Digital Communications and Networks 7 (2021) 453–460

Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.keaipublishing.com/dcan

Fooling intrusion detection systems using adversarially autoencoder

Junjun Chen [a], Di Wu [b,c], Ying Zhao [a,*], Nabin Sharma [b], Michael Blumenstein [c], Shui Yu [b]

[a] College of Information Science and Technology, Beijing University of Chemical Technology, Beijing, 100029, China
[b] School of Computer Science, University of Technology Sydney, Ultimo, 2007, Australia
[c] Centre for Artificial Intelligence, University of Technology Sydney, Ultimo, 2007, Australia

ARTICLE INFO

Keywords:
Intrusion detection
Cyber attacks
Autoencoder
Generative adversarial networks

ABSTRACT

Due to the increasing cyber-attacks, various Intrusion Detection Systems (IDSs) have been proposed to identify network anomalies. Most existing machine learning-based IDSs learn patterns from the features extracted from network traffic flows, and the deep learning-based approaches can learn data distribution features from the raw data to differentiate normal and anomalous network flows. Although having been used in the real world widely, the above methods are vulnerable to some types of attacks. In this paper, we propose a novel attack framework, Anti-Intrusion Detection AutoEncoder (AIDAE), to generate features to disable the IDS. In the proposed framework, an encoder transforms features into a latent space, and multiple decoders reconstruct the continuous and discrete features, respectively. Additionally, a generative adversarial network is used to learn the flexible prior distribution of the latent space. The correlation between continuous and discrete features can be kept by using the proposed training scheme. Experiments conducted on NSL-KDD, UNSW-NB15, and CICIDS2017 datasets show that the generated features indeed degrade the detection performance of existing IDSs dramatically.

1. Introduction

Cyber attacks are ever-present threats around the world. Some of the typical attack methods, such as the denial of service, unauthorized access, and malicious code, cause tremendous damage to governments, enterprises, and organizations [1]. Consequently, various Intrusion Detection Systems (IDSs) and network traffic classification systems based on Machine Learning (ML) or Deep Learning (DL) techniques have been proposed to detect anomaly attacks and analyze network traffic [2,3]. However, both ML and DL techniques are feature-based methods, where the ML techniques learn the patterns from handcraft features, and the DL techniques learn data distribution features from the raw data to classify the traffic flows. Therefore, if the attackers can mimic the benign network flow features, they can disable the classifier and bypass the IDS to initiate attacks.

Traditionally, the IDS consists of three main components: the preprocessor, the detector, and the response module [4]. The preprocessor captures the raw network traffic and transforms it into the data, which can be handled by the detector. Then the detector consists of one or more predefined classification models for differentiating the anomaly from normal network events. If an intrusion is detected, the response unit is triggered. Currently, researchers mainly focus on how to improve the detection performance of the detector.

Based on the detection mechanism, the IDS can be categorized into three main types: misused detection, anomaly detection, and hybrid detection [5]. The misuse detection uses previous knowledge about anomaly patterns to identify network intrusions, which can achieve good detection performance with low false alarm rates for known vulnerabilities. However, this type of approach can not be used to detect zero-day attacks whose patterns are unknown to the detector. For example, if one IDS does not have or update the associated knowledge about a novel attack, it can not identify this attack. The anomaly detection identifies anomalies by comparing the network traffic with a predefined normality model. In the detecting process, the network traffic that does not fit the normality model is considered as an anomaly by the IDS. The hybrid detection mechanism integrates the misuse and anomaly detection methods in the anomaly detection procedure.

Although having been successfully deployed in commercial and industrial environments, IDSs are still affected by threats from attackers. Recent research results demonstrate that the adversarial attacks limit the effectiveness of the ML-based or DL-based detectors in real scenarios [6]. For example, the attackers created elaborately manipulated samples of Android malware to induce the detector to produce outputs they expected [7]. Moreover, the Generative Adversarial Network (GAN) was

* Corresponding author.
E-mail address: zhaoy@mail.buct.edu.cn (Y. Zhao).

https://doi.org/10.1016/j.dcan.2020.11.001
Received 11 September 2019; Received in revised form 2 November 2020; Accepted 17 November 2020
Available online 25 November 2020

used to generate features to disable the IDS [8]. A series of solutions have been proposed to mitigate the adversarial attacks against ML/DL techniques. To defend attacks in the training phase, data sanitization [9] was used to identify and remove the poisoned data from the training dataset. To defend attacks in the testing phase, feature selection [10], adversarial training [11], and robust optimization methods [12] were proposed by researchers.

In the network security field, the conflict between the attackers and the defenders leads to an escalating arms race, where both attacks and defenses continually evolve to achieve their goals and overcome their opponents [13,14]. The ML-based and DL-based detectors learn the normal or anomalous data features to identify malicious network traffic such that the attackers can generate network traffic flows with specific patterns to disable the IDS.

In this paper, we propose a feature generative framework against the IDS. Different from existing adversarial attacks that carefully craft adversarial perturbations to samples, the proposed method learns the distribution of the normal features and can generate features that follow the distribution of the normal features to bypass the IDS. Our purpose is not to promote cyber attacks and crimes, but rather to explore the limits of the IDS and improve the robustness of the detector. Our contributions can be summarized as follows:

- We propose a novel feature generative model, namely, the Anti-Intrusion Detection AutoEncoder (AIDAE), to learn the distribution of normal features and randomly generate features that can be used by attackers to generate real network traffic flows to bypass the existing IDSs.
- The multi-channel decoders are separated into continuous and discrete channels to generate continuous and discrete features, respectively. Moreover, the generated features can keep the correlation between continuous and discrete parts via the same well-trained encoder. Thus the generated features can follow the original distribution of normal features.
- We evaluate the AIDAE on three representative network anomaly detection datasets (NSL-KDD, UNSW-NB15, and CIC-IDS-2017) with the ML and DL baseline IDSs. Experimental results show that the features generated by the proposed framework indeed disable the baseline IDSs.

The rest of this paper is organized as follows: we introduce the related work in Section 2. The attack model is given in Section 3. In Section 4, we describe the proposed AIDAE framework. The experimental evaluation is presented in Section 5, and the potential defense mechanisms are discussed in Section 6. In Section 7, we give the conclusion. To improve readability, the main acronyms used in this paper are listed in Table 1.

2. Related work

Cybersecurity has attracted widespread attention from research communities, and numerous anomaly detection methods based on ML/DL have been proposed [15]. The ML/DL techniques are strong and effective learning frameworks for complex classification tasks, and these techniques have advanced radically in the past years. However, they are vulnerable to the adversarial attacks, where the adversaries initiate attacks to compromise the detector [16].

