2021 International Conference on Computer Communications and Networks (ICCCN) | 978-1-6654-1278-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCCN52240.2021.9522215

Def-IDS: An Ensemble Defense Mechanism Against Adversarial Attacks for Deep Learning-based Network Intrusion Detection

Jianyu Wang*, Jianli Pan*, Ismail AlQerm*, and Yuanni Liu†
*Department of Computer Science, University of Missouri-St. Louis, St. Louis, Missouri, USA
†Institute of Future Network Technologies, Chongqing University of Posts and Telecommunications, Chong Qing, China
Email: *{jwgxc, pan, alqermi}@umsl.edu, †liuyn@cqupt.edu.cn

***Abstract*—Network intrusion detection plays an important role in the Internet of Things systems for protecting devices from security breaches. Facing challenges of the rapidly increasing amount of diverse network traffic, recent research has employed end-to-end deep learning-based intrusion detectors for automatic feature extraction and high detection accuracy. However, deep learning has been proved vulnerable to adversarial attacks that may cause misclassification by imposing imperceptible perturbation on input samples. Though such vulnerability is widely discussed in the image processing domain, very few studies have investigated its perniciousness against network intrusion detection systems (NIDS) and proposed corresponding defense strategies. In this paper, we try to fill this gap by proposing Def-IDS, an ensemble defense mechanism specially designed for NIDS, against both known and unknown adversarial attacks. It is a two-module training framework that integrates multi-class generative adversarial networks and multi-source adversarial retraining to improve model robustness, while the detection accuracy on unperturbed samples is maintained. We evaluate the mechanism over CSE-CIC-IDS2018 dataset and compare its performance with the other three defense methods. The results demonstrate that Def-IDS is able to detect various adversarial attacks with better precision, recall, F1 score, and accuracy.**

***Index Terms*—Network intrusion detection, adversarial attacks, deep learning**

I. Introduction

With the accelerating expansion in the number of Internet of Things (IoT) devices and the scale of computer networks, threats of network intrusions have severely grown in the past decade. By attacking vulnerable devices and network facilities, cybercriminals are able to compromise the confidentiality of communication, the availability of network services, and the integrity of data. According to Cisco's 2020 Annual Cybersecurity Report [1], the average cost of a security breach is $3.92M per company. The critical IoT infrastructures and services, such as public healthcare, transportation and power grid, are at risk of disruptive adversaries including malware, spam, sandbox evasion, abusive uses of cloud services, and botnet-based Distributed Denial of Service (DDoS) attacks [2]. In order to defend IoT security, deep learning-based network intrusion detection systems (NIDS) have been applied recently to identify malicious instances from massive and heterogeneous IoT traffic [3]. Deep learning outperforms the traditional signature-based and machine learning-based solutions at detection accuracy and scalability, because of its unique advantages such as feature extraction, non-convex optimization, and end-to-end learning model [4], [5].

Though deep learning is promising to build advanced intrusion detectors, researchers have found that deep neural networks (DNN) are vulnerable to adversarial attacks and thus the detectors can be compromised [6]. Specifically, the adversarial attacks add minor manipulation on the input samples, called adversarial examples, to mislead the detectors to misclassification. These attacks occur in the DNN deployment phase without tampering any model parameters. Adversarial examples exist due to two main reasons [7]: 1) The gap between the learned and the optimal model decision boundary commonly exists, due to the difference between training data and real data distribution. 2) The high degree of model linearity in neural networks is another weakness because it allows adversarial examples to be produced through gradient sign methods near the manifold of the training data. For the NIDS, if adversarial examples are found, the deep learning-based detectors can be severely compromised. As illustrated in Fig.1, an adversarial example (marked as red dots) from certain intrusion class but outside the learned model boundary would be misidentified as normal traffic, and vice versa. Such false negative or false positive detection results would dramatically increase the false alarm rates and undermine the NIDS function.

In the NIDS domain, some research works have investigated the adversarial attacks against deep learning-based intrusion detection [8], [9], [14], [15], but very few defense methods were proposed [21]–[23]. There are three main difficulties to design an effective defense. First, how does it mitigate the threat of various known attacks and new emerging attacks? Second, how does it maintain the high detection accuracy on clean inputs while the robustness is improved? Third, how to keep the functionality of IoT network traffic features in order to make sure the intrusions valid? Unfortunately, the existing defense methods are constrained by: 1) low detection accuracy and high deployment cost against limited known attacks, as well as no consideration of emerging unknown attacks; 2) degraded detection accuracy on unperturbed inputs; 3) ignoring the preservation of feature attributes and adopting the outdated datasets such as KDD99 and NSL-KDD.

In this paper, we propose a novel ensemble adversarial

978-1-6654-1278-0/21/$31.00 ©2021 IEEE

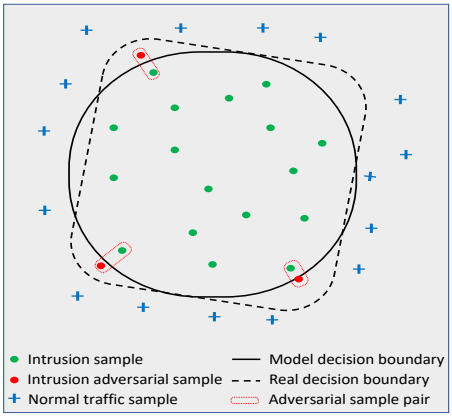

Fig. 1. Adversarial examples located between the real intrusion decision boundary and the decision boundary learned by deep learning model.

retraining-based defense mechanism, named Def-IDS, to enhance the robustness of deep learning-based intrusion detectors against adversarial attacks. Our work has three key merits in handling the three constraints: 1) it resists both known and unknown attacks while guaranteeing the intrusion detection accuracy; 2) it enables efficient one-time retraining for the multi-class detector rather than costly retraining for every intrusion class; 3) it adopts a recent benchmark dataset without compromising the functionality of traffic features. To achieve the above merits, our contributions include:

