://www.stratosphereips.org/datasets-ctu13/>

Additionally, we pre-train word embeddings using word2vec before generating adversarial traffic. Pre-trained word embeddings are able to capture the semantic relationships of words when they are trained on large datasets. In most cases, they help enhance the performance of a model, which is widely used in the natural language processing field. Specifically, we adopt word2vec to obtain pre-trained word embeddings with 18532 malicious packets and 40000 benign packets. As a result, each word (token) obtains a numerical representation in 32 dimensions.

Moreover, the attribute in a packet can also be represented with two bytes or even four bytes. In order to evaluate the impact of byte embedding on performance, this paper also defines *Two-bytes* embedding as a comparison. Therefore, we have *One-byte* embedding and *Two-bytes* embedding in Attack-GAN, which denotes that a word is represented with one byte and two bytes, respectively. In *One-byte* embedding, we attain a vocabulary set of 256. For *Two-bytes* embedding, the vocabulary set explodes to 65536. The vocabulary set of four bytes is considerably large, which is not discussed in this paper.

C. Performance Metrics

In this paper, we define the following metric to measure the performance of the generated adversarial traffic.

- **Mean Absolute Percentage Error** In most cases, adversarial examples are generated by adding subtle perturbations in the input features. In this regard, it is essential to reserve the features of malicious traffic when generating adversarial examples. Even though we retain the bytes corresponding to the functional attack features unchanged, it still requires quantifying whether the generated adversarial traffic looks like original malicious traffic. A well-recognized method turns to estimate the similarity between adversarial samples and original real samples. Mean absolute percentage error (MAPE) is an important measure to quantify the generated adversarial traffic [22]. Similar to [22], we define a modified version of the mean absolute percentage error to quantify the deviation between generated traffic and malicious traffic. Given an original malicious packet $x_{1:T} = x_1 \oplus \dots \oplus x_t \oplus \dots \oplus x_T$ and a generated adversarial packet $y_{1:T} = y_1 \oplus \dots \oplus y_t \oplus \dots \oplus y_T$, the MAPE value can be computed by :

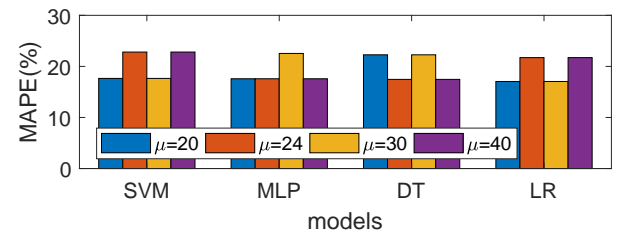
$$\text{MAPE}(x_{1:T}, y_{1:T}) = \frac{100\%}{T} \sum_{t=1}^T \left| \frac{\|\hat{x}_t\|_2 - \|\hat{y}_t\|_2}{\|\hat{x}_t\|_2} \right| \quad (10)$$

where T is the total number of tokens and x_t, y_t is an individual byte within the original malicious sample and generated adversarial sample, respectively. Since each byte is represented with a numerical vector in 32 dimensions, we first normalize the data in the 0-1 range using Min-Max Normalization, denoted by \hat{x}_t and \hat{y}_t . Afterwards, we employ the l^2 -norm of \hat{x}_t and \hat{y}_t to calculate the deviation. Generally, lower MAPE values reflect a higher similarity between generated traffic and real malicious traffic.

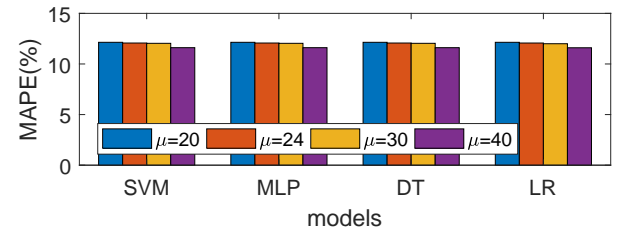
- **Attack Failure Rate** Above all, the attack performance of the generated adversarial traffic is a focus of the evaluation. If the generated malicious adversarial packet is predicted benign by a black-box NIDS, we consider it a successful attack. In contrast, a malicious label predicted by the NIDS is deemed as an attack failure. Attack failure rate (AFR) denotes the ratio of the malicious samples predicted by the black-box NIDS to all the attack attempts. A higher attack failure rate reflects the robustness of a black-box IDS against adversarial examples.
- **Attack Success Increase Rate** As mentioned above, the attack success rate (ASR) denotes the ratios of the malicious sample predicted benign by the NIDS to all the attack samples. Further, attack success increase rate (ASIR) is the difference between the ASR using Attack-GAN and the original ASR, illustrating the improvement of attack success with adversarial traffic generated by Attack-GAN. A higher ASIR reflects that more adversarial samples have deceived the NIDS.