The adversaries can modify the input instances to evade detection from the IDS. The evasion attacks can be divided into two categories: 1) problem space attacks that generate malicious instances for the specific detection system, and 2) feature space attacks where the attackers manipulate the features used by ML/DL to bypass the detection [17]. For the problem space attacks, Biggio et al. [18] used a gradient-based approach to systematically assess the security of several classification algorithms against the evasion attacks. For the feature space attacks, different approaches have been proposed. Yu et al. [19] used a semi-Markov model to simulate four user browsing behavior parameters to initiate attacks. However, the simulated parameters are simple, and the second-order statistical metrics can detect this attack.

In computer vision research, the GAN and AutoEncoder (AE) have shown their powerful ability in generating high-quality fake images or videos. Inspired by the success of the GAN and AE, some researchers used the generative model to disable the IDS. A self-adapting malware communication framework [20] was proposed, where a GAN was deployed to generate parameters. The malware received the generated parameters and used them to mimic the normal Facebook chat network traffic. The generative model can only generate three continuous features and does not divide the features into discrete features and continuous features, so it is hard to generate real network traffic with the generated features. Another method, namely, MalGAN [21], used the GAN to generate binary malware features to disable the black-box ML malware detection system, but these generated binary features are hard to represent complex network traffic features. Additionally, Lin et al. [8] designed the IDSGAN to generate features to deceive and evade the IDS. Because this method needs the outputs of the IDS to calculate the loss in each training epoch, the IDS can easily identify this adversarial training pattern and block it. Different from other evasion attack methods, the proposed AIDAE learns the distribution of the normal features to generate features randomly and does not need the feedback of the IDS in the training procedure. Moreover, the AIDAE takes into account the correlation between continuous features and discrete features in the feature generation process. Therefore, the AIDEA can be used by attackers in real scenarios.

3. Attack model

A network intrusion is an unauthorized penetration of the target network, where the attackers transmit malicious information or misuse network resources. The IDS is a protector of the target network, which plays an adversarial game with the intruders in the network world. To facilitate understanding of the intrusion attack, it is necessary to disclose the goal, knowledge, and capability of the attacker. Fig. 1 presents the evasion attack scenario, where the network traffic of attacker 1 is identified as anomaly traffic and is blocked by the IDS, and attacker 2 bypasses the IDS through the network traffic camouflage technique.

Table 1
Acronyms used in the manuscript.

Acronym	Definition
AIDAE	Anti-intrusion Detection Autoencoder
AE	Autoencoder
GAN	Generative Adversarial Network
ARAE	Adversarially Regularized Autoencoder
LR	Logistic Regression
k-NN	k-Nearest Neighbor
DT	Decision Tree
RF	Random Forest
ML/DL	Machine/Deep Learning

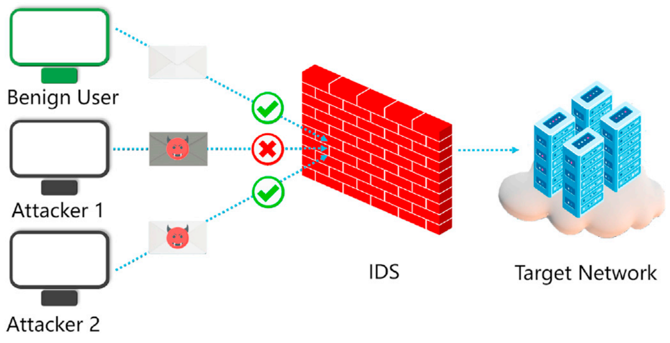


Fig. 1. Attack scenario.

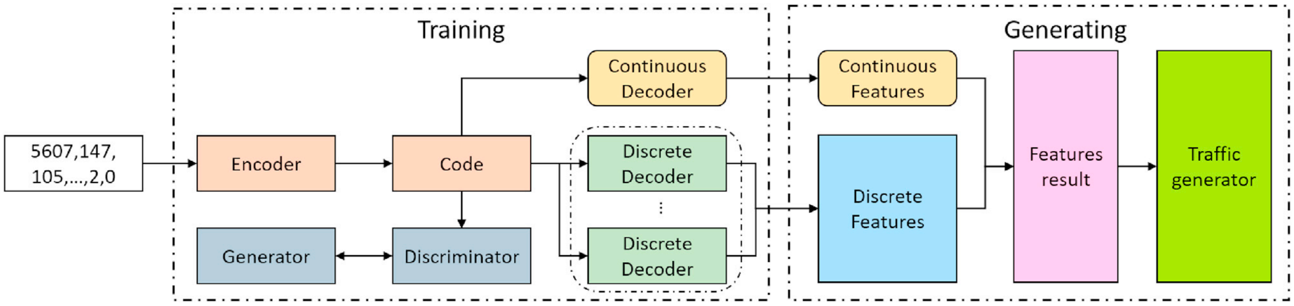


Fig. 2. The framework of the AIDAE (anti-intrusion detection autoencoder).

Attacker's goal. In this attack setting, the attacker's goal is to keep communication with the target network so that the IDS can not detect the network traffic with malicious content. To achieve this goal, the attacker needs to generate traffic following the distribution of the normal features, which can disable the IDS.

Attacker's knowledge. The attacker's knowledge about the target IDS is vital for launching an evasion attack. According to Kerckhoffs's principle, the attacker knows the details about the IDS [22]. Such knowledge may include the training data, detection algorithm, sample features, detection procedure, and others. In this scenario, the attackers can manipulate their network packets to bypass the specific detection algorithms. However, the IDS struggles to protect itself from attacks in the real world, so the attacker can not know everything about the IDS. In this paper, we assume that the attacker knows the features extracted from the network traffic by the IDS.

Attacker's capability. The attacker's capability is limited to spoof or disguise the network traffic in the network intrusion scenario because the attacker has no permission to access and modify the IDS. As the encrypted protocols (TLS) are widely used in network transmission, the traffic classifiers based on Deep Packet Inspection (DPI) make it hard to identify the traffic by the packet payload [23]. Therefore, the attacker can hide the malicious information in the encrypted traffic, and make the flow-level features and statistical-level features of the generated traffic obey the normal distribution, thereby initiating attacks.