- We design a two-module ensemble mechanism that integrates both generative adversarial networks and adversarial retraining techniques. The integration not only preserves the intrusion detector's accuracy on the clean samples, but also significantly improves the robustness against adversarial attacks.
- In the first module, we propose a multi-class generative adversarial networks (MGAN)-based dataset enhancement method. It enhances the original training dataset by simultaneously oversampling the multi-class intrusions in order to reduce the gap between training and real data distribution. We extend the generative ability of the original GAN from a single class to multiple classes for cost efficiency. By applying the enhanced dataset into the training process, the intrusion detector is more robust against both known and unknown adversarial attacks.
- In the second module, we propose a multi-source adversarial retraining (MAT) model. It retrains the detectors with various types of adversarial examples in order to further smooth the decision boundaries and deceases the adversarial data space. MAT not only resists the specific adversarial attacks but also partially transfers resistance against one attack to another.
- We implement the state-of-the-art adversarial attacks and evaluate the performance of our mechanism using CSE-CIC-IDS2018 dataset [10]. Experiments show that Def-IDS achieves significant robustness improvement and is a competitive defense strategy.

The rest of the paper is organized as follows. Section II discusses the recent research on deep learning-based NIDS adversarial attacks and defense attempts. Section III analyzes threat model of adversarial attacks. In section IV, we describe the proposed ensemble defense mechanism. Section V presents the mechanism implementation and the evaluation results. Finally, the paper is concluded in Section VI.

II. Related Work

Recent research has exposed the vulnerability of DNN to adversarial attacks, and most of the studies focused on the attacks against the models for image processing tasks. Currently, limited works conducted experiments to launch adversarial attacks and proposed defense methods in the NIDS domain.

For the attack methods, Warzynski et al. [8] attempted to use Fast Gradient Sign Method (FGSM) attack [11] against the intrusion detector which was built with a three-hidden-layer neural network. The results showed that the detector was seriously compromised as all perturbed test examples were misclassified as legitimate. Wang et al. [9] investigated four popular white-box attacking algorithms and employed them on a two-hidden-layer feed (FNN) with dropout function. The evaluation showed that the attacks were able to impose perturbations on different features with different efficiency. However, neither of these two works considered the perturbation constraints of functional features. Lin et al. [14] proposed to generate adversarial examples in the black-box setting against NIDS using the generative adversarial networks (IDS-GAN) where only the non-functional features were utilized to inject malicious perturbations. Yang et al. [15] tested two more black-box attacks including substitute model [16] and ZOO [17], where the latter achieved better effect but required more computation. However, these two works generated attacks based on the outdated datasets and placed no constraints on perturbation magnitude.

For the defense strategies, Usama et al. [21] briefly mentioned using the adversarial retraining for resisting IDS-GAN attack. However, the work lacked detailed discussion how to implement and configure the retraining. Besides, their method proposed to build one GAN model for one class of intrusions which would cause high computation overhead. Ibitoye et al. [22] analyzed the attack success rate on FNN and self-normalization neural network (SNN). FGSM and Basic Iteration method (BIM) attacks were implemented on the IoT botnet dataset. Even though their experiments demonstrated that SNN had better robustness than FNN, but the overall detection rate of adversarial examples was still less than 60%. Pawlicki et al. [23] introduced an extra adversarial attack filter working independently with the intrusion detector to differentiate between adversarial and non-adversarial examples. Our proposal is different to this work because we aim to improve the robustness of intrusion detector directly.

To the best of our knowledge, there are very limited works for improving the robustness of deep learning-based NIDS in order to mitigate the impact of adversarial examples. Different

from the existing works, our ensemble mechanism is a cost-efficient and effective defense method against both known and unknown attacks, while the intrusion detection performance is preserved on the clean samples.

III. Adversarial Attack Threat Models

In this section, we first analyze the vulnerability of deep learning-based intrusion detection. Then the adversarial attack model is introduced from the aspects of attack formulation, taxonomy, and attack algorithms adopted in our study.

A. Vulnerability

Adversarial attacks utilize two main vulnerabilities of deep learning models: 1) inaccurate approximation to the real distribution; and 2) linearity of neural network models. Correspondingly, the attacks either search for adversarial examples indirectly near the decision boundaries or generate the examples directly based on the parameters of models.

First, the inaccurate approximation means that the classification boundaries of the trained models are not 100% equivalent to the expected boundaries in the real data distribution. Such a gap may result from data and model: 1) the size of the training dataset may not be sufficient to present all patterns of real distribution. Moreover, the class imbalance in the dataset, if existed, would cause decision bias; 2) the model structure and related hyperparameters configuration affect the learning ability of models, including choice of layers number, activation functions, and optimization algorithms.

Second, the linearity of neural networks can be leveraged to produce adversarial examples in high-dimensional input data space. Assuming that an adversarial example is $x_{adv} = x + \eta$ and the amount of perturbation is η, the linearity magnifies the effect of small perturbation to a large deviation that is enough to change the decision of the classifier. Even if the single η_i on each data feature is minor, the magnitude of the cumulative η will increase to $n \cdot \eta_i$ in n-dimensional space. Thus, a large dimensionality leads to a substantial change in the output of the end-to-end model. Though many neural networks employ non-linear activation functions such as rectified linear unit (ReLU), they still behave linearly because the models are trained in linear ways to obtain the gradient-based loss during model optimization [11].

B. Adversarial Attack Model

1) Attack Formulation: We denote the deep learning-based classifier by C, as the attack target. The classification results of adversarial example x_{adv} and original data x are represented with $C(x_{adv})$ and $C(x)$ respectively. The optimization goal of adversarial attacks is formulated as:

$$\max_{\|x_{adv}-x\|_p \leq \epsilon} |C(x_{adv}) - C(x)| \tag{1}$$

where $\|\cdot\|_p$ is the metric of distances in p-norm. In order to guarantee that an adversarial example is still in its original class, the difference between x_{adv} and x is constrained by $\|x_{adv} - x\|_p \leq \epsilon$, where the limitation of perturbation size is ϵ. In NIDS domain, adversarial perturbation needs to preserve the functionality of network traffic. Therefore, ϵ should be small enough for not changing the traffic pattern but x_{adv} is misclassified by the classifier C.