D. Experimental Results

The CTU-13 dataset captures real botnet traffic in a pcap file and provides rich information about characteristics in botnet scenario. This information facilitates to discover of malicious features in raw packets. We finally keep 20 bytes corresponding to malicious features (e.g., the source and destination port, header length, etc.) unchanged in adversarial crafting. If we use these malicious bytes to replace the corresponding bytes in benign traffic, the modified benign traffic will be predicted malicious with a high detection accuracy, ranging from 89% to 98% in four NIDS models. Thus, these features are notable enough to differentiate malicious packets and benign packets.



(a) MAPE value using *One-byte* embedding



(b) MAPE value using *Two-bytes* embedding

Fig. 3. MAPE value under different machine learning-based NIDS models: (a) MAPE value between the adversarial traffic and raw malicious traffic using *One-byte* embedding; (b) MAPE value between the adversarial traffic and raw malicious traffic using *Two-bytes* embedding.

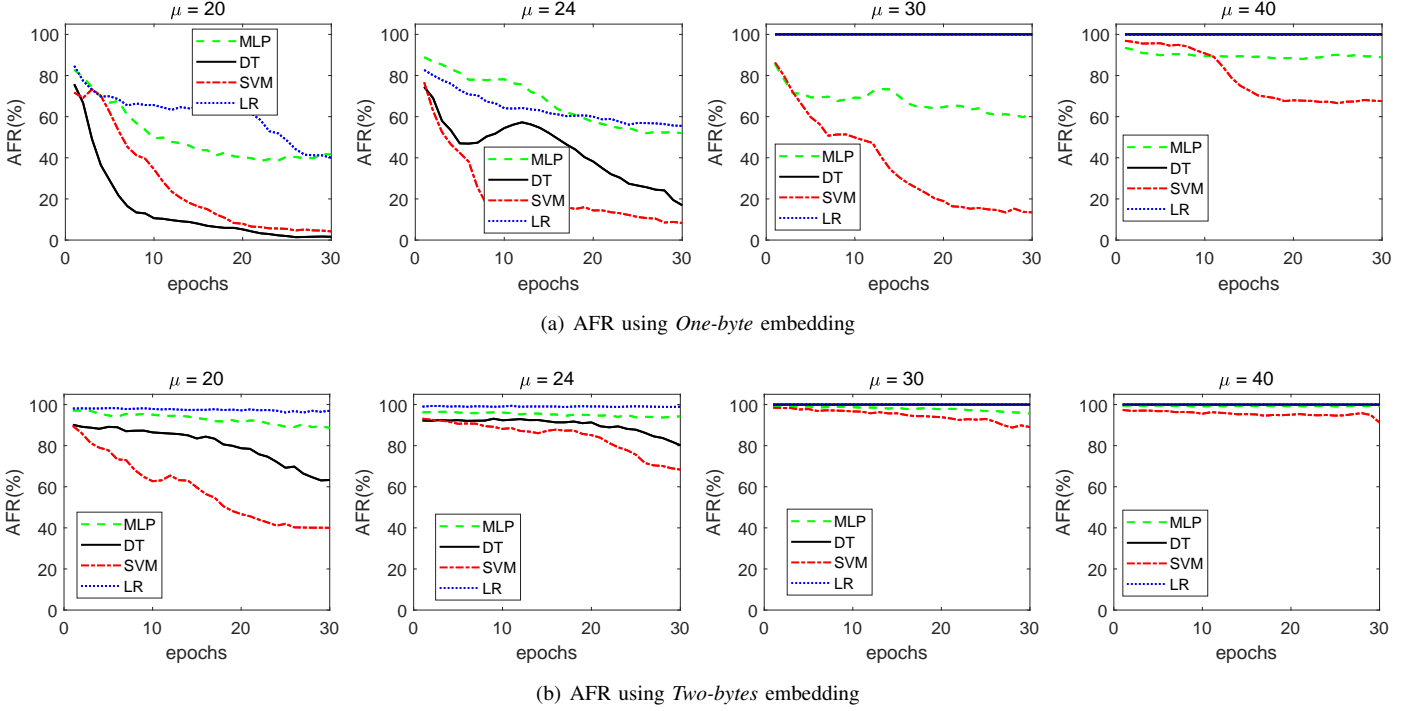


Fig. 4. AFR using different byte embedding: (a) AFR under different black-box NIDS using *One-byte* embedding; (b) AFR under different black-box NIDS using *Two-bytes* embedding.

To estimate the impact of the number of malicious bytes on attack performance, we also introduce a variable μ to control the malicious bytes that remain unchanged in adversarial crafting. Specifically, we set μ as 20, 24, 30, 40 to estimate the attack performance of generated traffic using Attack-GAN. We conduct adversarial training with 30 epochs and measure the MAPE value, AFR, as well as ASIR, respectively.

1) *MAPE*: This subsection first measures the similarity between the generated adversarial traffic and real malicious traffic. We calculate the mean absolute percentage error in the entire adversarial training process. Fig. 3 demonstrates the MAPE value in 30 epochs of adversarial training.

In Fig. 3(a), the generated adversarial samples using *One-byte* embedding for all IDS models yield a MAPE value between 17% to 22.81%. In general, there is no distinct variation in the MAPE value as the fixed malicious features increases. Even though the fixed malicious features vary from 20 to 40, the MAPE value under different IDS models still fluctuates around 20%. Zhang *et al.* [22] has claimed that a MAPE value round 20% is capable of staying the adversarial samples deviating from raw malicious traffic.