4. Proposed framework

4.1. Overview of AIDAE

In this section, we present the framework and training procedure of the proposed method. To process discrete data, the Adversarially Regularized AutoEncoders (ARAE) [24] was proposed to learn more robust discrete-space representations. Based on ARAE, we design the AIDAE with the multi-channel decoders where each discrete decoder represents one discrete feature, and the continuous decoder represents all continuous features. According to Ref. [24], the model performance is strongly dependent on the choice of prior distributions of latent space, and the Gaussian distribution $N(0;1)$ used as the prior distribution may lead to mode collapse in practice. Therefore, we also use a GAN to learn the flexible prior distribution of latent space in the proposed method. The AIDAE learns the distribution of the traffic flow features and then uses latent space codes to reconstruct the features that follow the distribution of the normal features by multi-channel decoders.

The proposed framework is composed of two major parts, the AE and the GAN. The encoder of the AE transforms the traffic features into a latent space code, and the decoders reconstruct the features from the code. The discriminator of the GAN discriminates whether the code is real or fake, and the generator generates a fake code via a random vector input and tries to disable the discriminator. The structure of the AIDAE is presented in Fig. 2. After training, the generated fake codes are sent to decoders to reconstruct the features. Because the fake code is generated from a random vector z, and the fake code c' is random, the features reconstructed by decoders are random.

The network structures of the AIDAE are shown in Table 2, where *input_size* is the dimension of the features, *z_size* is the dimension of the code c, d_i represents the dimension of the *i-th* one-hot encoded discrete feature, *con_size* represents the dimension of continuous features, and *noise_size* is the random vector size of the generator.

4.2. Continuous and discrete features

Features extracted from traffic flows can be categorized into continuous and discrete features. Fig. 3 shows the features of the NSL-KDD dataset.

Such numbers as "507", "437", and "14,421" in black represent the features of "duration", "sources bytes", and "destination bytes", respectively. The values of these features are continuous. Thus, we consider these kinds of features as continuous features. Moreover, numbers in red such as "1", "12", and "10" represent discrete features. The values of these types are within certain ranges. For example, "1" represents the feature "protocol type", whose range is from integer 1 to integer 3 to indicate different protocols.

4.3. Training algorithm

As shown in Algorithm.1, there are two steps for training the proposed method. First, we train the encoder, generator, discriminator, and continuous feature decoder. The input of the encoder is all the features

Table 2
The network structures of AIDAE.

Encoder	Continuous decoder	Discrete decoder	Generator	Discriminator
Linear(*input_size*, 256)	Linear(*z_size*, 128)	Linear(*z_size*, 256)	Linear(*noise_size*, 256)	Linear(*z_size*, 256)
LeakyReLU()	LeakyReLU()	LeakyReLU()	ReLU()	LeakyReLU()
Linear(256, 128)	Linear(128, 256))	Linear(256, 512)	Linear(256, 128)	Linear(256, 512)
LeakyReLU()	LeakyReLU()	LeakyReLU()	ReLU()	LeakyReLU()
Linear(128, 64)	Linear(256, 512)	Linear(512, d_i)	Linear(128, *z_size*)	Linear(512, 256)
LeakyReLU()	LeakyReLU()	GumbelSoftmax()		LeakyReLU()
Linear(64, *z_size*)	Linear(512, 128)			Linear(128, 1)
	LeakyReLU()			Sigmoid()
	Linear(128, *con_size*)			
	ReLU()			

(continuous and discrete features). Thus, the raw continuous and discrete features can be represented by a latent space code via the encoder. Then, we train the discrete feature decoders.

Algorithm 1. AIDAE Training

```
Input: Normal traffic features set $\{f_1, f_2, ..., f_m\}$

(1) Train the encoder, continuous features
  decoder, and GAN
1. The encoder embeds $f$ to the code $c$, and the
  continuous features decoder reconstruct
  continuous features by this $c$;
2. Update the encoder and continuous features
  decoder parameters $(\phi, \theta^{\diamond})$ by minimizing $\mathcal{L}^{\diamond}$;
3. The generator $\mathcal{G}$ generates the fake code $c'$,
  and the discriminator $\mathcal{D}$ discriminates $c'$ from
  the real code $c$;
4. Update the parameters of the $\mathcal{G}$ and $\mathcal{D}$;

(2) Train the discrete feature decoders
1. The trained encoder embeds $f$ to $c$, and the
  discrete feature decoders reconstruct the
  discrete features by $c$;
2. Update the discrete decoders parameters $(\theta^*)$
  by minimizing $\mathcal{L}^*$;
```

The loss function of the AIDAE is described as follows: consider F as the set of input features, $F = \{f_1, f_2, \dots, f_m\}$ where m is the number of instances. $f_k^{\diamond}$ and $f_k^* = \{f_{k,1}^*, f_{k,2}^*, \dots, f_{k,n}^*\}$ represent continuous features and discrete features of the k-*th* instance, respectively, where n is the number of discrete features. E and D are the encoder and the decoder, and parameters φ and $\theta^{\diamond}$ indicate the parameters of the encoder and the continuous decoder, respectively. $\mathbb{P}_{data}^{\diamond}$ represents the distribution of continuous features. For the continuous features, the Mean Square Error (MSE) loss $\mathscr{L}^{\diamond}$ can be represented as Eq. (1).

$$\mathscr{L}^{\diamond}(\varphi, \theta^{\diamond}) = \mathbb{E}_{f^{\diamond} \sim \mathbb{P}_{data}^{\diamond}} \left|\left| f^{\diamond} - D_{\theta^{\diamond}}(E_{\varphi}(f)) \right|\right|^2 \tag{1}$$

For each discrete decoder, the result can be represented as $D_{\theta_i^*}(E_{\varphi}(f))$, where θ_i^* is the parameters of the *i-th* discrete decoder ($i \in \{1, n\}$). Thus, all results from each discrete decoder will be concatenated based on their original positions through function *M*, which is as shown as follows:

$$D_{\theta^*} = M\left(D_{\theta_1^*}(E_{\varphi}(f)), D_{\theta_2^*}(E_{\varphi}(f)), \dots, D_{\theta_n^*}(E_{\varphi}(f))\right)$$

Each discrete feature $f_{k,t}^*$ can be represented with a one-hot encoded vector x_t. The cross-entropy loss of the discrete features $\mathscr{L}^*$ can be represented as Eq. (2) to minimize the reconstruction error:

$$\mathscr{L}^*(\varphi, \theta^*) = \sum_{t=1}^{n} \sum_{j=1}^{d_t} -x_t^j log \widehat{x}_t^j \tag{2}$$

where d_t is the dimension of x_t. x_t^j and $\widehat{x}_t^j$ are the real and generated values of *j-th* dimension, respectively.