2) Taxonomy of Adversarial Attacks: The adversarial attack methods can be categorized by the knowledge of the target models into *white-box* and *black-box* attacks. First, white-box attacks assume that the adversary knows all the parameters and configurations of DNN model, such as architecture, network weights, and activation functions. Thus, the same model is rebuilt by the adversary to find effective ways to generate perturbed samples. Second, black-box attacks assume that the adversary has no knowledge of the model except the inputs and the corresponding outputs. A synthetic dataset is collected, and a substitute model is trained to approximate the targeted model. Researchers [6] have found that the adversarial examples can transfer between similar model architectures for the same classification purpose. Such transferability allows the adversary to apply the adversarial examples against the substitute model to the victim model.

In this paper, we use black-box attacks since the intrusion detector are usually confidential in practice.

3) Attack Algorithms: We adopt four cost-efficient adversarial attacks to evaluate the defense performance of our proposal.

First, we adopt Fast Gradient Sign Method (FGSM) [11], which is a relatively less computation-intensive attack. It utilizes the linear behavior of neural networks to perform gradient descent in order to direct how the clean samples should be manipulated. The optimal perturbation is computed using back-propagation algorithm.

Second, we adopt Basic Iterative Method (BIM) [18]. It extends the FGSM by performing gradient calculation multiple times with small step size. In each iteration, the value of perturbation is clipped to avoid large change on traffic features.

Third, DeepFool [13] is implemented to find the closest path from the original data point to the decision hyperplane. The projection to the hyperplane determines the minimal perturbation used for generating adversarial examples.

Fourth, Jacobian-based Saliency Map attack (JSMA) [12] is implemented, which utilizes the sensitive features of x that can make most significant changes to the classification result. Saliency map is built to understand the mapping relation between inputs and outputs.

For the readers who are interested in the principles of the above attack algorithms, we refer them to the works [11]–[13], [18]. The details of their usage for launching adversarial attacks are introduced in the next two sections.

IV. Proposed Def-IDS Defense Mechanism

A. Mechanism Overview

In this section, we introduce a novel ensemble defense mechanism against adversarial attacks. The goal is to mitigate the misclassification rate of adversarial examples as much as possible. Fig. 2 shows the rationale of our mechanism. We utilize the upper bound and lower bound to characterize the robustness degree of a DNN model and show how our

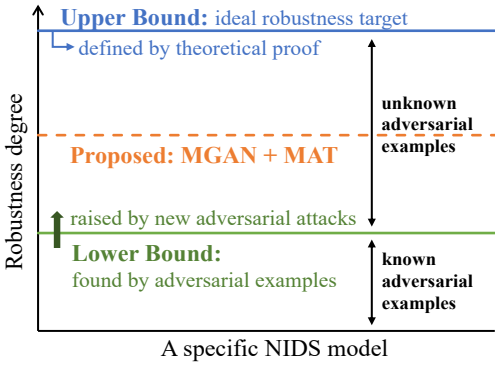

Fig. 2. The upper bound and the lower bound that characterize the robustness degree of a NIDS model against adversarial examples.

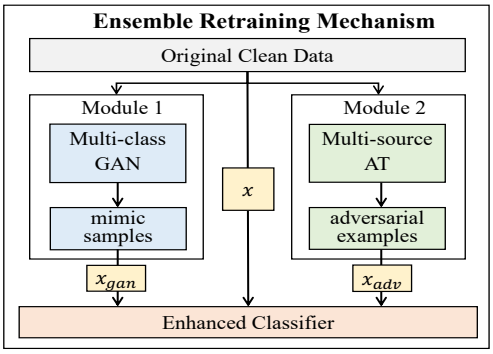

Fig. 3. Framework of the proposed Def-IDS ensemble retraining mechanism.

mechanism differs from the existing methods. The area under the bounds is the data space that adversarial examples may exist. Specifically, the upper bound is the ideal robustness defined by a theoretical proof. A model with upper-bound robustness can correctly classify the adversarial examples within a quantifiable guarantee for whatever known or unknown attacks. In this case, the theoretical proof guides the model training process to narrow down the gap between model and true decision boundaries into a certain level. However, there is no widely accepted proof method yet [6]. Correspondingly, the lower bound is defined by adversarial examples generated from the known attacks. The deep learning models trained with these adversarial examples are resistant to the related attacks. The lower bound would be raised when new attacks emerge. The existing defense method [21] for NIDS is built on the lower bound aiming at identifying a known adversarial attack, but the defense against the other attacks located between the upper bound and the lower bound is still missing.

Therefore, we propose an ensemble retraining mechanism that aims at approximating the upper bound against both known and unknown adversarial attacks. The mechanism consists of two complementary modules, as presented in Fig. 3. In the first module, we design a cost-efficient **multi-class generative adversarial network (MGAN)**-based method that: 1) generates mimic samples for multi-class intrusions using a single GAN model; 2) augments training dataset near the model decision boundary to reduce the learning gap; 3) improves the overall model robustness without engaging any specific adversarial attack algorithms. In the second module, we create a **multi-source adversarial retraining (MAT)** method to further improve the model robustness by integrating the adversarial examples from various attacks into the training process as the regularization of decision boundary. We find that the robustness against one attack is also partly effective on other attacks. Therefore, the multi-source adversarial examples-based retraining achieves better performance than the single source.

By combining the two modules, the proposed "MGAN+MAT" approach significantly enhances the defense ability of the NIDS model. It remedies more adversarial data space than the lower boundary. In addition, it tackles the fitting bias occurred in the existing works. This bias severely degrades the detection performance on clean data once the adversarial examples are employed in the retraining. Our mechanism achieves good detection accuracy on both clean data and adversarial examples.