If the generated adversarial traffic are using *Two-bytes* embedding, the quantified MAPE value is smaller than that of *One-byte* embedding, as illustrated in Fig. 3(b). Overall, this value fluctuates around 12%, which indicates that the adversarial samples are very similar to the real malicious samples. To be more specific, the MAPE value obtained by different IDS models are very close to each other, even with different fixed malicious features. This is due to word embedding using *Two-bytes* embedding after normalization is very close. Generally, generated adversarial traffic using *Two-*

bytes embedding is closer to real malicious traffic.

2) *AFR*: Fig. 4 illustrates the attack failure rate using different byte embedding modes under four NIDS models over 30 epochs of adversarial training. Fig. 4(a) shows the AFR using *One-byte* embedding. Overall, the AFR under four models with $\mu = 20$ all decreases as the epoch increases. In particular, SVM and DT present the lower AFR, both decreasing sharply at the beginning of adversarial training and finally decrease to about 2% after 30 epochs. In comparison, the AFR of LR and MLP start with a slight decrease over epochs and then approached around 40%. If the number of fixed malicious bytes rises to 24, DT exhibits a wavelike decline to 16%. This fluctuation is mainly caused by the instability when training GANs [37] [38]. Even though the AFR of SVM descends dramatically to 8.2%, LR falls slowly to 60%. In general, the AFR values of the four models all show a downturn with fewer fixed malicious bytes.

In contrast, the AFR demonstrates significantly different results if we keep more malicious features unchanged. For $\mu = 30$, both DT and LR remain constant AFR at approximately 99%. MLP manifests a slight fluctuating downward trend of AFR through the whole adversarial training process and finally falls to 61%. Unlike other models, SVM achieves better performance. Its AFR first drops significantly in the previous 20 epochs and then stays static at around 13%. If we still increase fixed malicious features to 40, the value of AFR will be much higher. DT and LR performed similarly with that of $\mu = 30$ with a fairly higher AFR. MLP shows a very slight decline from 93.4% to 88.8%. SVM descends mildly at the previous 10 epochs and then rapidly falls to 67.5%. Generally, AFR exhibits a downward trend in four