A GAN is used to learn the flexible prior distribution of the latent space in the AIDAE. c is the code from the encoder, and z is a random vector. The proposed GAN training scheme can ensure the generated code c' follows the code distribution from the encoder. The min-max optimization for the GAN can be written as follows:

$$\min_{\mathscr{G}} \max_{\mathscr{D}} \mathbb{E}_{\mathbf{c} \sim \mathbb{P}(\mathbf{c})}[log \mathscr{D}(c)] + \mathbb{E}_{\mathbf{z} \sim \mathbb{P}(\mathbf{z})}[\log(1 - \mathscr{D}(\mathscr{G}(z)))]$$

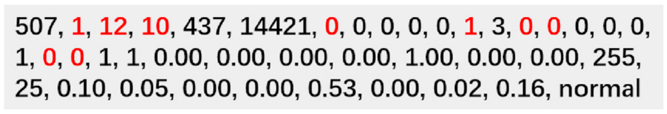

507, 1, 12, 10, 437, 14421, 0, 0, 0, 0, 0, 1, 3, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 255, 25, 0.10, 0.05, 0.00, 0.00, 0.53, 0.00, 0.02, 0.16, normal

Fig. 3. Features of one traffic flow in the NSL-KDD dataset.

4.4. Feature concatenating

In the feature generation process, the GAN generator generates and sends a fake code to both continuous and discrete decoders. The decoders reconstruct continuous features and discrete features by the fake code, respectively. After that, the generated features will be concatenated according to their positions, as shown in Fig. 4. The combination of continuous and discrete features follows the distribution of the original training features. Therefore, the correlation between the continuous and discrete features will be kept.

Compared with other methods, the proposed framework can mimic the complicated features and ensure that the continuous features and discrete features keep the correlation. Thus, the generated features can be used to generate real network traffic flows.

5. Evaluation

5.1. Datasets

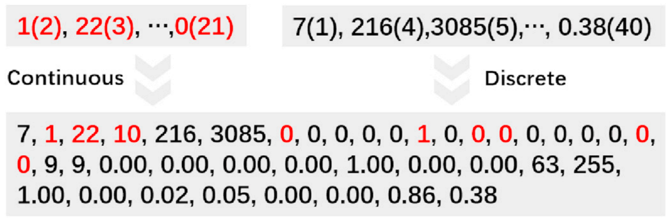
To evaluate the AIDAE, we conducted experiments on three typical intrusion detection datasets, namely, NSL-KDD, UNSW-NB15, and CICIDS2017. The details of the features of these datasets are shown in Table 3.

NSL-KDD [25] is refined from the KDD99 dataset, and it is still a benchmark dataset for testing different intrusion detection methods. Since the feature "num_outbound_cmds" only has a unique value, we removed this feature in the experiments. Therefore, there are 1 label, 33 continuous features, and 7 discrete features in the NSL-KDD dataset.

UNSW-NB15 [26] is a well-known network intrusion detection dataset, which is released by the Australian Centre for Cyber Security. UNSW-NB15 provides the training dataset and testing dataset in the CSV format. There are 1 label, 37 continuous features, and 5 discrete features in the dataset.

CICIDS2017 [27] is another state-of-art dataset proposed by Canadian Institute for Cybersecurity in 2017. This dataset consists of benign network events and six up-to-date common attacks, which are produced by realistic background traffic. We removed the flag features, such as "Fwd PSH Flags" and "Fwd URG Flags". For the discrete feature(destination port), we selected common application port numbers as its range of values, such as port 21, port 53, port 80, and so on. We selected 1 label, 59 continuous features, and 1 discrete feature from the CICIDS2017 dataset.

For all datasets, discrete features are one-hot encoded, and each continuous feature is logarithmically transformed and then is scaled by Min-Max normalization (Eq. (3)) to eliminate the impact of different


1(2), 22(3), ⋯,0(21)

7(1), 216(4),3085(5),⋯, 0.38(40)

7, 1, 22, 10, 216, 3085, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 63, 255, 1.00, 0.00, 0.02, 0.05, 0.00, 0.00, 0.86, 0.38

Fig. 4. The features concatenating process.

values range between features.

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3}$$

where x is a feature value, x_{min} is the minimum logarithm value of this particular feature, x_{max} is the maximum logarithm value, and x' is the value after normalization.

For each dataset, 100,000 records were randomly selected to create new datasets, which are composed of a training set D_{train} and a testing set D_{test}, to evaluate the proposed model. D_{train} includes 30,000 normal records and 30,000 anomaly records, and D_{test} includes 20,000 normal records and 20,000 anomaly records. All experiments were conducted by using Python and PyTorch on an RHEL7.5 server with Intel Xeon W-2133 3.6 GHz CPU and Nvidia Quadro P5000 GPU.

5.2. Effectiveness of AIDAE

In this evaluation, we focused on whether the generated features can disable the IDS. For evaluating the evasion ability of the generated features, the Detection Rate (DR) and the Evasion Increase Rate (EIR) [8] were measured, showing the effectiveness of the proposed features generative model directly. The DR means the proportion of the correctly detected anomaly features by the IDS to all detected anomaly features. The EIR reflects the evasion ability by comparing the adversarial DR and the original DR, which is formulated as Eq. (4).

$$EIR = 1 - \frac{Adversarial\ DR}{Original\ DR} \tag{4}$$

In the training phase, D_{train} was used to train the intrusion detection classifiers, and the proposed AIDAE was trained only by the normal records in D_{train}. In the testing phase, the 20,000 anomaly records in D_{test} were fed to the intrusion detection classifiers to obtain the original DR. Then, we evaluated the DR of 20,000 generated records that were labeled as anomalies. We repeated the experiments five times, and randomly reselected records from the original dataset to create the D for each evaluation.

Table 4 shows the DRs on different datasets. Both the DRs of ML/DL baseline IDS methods decrease dramatically. For the NSL-KDD dataset, The DR of CNN + LSTM decreases from 98.71± 0.86% to 1.44± 0.39%. Moreover, LR has the minimum decrease, which is from around 87.51 ±3.73% to around 5.53 ± 1.02%. For the UNSW-NB15 dataset, CNN + LSTM has the maximum decrease from around 98.83±0.61% to around 1.51±0.46%, and k-NN has the minimum decrease from around 97.14±0.70% to around 7.11±1.38%. For the CICIDS2017 dataset, the maximum decrease happens in CNN + LSTM which decreases from 99.15±0.06% to 0.94±0.11%, and the minimum decrease appears in LR, which is from 93.47±2.53% to 3.02±1.28%. The results show that the features generated by the AIDAE successfully disable the baseline IDS.