B. Module 1: Multi-class GAN-based Retraining

Since the regular training of classifier C is performed on the limited training dataset, it is feasible to improve the generalization of the model in the testing environment with more real-distributed samples. However, in practice, there may be no access or too expensive to get more labeled data. Besides, the increasing types of intrusions make the data collection more costly. To solve these limitations, we propose a novel GAN framework to learn the data distribution of all classes of traffic in one model. The trained MGAN is able to generate new samples that could be regarded as the perturbed data but still belongs to the original classes. Then, we use the augmented dataset to retrain the intrusion detector.

We first introduce the principles of the original GAN model [19] and then redesign it for our use. Its unique advantage is generating high-quality mimic samples by learning the distribution of a given dataset in the adversarial setting. GAN contains two components: a generator model (G) and a discriminator model (D), which mutually improve each other in a competitive minimax game. The generator implicitly learns the underlying data distribution without directly seeing training data and produces mimic samples to cheat the discriminator. Meanwhile, the discriminator learns the class boundary based on the real data and tries to identify the fakes sent from the generator. The model can be formulated as follows. Let z be the latent variables in the manifold of G in low dimensionality, x be the real traffic samples, and $x' = G(z)$ be the output of G and generated sample. The optimization goal of G is to minimize the possibility of x' being recognized, formulated as $log(1 - D(G(z)))$. On the other hand, the optimization goal of D is to maximize the recognition rate for both x and x', formulated as $log(D(x)) + log(1 - D(G(z)))$. Overall, the optimization of the whole model is defined as:

$$\min_G \max_D E_{x\sim p_x}[log(D(x)] + E_{z\sim p_z}[log(1 - D(G(z)))] \quad (2)$$

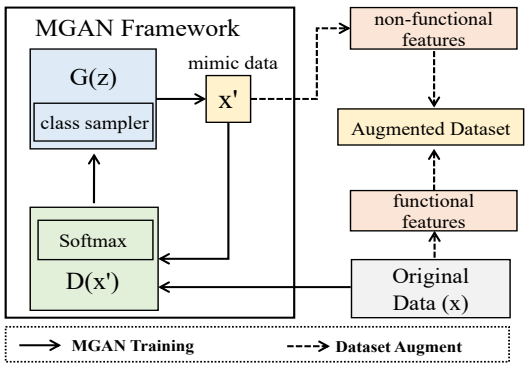

Fig. 4. Multi-class GAN (MGAN)-based retraining with augmented dataset.

where p_x is the real data distribution and p_z is the feature space distribution learned by G. With such an adversarial setting, G tends to explore mimic data near the class boundary in order to fool D. The generated data is potentially unknown adversarial examples against the intrusion detector.

However, the original GAN cannot meet the requirement that a single model learns multi-class traffic distribution and generate class-specific samples. It is because the discriminator model is a binary classifier only identifying fake or real data. If using the original GAN, we have to train an individual model for every intrusion class, which will cause too much overhead. In order to build a multi-class model, we integrate two variant techniques: AC-GAN [20] and SGAN [24]. The former introduces the function of class sampling into the generator for producing samples with specified labels. The latter transforms the discriminator to a multi-class classifier with an extra Softmax layer to predict the probability distribution of a sample over various labels. Through such integration, we are able to train single MGAN model to sample new data among all traffic classes. In detail, our retraining design for detector C is shown in Fig. 4. During the training phase, the generator learns the distribution of different classes and the discriminator offers both classifications results and fake or real judgment. Assuming that the true label of x is y_x and the target label of G is y_z, the optimization goal is redefined as:

$$\min_G \max_D E_{x\sim p_x}[log(D(x,y_x)] + E_{z\sim p_z}[log(1-D(G(z,y_z)))] \tag{3}$$

In order to obtain an optimized generator, the ideal stop criteria is that the dual models reach a Nash Equilibrium [19] where neither G or D can improve by changing any parameters of neural networks. However, since the optimization is based on stochastic gradient descent, the learning loss of G and D would oscillate and the explicit equilibrium is hard to determine. A commonly used solution is to define a fixed number of training iterations, but it is not accurate enough. Therefore, we define a stop indicator T based on the moving average calculated by a series of values after each training iteration. The values are extracted from the loss function of G that measures its current generative ability: $G_{loss} = \sigma(D(G(z)), y_{target})$. σ is the cross-entropy between the discrimination result of generated data and the target class. T is formulated as $T = \frac{1}{m}\sum_i^m G_{loss\_i}$, where m is the number of the latest iterations. If a moving average is lower than a pre-defined threshold TH, MGAN is deemed in a steady status and the generator becomes approximately optimal.

After obtaining the generator, we retrain the intrusion classifier C with newly generated samples to smooth the decision boundaries and improve the generalization capability of real data distribution. For preserving the functionality of traffic samples, we filter out the functional features and use the non-functional features to generate new samples. Specifically, we first find the nearest data point $d(x')$ in the original training dataset only based on the generated functional features. Then we replace the non-functional features of $d(x')$ with MGAN's generated non-functional features. In this way, the newly generated samples are mixed with the original dataset to retrain model C, whose configurations keep unchanged, including neural network architecture, optimizer, and loss function.

C. Module 2: Multi-source Adversarial Retraining

Multi-source adversarial retraining is the process that trains the detector C explicitly with multiple kinds of adversarial examples. It helps the detector resist the related attacks and potentially be more robust to the other. The robustness is improved because the retraining mitigates the adversarial data space and softens the decision boundary.

In order to perform the retraining for the NIDS model, we ensure the following three essential prerequisites. First, the perturbation on a traffic sample is distributed among the non-functional features, where the perturbation magnitude is constrained to a minor scale. Second, functional traffic features such as network protocol cannot be perturbed. Third, the valid adversarial attack methods should be time-efficient to generate adversarial examples. For example, C&W [25], Boundary [26], and One-Pixel [27] attacks are either costly or change only one feature, which are infeasible for adversarial retraining. In our study, we implement the four attacks, as described in Section III, in the black-box setting and perturb only non-functional features. Meanwhile, the perturbation constraint is set to 2% of the original feature value.