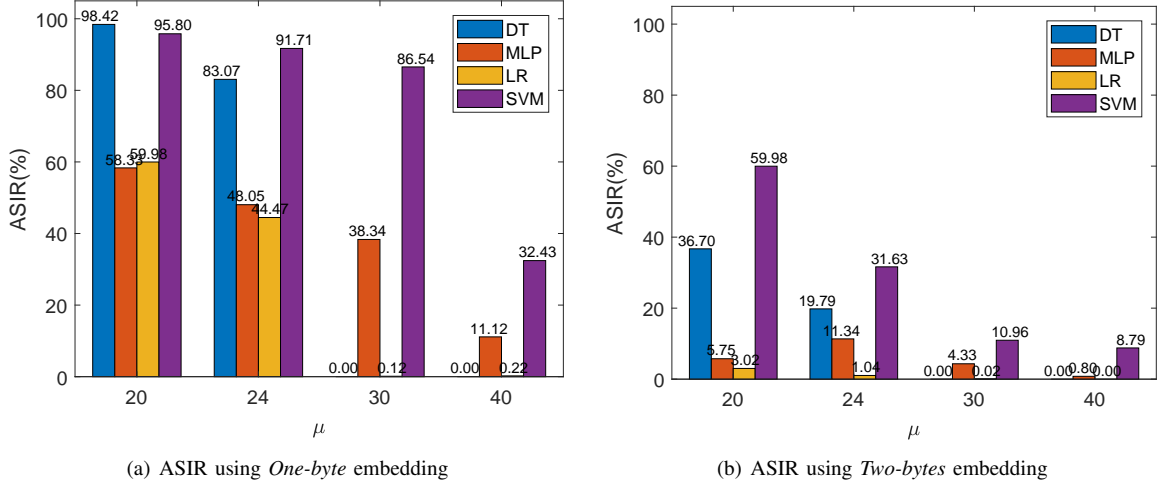


Fig. 5. ASIR under different machine learning-based NIDS models: (a) ASIR using *One-byte* embedding; (b) ASIR using *Two-bytes* embedding

models with few fixed malicious features. If we keep more malicious features unchanged in adversarial traffic crafting, most machine learning-based NIDS models attain a higher or constant AFR. This implies that adversarial attacks are harder to succeed with more malicious features unchanged. Luckily, SVM still decreases to a relatively lower AFR even with 30 or 40 fixed malicious features.

Fig. 4(b) demonstrates attack failure rate using *two-bytes* embedding under different NIDS models over 30 epochs of adversarial training. Similarly, we control 20, 24, 30, 40 bytes corresponding to malicious features unchanged as that of *One-byte* embedding. Therefore, we stay 10, 12, 15, 20 words unchanged accordingly in the word embedding. For $\mu = 20$, we can easily observe that SVM shows a substantial decline in AFR. In comparison, DT goes down relatively slowly from 90.2% to 63.3%. However, MLP and LR all decrease only two percentage points throughout the entire adversarial training process. If μ increases to 24, AFR decreases slowly in both SVM and DT. With higher malicious features, all these models attain a higher AFR. Luckily, the AFR of SVM still declines even with 30 or 40 fixed malicious features.

In general, both *One-byte* and *Two-bytes* embedding have displayed a downward trend over the adversarial training epochs. Nevertheless, with the fixed malicious features increases, the AFR for most models is falling more slowly. In other words, adversarial attacks are harder to succeed with more malicious features unchanged. On the whole, AFR with *Two-bytes* embedding drops slowly compared with that of *One-byte* embedding.

3) *ASIR*: According to the attack failure rate mentioned above, it is easy to calculate the attack success rate after adversarial training. Further, we can estimate the attack success increase rate towards the original attack success rate. Fig. 5(a) shows ASIR using *One-byte* embedding under four NIDS models. With few fixed malicious features, e.g., 20 or 24, there are greater improvements in attack success rate for all models, especially for DT and SVM accounted for 98.42% and 95.80%, respectively. If fixed malicious features are higher with 30, even 40, only SVM and MLP exhibit

an improvement of attack success rate comparing to original models. SVM still keeps a considerable higher ASIR even with more malicious features unchanged. Overall, ASIR has progressed substantially using Attack-GAN with few fixed malicious features, which indicates that adversarial traffic generated by Attack-GAN can successfully bypass existing most machine learning-based IDS models.