Fig. 5 shows the evasion increase rates of the baseline algorithms on different datasets. According to Eq. (4), a higher evasion rate indicates that more adversarial examples can evade the IDS. The experimental results show that all EIRs of different baseline detection methods on three datasets are higher than 0.9, which indicates that the proposed method can generate features to evade the IDS.

5.3. Performance of AIDAE

The performance of generating features is another important aspect of evaluating the proposed AIDAE. Therefore, we evaluated whether the model can efficiently generate diverse features.

For the feature generation model, the greater diversity of features means the larger range of the feature value. Because the values of the discrete features are from a fixed set, we only evaluate the diversity of continuous features. To obtain diverse feature values, we used ReLU as the activation function of the last layer in the continuous feature decoder. The diversity rate $_F$ can be formulated as Eq. (5).

$$F = \frac{f'_{max} - f'_{min}}{f_{max} - f_{min}} \tag{5}$$

where f'_{max} is the maximum value of the generated feature f', and f'_{min} is the minimum value of f'. f_{max} and f_{min} are the maximum value and the minimum value of the original feature f, respectively. We used $_F$ to evaluate the proposed model's ability to generate continuous features. If $_F$ is greater than 1, it means that the value range of f' is greater than the value range of f, and vice versa.

Fig. 6 provides the diversity rates of the continuous features generated in the NSL-KDD dataset. Except for feature 13, all diversity rates are higher than 0.95. The uneven value distribution is the reason for the anomalous diversity rate. For feature 13, the number of samples whose original values are higher than f'_{max} is only 0.65% of the total sample number, and the generated value can cover 99.35% training samples'

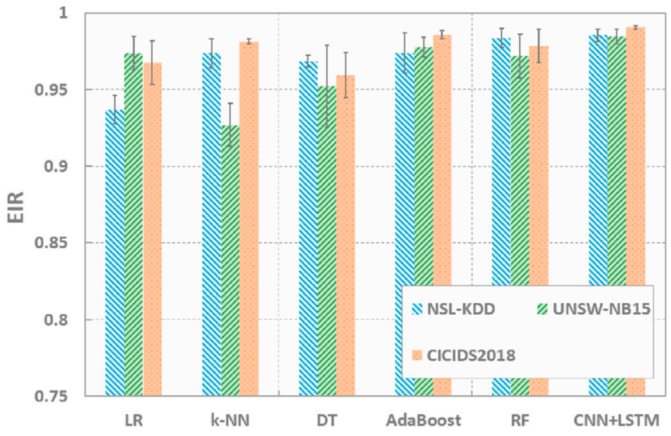


Fig. 5. Evasion increase rates of baseline algorithms.

Table 3
Feature number.

Dataset	Continuous Features	Discrete Features
NSL-KDD	33	7
UNSW-NB15	37	5
CICIDS2017	59	1

Table 4
Detection Rate(%) on different datasets.

Methods	NSL-KDD		UNSW-NB15		CICIDS2017	
	Original DR	Adversarial DR	Original DR	Adversarial DR	Original DR	Adversarial DR
LR	87.51±3.73	**5.53±1.02**	96.52±1.42	**2.54±1.07**	93.47±2.53	**3.02±1.28**
k-NN	91.64±3.16	**2.42±0.82**	97.14±0.70	**7.11±1.38**	98.25±0.36	**1.83±0.17**
DT	95.88±2.14	**3.01±0.33**	98.50±0.31	**4.72±2.64**	97.18±0.85	**3.94±1.45**
AdaBoost	93.53±1.42	**2.45±1.24**	95.21±1.25	**2.13±0.63**	96.57±1.33	**1.39±0.24**
RF	95.15±0.75	**1.56±0.61**	98.78±0.04	**2.77±1.41**	98.72±0.34	**2.14±1.07**
CNN + LSTM	98.71±0.86	**1.44±0.39**	98.83±0.61	**1.51±0.46**	99.15±0.06	**0.94±0.11**

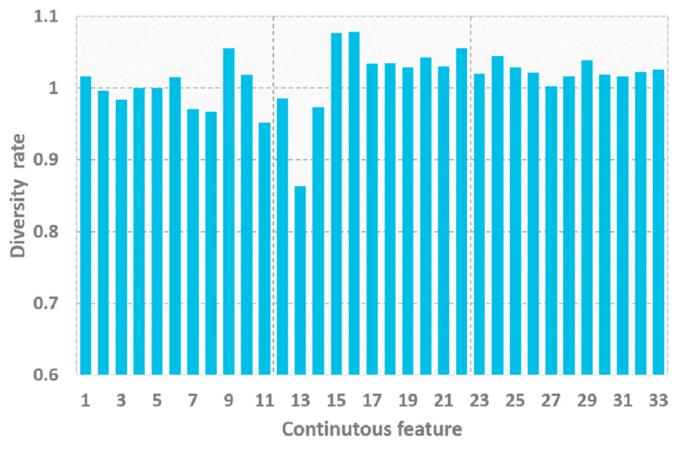


Fig. 6. Diversity rates of generated continuous features of NSL-KDD.

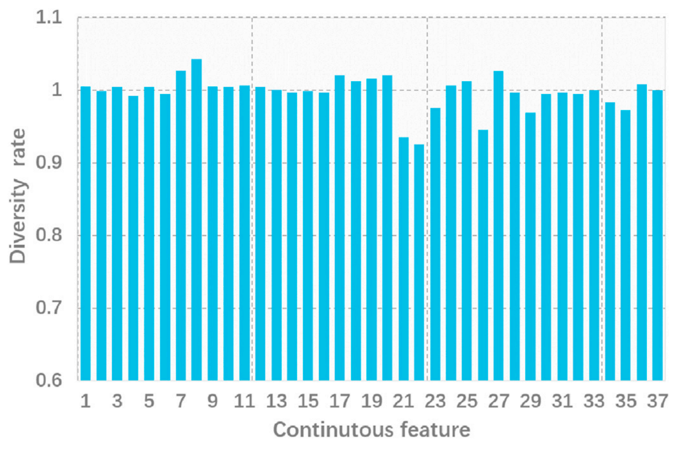


Fig. 7. Diversity rates of generated continuous features of UNSW-NB15.

values. Moreover, some features' diversity rates are higher than 1, which means that the AIDAE generates new values for some features. According to Table 4, the generated features can disable the IDS so that the attackers can use these new values. Therefore, the proposed generative framework can learn well the manifolds of continuous features. Figs. 7 and 8 show the diversity rates of the generated continuous features on the UNSW-NB15 dataset and the CICIDS2017 dataset, respectively. The diversity rates of the two experimental results are all higher than 92%, which means the generated features have a wide range of values, and the proposed method can be used to generate features.