Fig. 5 presents the workflow of adversarial retraining. Specifically, the retraining inputs of the classifier come from clean data x and multi-source adversarial examples x_{adv}. The classifier is optimized to assign correct labels y to the adversarial examples. A substitute model S is first trained with the clean data to achieve the same performance as classifier C. Then attacks are performed on the substitute S to generate adversarial examples. The retraining of C utilizes the x_{adv} from S that also compromise C. We observe the retraining on one specific attack can also help resist the other three attacks, though the effect is limited. Such a phenomenon will be further evaluated in the next section.

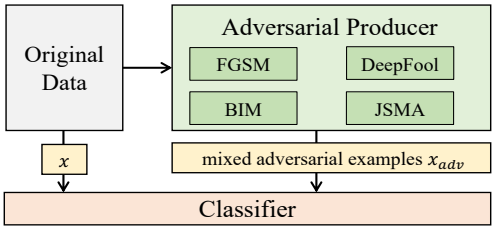

Fig. 5. Multi-source adversarial retraining (MAT) structure to tune classifier decision boundary.

Given the above description, we define the objective of the adversarial training as:

$$\min_{\|x_{adv}-x\|_p \leq \epsilon} E\left[\lambda L(C(x_{adv}), y) + (1-\lambda) L(C(x), y)\right] \quad (4)$$

In this case, the loss function of C involves the prediction error of both clean and adversarial data, where λ is the weight adjusting ratio. Through adversarial training, we proactively defend the intrusion detector to mitigate the threat of multiple adversarial attacks. It is worth noting that this idea is extensible to future emerging attacks against NIDS.

D. Ensemble Adversarial Retraining

To maximize defense ability against adversarial attacks, we combine MGAN and MAT modules together. The baseline classifier C is improved by the enhanced dataset and proactively resists adversarial examples. As illustrated in Fig. 3, our ensemble retraining consists of three data sources: 1) mimic data x_{gan} from MGAN; 2) adversarial examples x_{adv} obtained from multiple attacks; 3) the original clean data x. Therefore, we define the overall optimization goal of the ensemble retraining as:

$$\begin{aligned} \min E[&w_1 L(C(x), y) + w_2 L(C(x_{gan}), y) \\ &+ w_3 L(C(x_{adv}), y)] \end{aligned} \quad (5)$$

where a constraint exists $(w_1 + w_2 + w_3) = 1$.

To sum up, the ensemble mechanism is represented by a four-step training process summarized in Algorithm 1. Such retraining methodology explores the potential adversarial data space, and then utilize them to fine tune of the model decision boundary. The evaluation of robustness improvement on intrusion detection task is discussed in the next section.

Algorithm 1 Ensemble Defense Mechanism of NIDS Classifier for Improving Robustness against Adversarial Attacks.

Require: Deep neural network classifier C, original intrusion training dataset x, and adversarial attack types n.
Ensure: Trained classifier C with enhanced robustness against unknown and known adversarial examples.
BEGIN
Step 1: Implement baseline classifier
Initialize the classifier with model parameters θ.
Train the classifier with original dataset x, get a baseline classifer C_{base}.
Step 2: Generate new samples from multi-class GAN
Initialize the generator G and the discriminator D in MGAN framework.
while Training stop indicator $T >$ stop criteria threshold TH **do**
Train G and D by mutually updated Min-Max optimization .
end while
Generate a new set of mimic samples x_{gan} with optimized G.
Step 3: Obtain adversarial examples from multiple attacks
for (attack type i=1 to n) **do**
perform adversarial attack algorithms i to generate the adversarial examples x_{adv_i}.
end for
Aggregate a new set of adversarial examples x_{adv}.
Step 4: Ensemble retraining
Retrain C_{base} with new combined dataset with x, x_{gan}, and x_{adv}.
END

V. Evaluation

In this section, we first introduce the preprocessing of the benchmark intrusion dataset and the evaluation metrics. Next, we build a baseline intrusion detector and launch various adversarial attacks against it. Then we implement the proposed ensemble mechanism and measure its defense ability. Finally, the mechanism cost is estimated to validate its applicability.

A. Dataset and Metrics

1) Dataset Pre-processing: We employ CSE-CIC-IDS2018 dataset [10] published by Canadian Institute for Cybersecurity (CIC) for evaluation. Compared to the outdated KDD-99 and NSL-KDD datasets, it is up-to-date with comprehensive attacks implementation and more balanced data. Besides, features in the CIC dataset are numeric, continuous, and non-functional. This dataset contains normal network traffic and seven classes of intrusions, including Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks, and infiltration of the network from inside. CIC extracts 80 features from the packet-level and flow-level traffic such as number, size, and duration.

For our experiments, we preprocess the dataset in three steps. First, since the values of features are in different scales, we perform Min-Max standardization on the features to rescale them in the range [0, 1]. Second, there exist four features (dst port, protocol, flow byts/s, flow pkts/s) with too many empty or infinity values and one feature (timestamp) unrelated to traffic. So we drop these five features to keep the dataset consistent and clean, where 76 features in total are reserved. Third, we divide the datasets into training, validation, and testing part in the ratio (8:1:1) by randomly sampling in normal and intrusion samples.

2) Evaluation Metrics: The detection performance on adversarial examples and clean samples are measured by four metrics: Precision, Recall, F1 score, and Accuracy. Taking intrusion as positive class and normal traffic as negative class, we define the metrics as follows:

Precision indicates how many correct classifications in the all results that are identified as intrusions: $P = \frac{TP}{TP+FP}$.

Recall presents the proportion of correctly identified intrusions over all true intrusions, formulated as: $R = \frac{TP}{TP+FN}$.