If Attack-GAN crafts adversarial traffic using *Two-bytes* embedding, the attack success increase rate is much lower than that of *One-byte* embedding in general. Fig. 5(b) depicts ASIR using *Two-bytes* embedding. There is a downward trend of ASIR in the graph as the fixed malicious features increase, which is consistent with *One-byte* embedding. In contrast, the attack success rate for LR does not exhibit a distinct increase when μ is 30 or 40. Among these four models, SVM obtains the highest attack success increase rate with 59.98%. But this value still below that of *One-byte* embedding. The huge difference in attack success increase rate lies in their embedding methods. In *Two-bytes* embedding, the premise of the same length packet requires fewer words in a sentence during the training of word embedding. As a result, it captures less semantic information using *Two-bytes* embedding. Furthermore, it affects the prediction accuracy.

The results of AFR and ASIR observed in the experiments reveal that the malicious adversarial traffic generated by Attack-GAN can deceive current machine learning-based NIDS models. Notably, the attack success rate is related to the number of fixed malicious features. In most cases, the generated adversarial traffic is harder to bypass current NIDS if we keep more malicious features unchanged. Additionally, adversarial traffic using *One-byte* embedding obtains a higher attack success rate. Compared with the original attack success rate, there is a considerable improvement in adversarial traffic using Attack-GAN. Besides, the length of a packet (i.e., the length of a sentence in word embedding) is another essential factor in adversarial traffic crafting, which can be discussed in our future work.

VI. CONCLUSION

The current network intrusion detection systems (NIDS) in the internet of things (IoT) have shown vulnerabilities in adversarial examples. This paper proposes Attack-GAN based on the structure of sequence generative adversarial networks (SeqGAN) to generate malicious adversarial raw packets. The generation of a packet is considered a sequential decision making problem. Each byte in a packet is represented with word embedding. To evade NIDS detection, the results returned by a black-box NIDS are utilized in the discriminator to help update the parameters of the generator. The experimental results reveal that Attack-GAN can generate malicious adversarial samples that evade the detection of NIDS. Its attack failure rate is relevant to the modes of byte embedding, as well as the fixed malicious bytes. However, Attack-GAN can only generate adversarial samples with a fixed length. In our future work, we will attempt to craft adversarial samples with variable length.