Since the discrete feature value is from a fixed value set, we measured the distribution similarity between the generated discrete features and the original discrete features to evaluate the performance of the AIDAE. The Jensen-Shannon Divergence (JSD) [28] was used in this evaluation. For each dataset, we randomly sampled 20,000 normal records from D_{train} as $T_{standard}$, and sampled another 20,000 records from the generated records as T_{gen}. We also sampled 20,000 normal records from D_{test} as T_{real}. We computed the Average Jensen-Shannon Divergence (AJSD) between the discrete features from $T_{standard}$ and the T_{gen}, and compared it with the AJSD between $T_{standard}$ and T_{real}. The AJSD is shown as Eq. (6).

$$AJSD = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{2}\sum_{j=1}^{d_i}p_i\left(j\right)log\frac{p_i(j)}{M_i(j)} + \frac{1}{2}\sum_{j=1}^{d_i}q_i\left(j\right)log\frac{q_i(j)}{M_i(j)}\right) \tag{6}$$

where n is the number of discrete features, and d_i is the dimension of the i-th feature. $p_i(j)$ and $q_i(j)$ represent the possibility of the j-th dimension from the i-th discrete feature in different distributions, and $M_i(j)$ is equal to $(p_i(j) + q_i(j))/2$.

Fig. 9 shows that the AJSD between $T_{standard}$ and T_{gen} is similar to the AJSD between $T_{standard}$ and T_{real}, which means that the proposed method can learn well the distribution of discrete features.

As shown in Table. 5, the time cost is another evaluation metric. The 400 epochs training time of NSL-KDD, UNSW-NB15, and CICIDS2017 datasets are 876.64 s, 639.79 s, and 714.28 s, respectively. The time of generating 100,000 records by the trained models are 3.13 s, 2.76 s, and 2.92 s, respectively. The results suggest that the AIDAE has low computational complexity.

5.4. Generating network traffic

To disable the IDS, the attackers can generate traffic flows with malicious payloads from the generated features. Inspired by the patents in Refs. [29,30], we introduce the network traffic generation procedure in this section. We categorized the generated features as flow-level features and statistic-level features. The flow-level features include packet size, packet time features, protocol, etc. The statistic-level features include the number of connections that contain the same service and the source

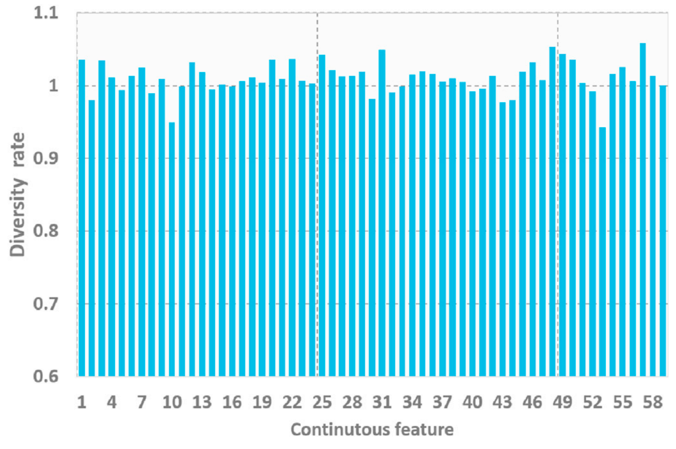


Fig. 8. Diversity rates of generated continuous features of CICIDS2017.

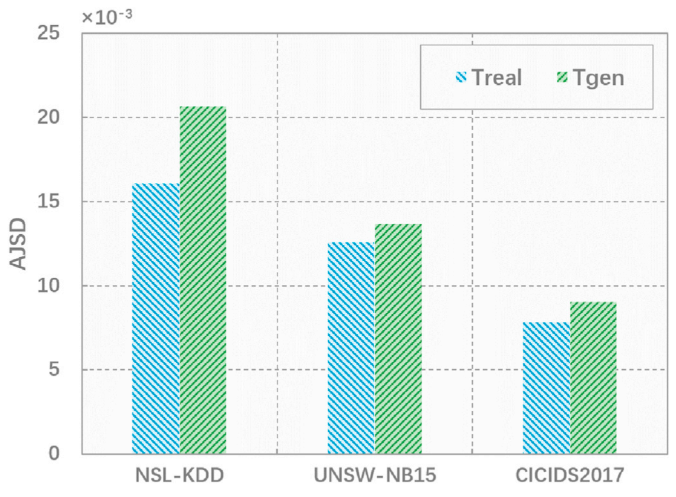


Fig. 9. Average Jensen-Shannon Divergence.

Table 5
Time cost(seconds) of both datasets.

Dataset	Training of AIDAE	Generating features
NSL-KDD	876.64	3.13
UNSW-NB15	639.79	2.76
CICIDS2017	714.28	2.92

address in 100 connections, etc. Note that the generated value of the integer feature needs to be rounded in the generation procedure. The generation procedure is as follows:

1. Determining the protocol and status of each layer;
2. Generating the payload according to the distribution of payload size;
3. Constructing the header of each layer protocol;
4. Determining the transmission time of packets and doing retransmission operation according to the corresponding features;
5. Modifying the flows to fit the other flow-level features;
6. Modifying the flows to fit the statistic-level features in the generation procedure.

Although NSL-KDD, UNSWNB-15, and CICIDS2017 are bidirectional flows, we consider both unidirectional flow and bidirectional flow scenarios. The unidirectional flow can be used in the scenario where the attackers send the traffic flows with malicious payloads to the target, and the bidirectional flow can be used in the scenario where the attackers communicate with malicious clients. The cost time and the number of generated packets are presented in Fig. 10 and Fig. 11, respectively.

It can be seen from the results that the time consumption and packet number scale linearly as the flow number grows. For the unidirectional flow, the time of generating 2,000 unidirectional flows (25,683 packets) by using the generated features based on the NSL-KDD dataset is 41.76 s, the time of generating 2,000 unidirectional flows (44,713 packets) by using the generated features based on the UNSW-NB15 dataset is 77.61 s, and the time of generating 2,000 unidirectional flows (26,579 packets) by using the generated features based on the CICIDS2017 dataset is 52.17 s. Additionally, the time of generating 2,000 bidirectional flows (35,640 packets) with the generated NSL-KDD features is 71.89 s, the time of generating 2,000 bidirectional flows (95,963 packets) with the generated UNSW-NB15 features is 204.96 s, and the time of generating 2,000 bidirectional flows (61,278 packets) with the generated CICIDS2017 features is 159.53 s. We only calculated the time of constructing packets and sending the packets throughout the network interfaces. The packet transportation time and the waiting time for fitting the time features distribution were not calculated.