F1 score is the harmonic mean of precision and recall, which is effective measurement, formulated as: $F1 = \frac{2 \cdot P \cdot R}{P+R}$.

TABLE I
DEFENSE ABILITY OF BASELINE CLASSIFIER C_{base}.

Attacks	Precision	Recall	F1 score	Accuracy
FGSM	0.324	0.46	0.38	0.413
BIM	0.398	0.525	0.452	0.425
DeepFool	0.425	0.412	0.418	0.434
JSMA	0.354	0.312	0.331	0.328

Accuracy refers to the percentage of correctly classified normal traffic and intrusion samples over the total results: $Acc = \frac{TP+TN}{TP+TN+FP+FN}$.

where True Positive (TP) is the number of intrusion samples that are correctly classified; False Positive (FP) is the number of normal samples misclassified as intrusion; False Negative (FN) is the number of intrusion samples misclassified as normal; True Negative (TN) is the number of normal samples that are correctly classified.

B. Baseline Detector Implementation

1) Detector Implementation: We implement a baseline classifier C_{base} as the detector to identify intrusions from normal traffic. C_{base} is constructed by four sequential layers, including one input layer, two hidden layers and one output layer (76-128-64-8). The hidden layers are fully connected layers with the ReLU activation function. The output layer adopts Softmax function for multi-class traffic classification. To build the testbed, we employ Keras library with Tensorflow backend [28] as deep learning platform and Ubuntu 18.04 operating system running on a desktop with 3.6GHz CPU and 16GB RAM. To train the model, we adopt Adam optimizer, 0.001 learning rate, 20 training epochs, 5000 mini-batch size, and categorical cross-entropy loss function. During the training process, ten-time cross validations are performed and the average metric value are calculated. After the training, the C_{base} is evaluated using the test dataset.

We evaluate the detection performance using the pre-defined metrics on both training and test sets. C_{base} achieves 0.997 and 0.993 F1 score, respectively. The results show that C_{base} is a well-performed detector on the clean data.

2) Adversarial Attacks against Baseline Classifier: To implement adversarial attacks, we use foolbox [29], an open-source python library, to generate adversarial examples. The attacks includes FGSM, BIM, DeepFool, and JSMA, as introduced in Section III We randomly choose 2,000 samples (500 for each attack) from the training dataset where 1,000 are normal traffic and the rest are intrusions. For remaining the attributes of the original traffic samples, the size ϵ of perturbation is limited to 0.02 (2%) where the measure metric is the mean squared error norm $p = 2$. By adding perturbations, the baseline detector C_{base} would misclassify the normal as intrusions and vice versa.

We first train a substitute model C_{sub} with two hidden-layers (128-32) using training dataset. Without preknowledge of the baseline model C_{base}, we use Adam optimizer, 0.002 learning rate, 20 training epochs, 3000 mini-batch size, and categorical cross-entropy loss function. The trained C_{sub} achieves the similar detection ability with C_{base} on the clean set, with an average 0.987 F1 score on both normal and intrusion samples. Then, the C_{sub} is attacked by the four adversarial attacks to generate 2,000 valid adversarial examples (4×500) with the perturbation constraint 0.02. Then, the adversarial examples from the C_{sub} are tested on the baseline model C_{base}. The attacking results are shown in Table I. C_{base} is compromised with less than 0.46 detection accuracy, which means more than half of the adversarial examples succeed in transferring from C_{sub} to C_{base}.

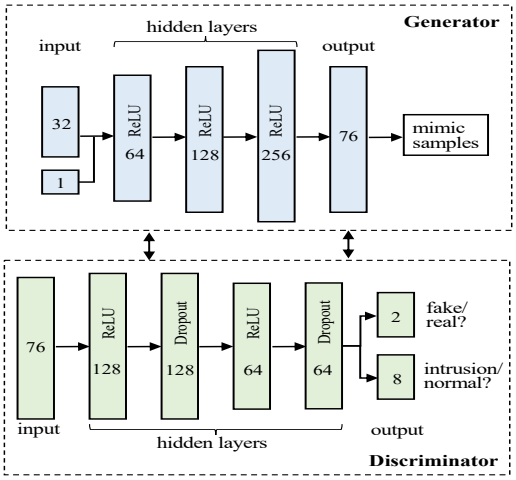

Fig. 6. Neural network structures of generator and discriminator in MGAN .

C. Def-IDS Defense Evaluation

1) Multi-class GAN-based Retraining: Multi-class GAN-based retraining enhances the baseline detector with the mimic new samples following the distribution of training data. Fig. 6 presents its implementation structure. Specifically, we construct the generator with a five-layer neural network ((32,1)-64-128-256-76), including two separate input layers for initializing latent features and class label, three hidden layers with ReLU activation functions for learning real data distribution, and one output layer for producing samples in real data space. The discriminator network consists of four layers (76-128-64-(2,8)), including one input layer, two hidden layers with ReLU activation functions, and two separate output layers for judging fake/real sample and class label correspondingly. Dropout function is used in the two hidden layers with $rate = 0.2$ to mitigate the overfitting during the training process. To optimize the networks, binary cross-entropy and categorical cross-entropy are used as loss functions for the real or fake identification and intrusion classification, respectively. We use Adam optimizer with learning rate 0.002 for both generator and discriminator, and set the stopping threshold T based on ten-time repeated experiments.

For obtaining the augmented dataset, we generate 10% more samples for each class of the original clean data. The retraining configurations remain same as the baseline. Then 2,000 adversarial examples generated from black-box attacks are used to

evaluate the defense ability of the new detector C_{gan}. Table II shows the evaluation results. We find that C_{gan} is able to resist more than 59% adversaries where the defense against JSMA has most significant increment. However, none of the metrics reach 0.8 that means MGAN-based retraining alone is not robust enough. At least 28.7% adversarial examples are unidentified in term of the best accuracy. The reason is that the generated samples do not aim at any specific attack but generally unknown attacks. In this way, C_{gan}'s model decision boundary is tuned to be insensitive to micro perturbations.