REFERENCES

- [1] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on internet-scale IoT exploitations," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2702–2733, Apr. 2019.
- [2] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for IoT security based on learning techniques," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2671–2701, Jan. 2019.
- [3] N. Moustafa, B. Turnbull, and K.-K. R. Choo, "An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4815–4830, Jun. 2019.
- [4] M. Usama, M. Asim, S. Latif, J. Qadir, and A.-A. Fuqaha, "Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems," in *Proc. Int. Wireless Commun. Mobile Comput. conf. (IWCMC)*, Tangier, Morocco, Jun. 2019, pp. 78–83.
- [5] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE S&P*, San Jose, CA, USA, Jun. 2017, pp. 39–57.
- [6] M. J. Hashemi, G. Cusack, and E. Keller, "Towards evaluation of NIDSs in adversarial setting," in *Proc. ACM CoNEXT Workshop big data, mach. learn. artif. intell. data commun. netw.*, Orlando, FL, USA, Dec. 2019, pp. 14–21.
- [7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.
- [8] R. Sheatsley, N. Papernot, M. Weisman, G. Verma, and P. McDaniel, "Adversarial examples in constrained domains," *arXiv preprint arXiv:2011.01183*, 2020.
- [9] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial perturbations against deep neural networks for malware classification," *arXiv preprint arXiv:1606.04435*, 2016.
- [10] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," *arXiv preprint arXiv:1412.1897*, 2014.
- [11] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Security Privacy (EuroS&P)*, Saarbrücken, Germany, Mar. 2016, pp. 372 – 387.
- [12] I. J. Goodfellow, J. Shlens, and S. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, San Diego, USA, May. 2015, pp. 1–11.
- [13] J. Kos, I. Fischer, and D. Song, "Adversarial examples for generative models," in *Proc. IEEE S&P Workshops*, San Francisco, USA, May. 2018, pp. 36–42.
- [14] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
- [15] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Variational data generative model for intrusion detection," *Knowl. Inf. Syst.*, vol. 60, pp. 569–590, Jul. 2019.
- [16] A. B. L. Larsen, S. K. S?nderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," *arXiv preprint arXiv:1512.09300*, 2015.
- [17] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. NIPS*, Montreal, Canada, Dec. 2014, pp. 2672–2680.
- [18] C. Yin, Y. Zhu, S. Liu, J. Fei, and H. Zhang, "An enhancing framework for botnet detection using generative adversarial networks," in *Proc. Int. Conf. Artif. Intell. Big Data*, Chengdu, China, Jun. 2018, pp. 228–234.
- [19] M. H. Shahriar, N. I. Haque, M. A. Rahman, and M. A. Jr, "G-IDS: Generative adversarial networks assisted intrusion detection system," *arXiv preprint arXiv:2006.00676*, 2013.
- [20] L. Han, Y. Sheng, and X. Zeng, "A packet-length-adjustable attention model based on bytes embedding using Flow-WGAN for smart cyber-security," *IEEE Access*, vol. 7, pp. 82 913–82 926, Jun. 2019.
- [21] M. Ring, D. Schlor, D. Landes, and A. Hotho, "Flow-based network traffic generation using generative adversarial networks," *Computers & Security*, vol. 82, pp. 156–172, May. 2019.
- [22] C. Zhang, X. Costa-Prez, and P. Patras, "Tiki-Taka: Attacking and defending deep learning-based intrusion detection systems," in *Proc. ACM SIGSAC Conf. Cloud Comput. Security Workshop*, USA, Nov. 2020, pp. 27–39.
- [23] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on GAN," *arXiv preprint arXiv:1702.05983*, 2017.
- [24] Z. Lin, Y. Shi, and Z. Xue, "IDSGAN: generative adversarial networks for attack generation against intrusion detection," *arXiv preprint arXiv:1809.02077*, 2018.
- [25] P. Wang, S. Li, F. Ye, Z. Wang, and M. Zhang, "PacketCGAN: Exploratory study of class imbalance for encrypted traffic classification using cgan," *arXiv preprint arXiv:1911.12046*, 2019.
- [26] A. Cheng, "PAC-GAN: Packet generation of network traffic using generative adversarial networks," in *Proc. IEEE 10th Annu. Inf. Technol. Electron. Mobile Commun. Conf. (IEMCON)*, Vancouver, Canada, Oct. 2019, pp. 728–734.
- [27] D. Baik, Y. Jung, and C. Choi, "PcapGAN: Packet capture file generator by style-based generative adversarial networks," in *Proc. IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Boca Raton, Florida, USA, Dec. 2019, pp. 1149–1154.
- [28] X. Zhang, Y. Zhou, S. Pei, J. Zhuge, and J. Chen, "Adversarial examples detection for XSS attacks based on generative adversarial networks," *IEEE Access*, vol. 8, pp. 10 989–10 996, Jan. 2020.
- [29] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence generative adversarial nets with policy gradient," in *Proc. 31th AAAI Conf. Artif. Intell.*, San Francisco, USA, Feb. 2017, pp. 2852–2858.
- [30] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," in *Proc. NIPS*, Long Beach, CA, USA, Dec. 2017, pp. 5768 – 5778.
- [31] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.
- [32] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," *arXiv preprint arXiv:1802.04208*, 2018.
- [33] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," *arXiv preprint arXiv:1609.02612*, 2016.
- [34] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," in *Proc. ICML*, Stockholm, Sweden, Jul. 2018, pp. 3392–3401.
- [35] E. Min, J. Long, Q. Liu, J. Cui, and W. Chen, "TR-IDS: Anomaly-based intrusion detection through text-convolutional neural network and random forest," *Security Commun. Netw.*, pp. 1–9, Jul. 2018.
- [36] M. Tomas, C. Kai, C. Greg, and D. Jeffrey, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.
- [37] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *arXiv preprint arXiv:1701.07875*, 2017.
- [38] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Proc. NIPS*, Barcelona, Spain, Dec. 2016, pp. 2234 – 2242.