6. Discussion on defense mechanisms

Defenses against evasion attacks that identify and block nomalies to reduce the effects of malicious network communications are challenging tasks in cybersecurity. In this section, we focus on the potential defenses against the proposed method and discuss how our attack evades them.

6.1. Feature selection

Feature selection is one of the core steps in ML/DL methods, which selects a subset of relevant features to improve the classification performance or reduce the computational complexity [31]. However, if the algorithm designer does not consider the evasion attack, the feature selection may degrade the detection performance of the model because the attackers need to manipulate fewer features to initiate attacks. Zhang et al. [10] proposed an adversarial feature selection method for evading attacks to tackle the above issues, which used an optimization criterion to maximize the classifier's generalization capability and security to evasion. Although outperforming traditional approaches in classifier security, the feature selection method is not suitable for the proposed AIDAE. The AIDAE can generate features that conform to the normal distribution, and can maintain the correlation between different features, so the generated feature subsets also follow the distribution of normal features. Thus, the generated features can evade the feature selection--based defenses.

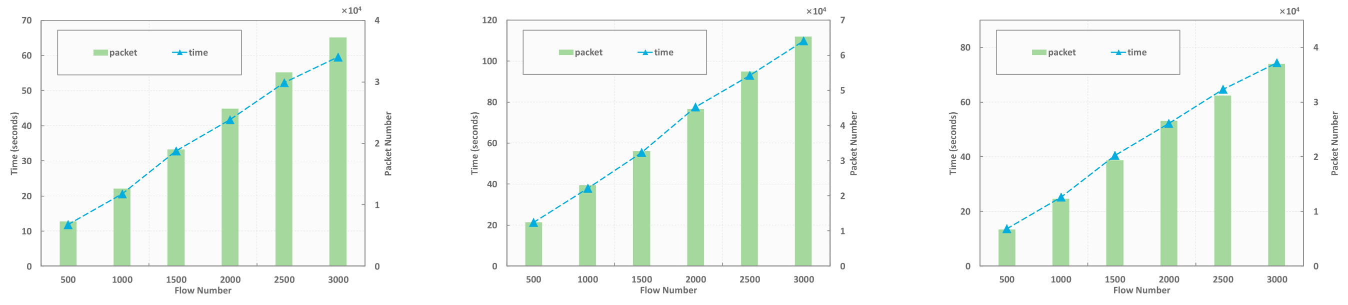
6.2. Adversarial attack detection

Adversarial attacks seriously compromise the security of ML/DL applications. In the past few years, researchers gave different explanations for the adversarial examples. According to Ref. [32], the adversarial examples are hard to find because adversarial examples represent low-probability "pockets" in the examples manifold. Moreover, Goodfellow et al. [33] demonstrated that the neural networks' linear nature is the primary cause of their vulnerability to adversarial examples.

To detect the adversarial attacks, substantial approaches have been devised in recent literature. Kantchelian et al. [11] used a prediction-based algorithm to create adversarial instances and added them to the training data to harden the detection model for evasion attacks. The robust optimization [12] is another solution, which smooths the decision boundaries of the ML algorithm to limit the influence of adversarial samples. The adversarial samples can be identified because they are only similar to, but do not follow, the distribution of the normal samples. Unlike the adversarial examples, we trained a generator to learn normal features manifolds and generate features for the attackers to generate network traffic with malicious payloads. Therefore, the adversarial attack detection can not identify the proposed attacks.

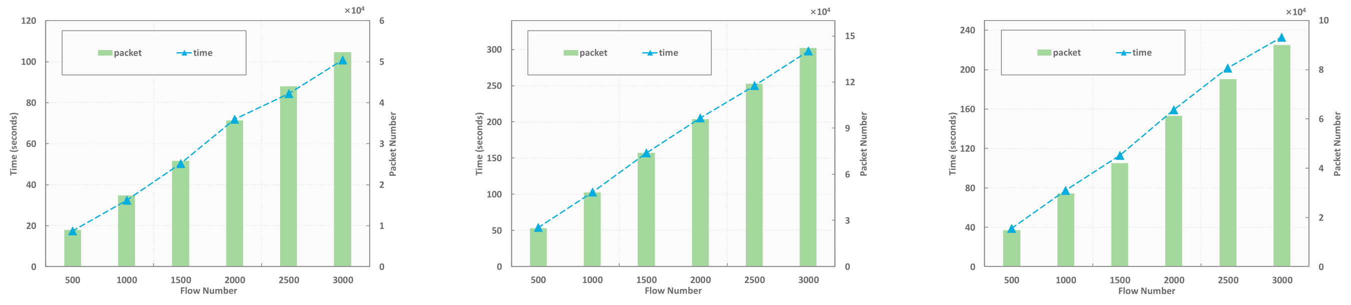
6.3. Payload-based detection

Payloads are raw data encapsulated in network frames, such as the contents of the removed IP header and TCP/UDP header in the datagram structure. Analyzing payloads is an effective method of identifying network anomalies because the malicious payload is an inevitable part of the network attack traffic [34]. However, the encryption protocol is widely used in real network communication that makes it hard to detect the payloads. Although some researchers focus on encryption traffic classification [35], these methods can only classify the network traffic with known patterns, but can not detect the 0-day attacks. Consequently, the attackers can use the proposed method to generate network traffic with encrypted payloads to evade payload-based detection.



(a) NSL-KDD (b) UNSW-NB15 (c) CICIDS2017

Fig. 10. Time cost and packets number of the generated unidirectional flow.



(a) NSL-KDD (b) UNSW-NB15 (c) CICIDS2017

Fig. 11. Time cost and packet number of the generated bidirectional flow.