2) Multi-source Adversarial Retraining: Multi-source adversarial retraining proactively utilizes the adversarial examples during the training process to adjust model decision boundary. For each of four attacks, we retrain a detector C_{at} with (9:1) ratio of clean data and adversarial examples. Specifically, we generate 500 valid adversarial examples from each attack and aggregate them together. Then, the training process follows the loss function 4 where parameter λ is set to 0.9. For evaluation, we still use the black-box way to generate another 2,000 adversarial examples based on C_{base}. As illustrated in Table II, the detection metrics over different adversarial attacks increase range [0.91, 0.93]. Moreover, even if retrained on only one attack, we find that C_{at} has robustness improvement against the other attacks. As presented in Table III, for example, C_{at} trained with FGSM only is found able to achieve 0.846, 0.741, and 0.867 detection accuracy against BIM, DeepFool, and JSMA respectively.

3) Ensemble of Two Modules: To evaluate the robustness of our proposed ensemble mechanism, we retrain a classifier following the schema as introduced in Fig. 3. According to equation 5, the weight parameter is set to $\{w_1 = 0.8, w_2 = 0.1, w_3 = 0.1\}$ in order to make sure: 1) the influence of re-sampled data x_{gan} and adversarial examples x_{adv} on the decision boundary learning are equal; 2) the training process mainly relies on the original clean data x while x_{gan} and x_{adv} are used for adjusting the boundary. The adversarial examples are generated in black-box way from the four attacks. For each type of attacks, we generate 500 training examples and 500 evaluation examples separately. Table II shows the ensemble mechanism C_{ensem} achieves the best defense, 0.979 detection accuracy, better than any single retraining modules.

4) Accuracy on Clean Data: We also test the detection accuracy of C_{gan}, C_{at} and C_{ensem} on the clean intrusion data. As shown in Table II, adversarial retraining alone does harm the accuracy but the ensemble mechanism mitigates it by introducing the MGAN module.

D. Comparison with Other Works

We compare our mechanism with other recent defense methods proposed by Usama et al. [21], Ibitoye et al. [22], and Pawlicki [23]. Their detection ability is tested over adversarial examples that are generated in the same setting as the evaluation of C_{ensem}. Fig. 7 presents the precision, recall, F1 score, and accuracy metrics of the four evaluated methods. The results show that Def-IDS achieves the best and most stable performance.

TABLE II
DEFENSE ABILITY OF CLASSIFIERS $C_{gan}, C_{at}, C_{ensem}$.

Attacks	Models	Precision	Recall	F1 score	Accuracy
FGSM	C_{gan}	0.775	0.535	0.633	0.628
	C_{at}	0.933	0.912	0.922	0.927
	C_{ensem}	**0.978**	**0.967**	**0.972**	**0.974**
BIM	C_{gan}	0.795	0.461	0.582	0.597
	C_{at}	0.941	0.923	0.932	0.934
	C_{ensem}	**0.973**	**0.983**	**0.977**	**0.981**
DeepFool	C_{gan}	0.808	0.517	0.626	0.642
	C_{at}	0.934	0.918	0.926	0.931
	C_{ensem}	**0.989**	**0.976**	**0.984**	**0.983**
JSMA	C_{gan}	0.793	0.705	0.746	0.713
	C_{at}	0.931	0.917	0.924	0.923
	C_{ensem}	**0.981**	**0.977**	**0.978**	**0.979**
Clean Data (not perturbed)	C_{gan}	0.998	0.996	0.997	0.998
	C_{at}	0.965	0.978	0.971	0.963
	C_{ensem}	**0.998**	**0.997**	**0.998**	**0.998**

TABLE III
DEFENSE ABILITY OF MAT-BASED CLASSIFIER C_{at}.

Attack-based Retraining on	Adversarial Attacks Detection against			
	FGSM	BIM	DeepFool	JSMA
FGSM	0.924	0.846	0.741	0.867
BIM	0.742	0.931	0.853	0.773
DeepFool	0.624	0.756	0.925	0.692
JSMA	0.745	0.812	0.743	0.914

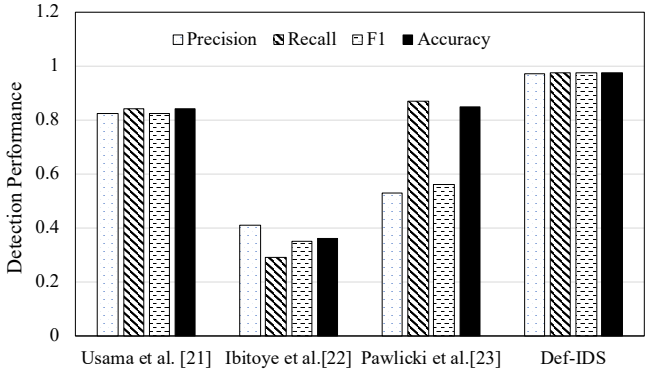

Fig. 7. Performance comparison between Def-IDS and other defense methods against adversarial examples.

E. Cost Estimation

At last, we measure the computation cost by time during the retraining process. For testing, we generate 2,000 adversarial examples applied in adversarial retraining. Given the training configuration and model architecture discussed in the previous evaluations, the list of cost is reported in Table IV. The training time of the baseline classifier is $12.86s$ on average. The adversaries generation takes from $5.34s$ to $27.84s$, where JSMA takes the most time and FGSM takes the least. The training of MGAN takes $20.43s$ while the sample generation uses negligible time. The final ensemble

retraining takes $15.62s$ with the enhanced dataset. In total, the whole ensemble mechanism takes $111.51s$, where the dataset generation occupies most of the time. Therefore, the practical implementation of the mechanism should consider the tradeoff between robustness and execution cost, especially for the dynamically updated IDS.

TABLE IV
COMPUTATION TIME COST OF ENSEMBLE RETRAINING MECHANISM.