7. Conclusions

In this paper, we propose the AIDAE framework against the existing IDSs. Compared with other generation methods, our proposed AIDAE can not only generate features matching normal feature distribution, but also keep the correlation between the generated continuous and discrete features. The attackers can initiate attacks by using the generated features to generate network traffic flow. Experiments prove that our proposed framework can indeed generate features to disable the baseline IDS. In our future work, we have a significant interest in defending against this type of attack. Moreover, we will research the evasion attacks based on the semantical level information.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] N. Sun, J. Zhang, P. Rimba, S. Gao, L.Y. Zhang, Y. Xiang, Data-driven cybersecurity incident prediction: a survey, IEEE Communications Surveys & Tutorials 21 (2) (2018) 1744–1772.
[2] Q. Liu, P. Li, W. Zhao, W. Cai, S. Yu, V.C. Leung, A survey on security threats and defensive techniques of machine learning: a data driven view, IEEE Access 6 (2018) 12103–12117.
[3] J. Zhang, Y. Xiang, Y. Wang, W. Zhou, Y. Xiang, Y. Guan, Network traffic classification using correlation information, IEEE Trans. Parallel Distr. Syst. 24 (1) (2012) 104–117.
[4] I. Corona, G. Giacinto, F. Roli, Adversarial attacks against intrusion detection systems: taxonomy, solutions and open issues, Inf. Sci. 239 (2013) 201–225.
[5] P. Mishra, V. Varadharajan, U. Tupakula, E.S. Pilli, A detailed investigation and analysis of using machine learning techniques for intrusion detection, IEEE Communications Surveys & Tutorials 21 (1) (2018) 686–728.
[6] G. Apruzzese, M. Colajanni, L. Ferretti, M. Marchetti, Addressing adversarial attacks against security systems based on machine learning, in: 2019 11th International Conference on Cyber Conflict (CyCon), vol. 900, IEEE, 2019, pp. 1–18.
[7] X. Chen, C. Li, D. Wang, S. Wen, J. Zhang, S. Nepal, Y. Xiang, K. Ren, Android hiv: a study of repackaging malware for evading machine-learning detection, IEEE Trans. Inf. Forensics Secur. 15 (2019) 987–1001.
[8] Z. Lin, Y. Shi, Z. Xue, Idsgan: Generative Adversarial Networks for Attack Generation against Intrusion Detection, arXiv preprint arXiv:1809.02077.
[9] Y. Cao, J. Yang, Towards making systems forget with machine unlearning, in: 2015 IEEE Symposium on Security and Privacy, IEEE, 2015, pp. 463–480.
[10] F. Zhang, P.P. Chan, B. Biggio, D.S. Yeung, F. Roli, Adversarial feature selection against evasion attacks, IEEE transactions on cybernetics 46 (3) (2015) 766–777.
[11] A. Kantchelian, J.D. Tygar, A. Joseph, Evasion and hardening of tree ensemble classifiers, in: International Conference on Machine Learning, 2016, pp. 2387–2396.
[12] P. Russu, A. Demontis, B. Biggio, G. Fumera, F. Roli, Secure kernel machines against evasion attacks, in: Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security, ACM, 2016, pp. 59–69.
[13] L. Chen, Y. Ye, T. Bourlai, Adversarial machine learning in malware detection: arms race between evasion attack and defense, in: 2017 European Intelligence and Security Informatics Conference (EISIC), IEEE, 2017, pp. 99–106.
[14] L. Liu, O. De Vel, Q.-L. Han, J. Zhang, Y. Xiang, Detecting and preventing cyber insider threats: a survey, IEEE Communications Surveys & Tutorials 20 (2) (2018) 1397–1417.
[15] R. Coulter, Q. L. Han, L. Pan, J. Zhang and Y. Xiang, "Data-Driven Cyber Security in Perspective-Intelligent Traffic Analysis," in IEEE Transactions on Cybernetics, vol. 50, no. 7, pp. 3081-3093, July 2020, doi: 10.1109/TCYB.2019.2940940.
[16] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, D. Mukhopadhyay, Adversarial Attacks and Defences: A Survey, arXiv preprint arXiv:1810.00069.
[17] L. Tong, B. Li, C. Hajaj, C. Xiao, N. Zhang, Y. Vorobeychik, Improving robustness of ml classifiers against realizable evasion attacks using conserved features, in: 28th USENIX Security Symposium, vol. 19, USENIX Security, 2019, pp. 285–302.
[18] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, F. Roli, Evasion attacks against machine learning at test time, in: Proc Conf, ECML-PKDD, Springer, 2013, pp. 387–402.
[19] S. Yu, S. Guo, I. Stojmenovic, Fool me if you can: mimicking attacks and anti-attacks in cyberspace, IEEE Trans. Comput. 64 (1) (2015) 139–151.
[20] M. Rigaki, S. Garcia, Bringing a gan to a knife-fight: adapting malware communication to avoid detection, in: Proc IEEE Symp Secur Priv Workshops, SPW, 2018, pp. 70–75.
[21] W. Hu, Y. Tan, Generating Adversarial Malware Examples for Black-Box Attacks Based on gan, arXiv preprint arXiv:1702.05983.
[22] I. Rosenberg, E. Gudes, Bypassing system calls–based intrusion detection systems, Concurrency Comput. Pract. Ex. 29 (16) (2017), e4023.
[23] S. Rezaei, X. Liu, Deep learning for encrypted traffic classification: an overview, IEEE Commun. Mag. 57 (5) (2019) 76–81.
[24] J.J. Zhao, Y. Kim, K. Zhang, A.M. Rush, Y. LeCun, Adversarially regularized autoencoders, in: Proc Int Conf, ICML, 2018, pp. 5897–5906.
[25] M. Tavallaee, E. Bagheri, W. Lu, A.A. Ghorbani, A detailed analysis of the kdd cup 99 data set, in: Proc IEEE Symp, CISDA, 2009, pp. 1–6.
[26] N. Moustafa, J. Slay, Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set), in: Proc Conf, MilCIS, 2015, pp. 1–6.
[27] I. Sharafaldin, A.H. Lashkari, A.A. Ghorbani, Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization, 2018.
[28] B. Fuglede, F. Topsoe, Jensen-shannon divergence and hilbert space embedding, in: International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings, IEEE, 2004, p. 31.
[29] H. Cheng, L. Pan, Method and Apparatus for Network Traffic Simulation, Aug. 22 2017 uS9740816B2.
[30] J. Tang, D. Wang, X. Zhou, L. Dong, Q. Yan, X. Zhang, H. Zhi, J. Zhang, Y. Wu, H. Jin, A Method for Generating Network Traffic Data Stream Based on the Feature, Nov. 13 2018, cN105049277B.
[31] S. Aljawarneh, M. Aldwairi, M.B. Yassein, Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model, Journal of Computational Science 25 (2018) 152–160.
[32] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing Properties of Neural Networks, arXiv preprint arXiv:1312.6199.
[33] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in: International Conference on Learning Representations, 2015.
[34] H. Liu, B. Lang, M. Liu, H. Yan, Cnn and rnn based payload classification methods for attack detection, Knowl. Base Syst. 163 (2019) 332–341.
[35] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescapé, Mobile encrypted traffic classification using deep learning: experimental evaluation, lessons learned, and challenges, IEEE Trans Netw Serv Man.