Operations	**Time (s)**
Baseline Classifier Training	12.86
FGSM Advs Generation	5.34
BIM Advs Generation	16.15
DeepFool Advs Generation	13.27
JSMA Advs Generation	27.84
MGAN Training	20.43
Ensemble Retraining	15.62
Total	111.51

VI. CONCLUSION

In this paper, we designed and implemented an ensemble defense mechanism that significantly improved the robustness of deep learning-base network intrusion detector against both known and unknown adversarial attacks. By novelly integrating generative adversarial networks and adversarial retraining technology, our mechanism fine tuned the decision boundary of the detector to recognize multi-source adversarial examples and preserve its detection accuracy on clean inputs. The evaluation results demonstrated the effectiveness of our mechanism using the CSE-CIC-IDS2018 benchmark dataset. In the future, we plan to adapt the mechanism to real intrusion detection systems and more adversarial attacks, especially for IoT networks.

ACKNOWLEDGEMENTS

The work is supported in part by National Science Foundation (NSF) CNS core grant No. 1909520.

REFERENCES

[1] "Cisco 2020 Annual Cybersecurity Report," Cisco. [Online]. Available: https://www.cisco.com/c/en/us/products/security/security-reports.html.
[2] 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," in IEEE Communications Surveys & Tutorials, vol. 21, no. 3, pp. 2702-2733, thirdquarter 2019.
[3] K. AP. da Costa, J.P. Papa, C. O. Lisboa, R. Munoz and V. H. C. de Albuquerque, "Internet of Things: A survey on machine learning-based intrusion detection approaches," Computer Networks 151 (2019): 147-157.
[4] L. N. Tidjon, M. Frappier and A. Mammar, "Intrusion Detection Systems: A Cross-Domain Overview," in IEEE Communications Surveys & Tutorials, vol. 21, no. 4, pp. 3639-3681, Fourthquarter 2019.
[5] A. Abeshu and N. Chilamkurti, "Deep Learning: The Frontier for Distributed Attack Detection in Fog-to-Things Computing," in IEEE Communications Magazine, vol. 56, no. 2, pp. 169-175, Feb. 2018.
[6] I. Goodfellow, P. McDaniel and N. Papernot, "Making machine learning robust against adversarial inputs," Commun. ACM 61, 7 (June 2018), 56-66.
[7] X. Yuan, P. He, Q. Zhu and X. Li, "Adversarial Examples: Attacks and Defenses for Deep Learning," in IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 9, pp. 2805-2824, Sept. 2019.
[8] A. Warzynski and G. Kolaczek, "Intrusion detection systems vulnerability on adversarial examples," 2018 Innovations in Intelligent Systems and Applications (INISTA), Thessaloniki, 2018, pp. 1-4.
[9] Z. Wang, "Deep Learning-Based Intrusion Detection With Adversaries," in IEEE Access, vol. 6, pp. 38367-38384, 2018.
[10] CSE-CIC-IDS2018 Dataset. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html.
[11] I. Goodfellow, J. Shlens and C. Szegedy, "Explaining and harnessing adversarial examples," arXiv preprint arXiv:1412.6572 (2014).
[12] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik and A. Swami, "The Limitations of Deep Learning in Adversarial Settings," 2016 IEEE European Symposium on Security and Privacy (EuroS&P), Saarbrucken, 2016, pp. 372-387.
[13] S. Moosavi-Dezfooli, A. Fawzi and P. Frossard, "DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 2574-2582.
[14] Z. Lin, Y. Shi and Z. Xue, "IDSGAN: Generative Adversarial Networks for Attack Generation against Intrusion Detection," arXiv preprint arXiv:1809.02077 (2018).
[15] K. Yang, J. Liu, C. Zhang and Y. Fang, "Adversarial Examples Against the Deep Learning Based Network Intrusion Detection Systems," In MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM), pp. 559-564. IEEE, 2018.
[16] N. Papernot, P. McDaniel and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," arXiv preprint arXiv:1605.07277 (2016).
[17] P. Chen, H. Zhang, Y. Sharma, J. Yi and C. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," In Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, pp. 15-26. ACM, 2017.
[18] A. Kurakin, I. Goodfellow and S. Bengio, "Adversarial examples in the physical world," arXiv preprint arXiv:1607.02533 (2016).
[19] I. Goodfellow et al., "Generative adversarial nets," Advances in neural information processing systems (NIPS). 2014.
[20] A. Odena, C. Olah and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," arXiv preprint arXiv:1610.09585 (2016).
[21] M. Usama, M. Asim, S. Latif, J. Qadir and Ala-Al-Fuqaha, "Generative Adversarial Networks For Launching and Thwarting Adversarial Attacks on Network Intrusion Detection Systems," 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), Tangier, Morocco, 2019, pp. 78-83.
[22] O. Ibitoye, O. Shafiq and A. Matrawy, "Analyzing Adversarial Attacks Against Deep Learning for Intrusion Detection in IoT Networks," arXiv preprint arXiv:1905.05137 (2019).
[23] M. Pawlicki, M. Choras and R. Kozik, "Defending network intrusion detection systems against adversarial evasion attacks," Future Generation Computer Systems (2020).
[24] A. Odena, "Semi-supervised learning with generative adversarial networks," arXiv preprint arXiv:1606.01583 (2016).
[25] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, 2017, pp. 39-57.
[26] W. Brendel, J. Rauber and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," arXiv preprint arXiv:1712.04248 (2017).
[27] J. Su, D. V. Vargas and K. Sakurai, "One Pixel Attack for Fooling Deep Neural Networks," in IEEE Transactions on Evolutionary Computation, vol. 23, no. 5, pp. 828-841, Oct. 2019.
[28] "Keras: The Python Deep Learning Library." https://keras.io/
[29] J. Rauber, W. Brendel and M. Bethge, "Foolbox: a python toolbox to benchmark the robustness of machine learning models," arXiv preprint arXiv:1707.04131 (2017)