MANDA: On Adversarial Example Detection for Network Intrusion Detection System

Ning Wang, Yimin Chen, Yang Hu, Wenjing Lou, Y. Thomas Hou

Virginia Polytechnic Institute and State University, VA, USA

IEEE INFOCOM 2021 - IEEE Conference on Computer Communications | 978-1-6654-0325-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/INFOCOM42981.2021.9488874

***Abstract*—With the rapid advancement in machine learning (ML), ML-based Intrusion Detection Systems (IDSs) are widely deployed to protect networks from various attacks. Yet one of the biggest challenges is that ML-based IDSs suffer from adversarial example (AE) attacks. By applying small perturbations (e.g. slightly increasing packet inter-arrival time) to the intrusion traffic, an AE attack can flip the prediction of a well-trained IDS. We address this challenge by proposing MANDA, a MANifold and Decision boundary-based AE detection system. Through analyzing AE attacks, we notice that 1) an AE tends to be close to its original manifold (i.e., the cluster of samples in its original class) regardless which class it is misclassified into; and 2) AEs tend to be close to the decision boundary so as to minimize the perturbation scale. Based on the two observations, we design MANDA for accurate AE detection by exploiting inconsistency between manifold evaluation and IDS model inference and evaluating model uncertainty on small perturbations. We evaluate MANDA on NSL-KDD under three state-of-the-art AE attacks. Our experimental results show that MANDA achieves as high as 98.41% true-positive rate with 5% false-positive rate and can be applied to other problem spaces such as image recognition.**

I. Introduction

The increasing scale and complexity of modern networks and the tremendous amount of applications running on them render communication and networking systems highly vulnerable to various intrusion attacks. Intrusion detection system (IDS) plays a significant role in safeguarding networks from intrusion attacks [1]. There are mainly two types of IDS: signature-based detection [2] and anomaly-based detection [3]. Signature-based detection schemes work by extracting the traffic signature and comparing to those in a pre-built knowledge base. As a result, they are only effective in detecting known attacks but cannot detect attacks outside the knowledge base. Anomaly-based detection aims to detect deviations from an established norm traffic model. With the advancement in ML in recently year, ML techniques are increasingly used to train the "norm" model that represents the normal benign traffic and then to evaluate the credibility of incoming traffic. Considering intrusion attacks are ever evolving in these days, ML-based methods are showing much greater potential as they require only a little or no prior knowledge to work on emerging novel attacks.

ML technologies have seen great success in domains such as computer vision and natural language processing [4]–[6]. While applying to network intrusion detection, state-of-the-art IDSs usually implement advanced neural networks (e.g., LSTM) and learning schemes (e.g., meta-learning and active learning). An important security attack that is common to almost all machine learning models is the adversarial example (AE) attack [7], [8]. In such an attack, the adversary is able to craft a sample, often by applying small perturbations, and have a well-trained model output an arbitrary label other than its true label with a high probability. For IDS, an attacker can launch AE attacks to significantly increase false-positive rate and false-negative rate, rendering the IDS practically useless.

AE attacks have become more and more sophisticated that AE attacks on ML-based IDSs are becoming a real threat to network security. Lin et al. [9] leveraged a generative adversarial network (GAN) to transform original malicious traffic into adversarial traffic to fool the IDS. Wu et al. [10] employed deep reinforcement learning (DRL) to generate adversarial traffic flows to deceive the detection model automatically and adaptively. Rigaki et al. [11] utilized a GAN to adapt the Command and Control (C2) channel of malicious traffic to mimic the traffic of a legitimate application (e.g. the Facebook chat network traffic), and therefore evaded the IDS. Shu et al. [12] employed active learning and GAN to launch AE attacks on ML-based IDS, demonstrating the capability to compromise an IDS using only limited prior knowledge. The above attacks [9]–[13] confirm that AEs are inevitably turning into a huge threat to ML-based IDSs.

To defend against AE attacks, one can generally take two routes: 1) improving the robustness of an IDS model against adversarial perturbations, or 2) developing an auxiliary AE detector to reject suspicious inputs [14] before going into the IDS. Defense schemes in the first category [15], [16] usually need to customize the IDS models to every AE attack encountered. Considering many novel AE attacks are yet to come, we opt to design an effective AE detector as the defense, i.e., the second route.

In this paper, we propose MANDA, a MANifold and Decision boundary-based AE detection scheme for ML-based IDS. The benign or malicious traffic events usually reside in a low-dimensional manifold (i.e., a cluster) embedded in the ambient feature space. The goal of ML-based IDS is to learn a decision boundary that discriminates malicious network traffic from benign network traffic. To explain our intuitions behind MANDA clearly, we use an AE generated from a malicious network event for illustration. The AE fools the IDS model (i.e., evades the IDS) by traversing the decision boundary of IDS model. To preserve the malicious property of an intrusion traffic, the crafted AE should be inside or at least close to the malicious manifold. Therefore, although the IDS model classifies the AE as benign, a manifold detector is highly likely to discriminate it into the manifold of malicious samples. This motivates us to use such an inconsistency to detect AE. Also, considering that AEs are usually closer to the decision boundary of IDS model than normal samples, it is expected that when added small noise, the classification result of the AE is more likely to change than that of a clean sample. This motivates us to use such change of IDS classification results to detect AE.

The contributions of our paper are summarized as follows:

- We systematically investigate practical AE attacks and defenses of recent ML-based IDSs. To the best of our knowledge, we are the first to investigate AE attacks for IDS in problem space rather than in feature space, and also the first to propose an effective AE detection scheme to defend against such attacks.
- We propose `MANDA`, a novel MANifold and Decision boundary-based AE detection scheme for ML-based IDS. `MANDA` is designed by exploiting unique features we observe while trying to categorizing AE attacks from the viewpoint of machine learning model and data manifold. Based on our AE categorization, `MANDA` combines two building blocks (i.e., `Manifold` and `DB`) together to achieve effective AE detection regardless of which AE attack is used.
- Our experimental results show that `MANDA` achieves 98.41% true-positive rate (TPR) with a fixed 5% false-positive rate (FPR) under CW attack, the most powerful AE attack, and over 0.97 AUC-ROC under three frequently-used attacks (FGSM attack, BIM attack, and CW attack) on the NSL-KDD dataset. We also demonstrate that `MANDA` outperforms `Artifact` [17], a state-of-the-art solution on AE detector, on both IDS task and image classification task.

The remainder of the paper is organized as follows. Section II summarizes the related work. Section III introduces the system model and threat model. In Section IV, we elaborate the proposed AE detection scheme. We then present and compare the experimental results in Section V. Conclusion are drawn in Section VI.

II. Background and Related Work

In this section, we review the previous work most related to our paper including recent intrusion attacks on IDS and adversarial examples in deep learning. To the best of our knowledge, no prior work focuses on defense mechanisms against AE attacks on IDS.

A. Network Intrusion Attacks

Recently the information technology infrastructures including the Internet, telecommunication networks, computer systems, and embedded industrial processors, are subject to various network intrusion attacks [18]. A network probing attack searches for network vulnerabilities by scanning the connections (e.g., port scanning) of the network in order to launch further attacks. Another type of network intrusion attacks, the advanced persistent threat (APT) attacks [19], is powerful in a different way since the attack relies on coordinated human executions rather than running automated code. In an APT attack, continuous monitoring and interaction are conducted persistently to a target entity until the objectives are achieved. Different from APT attacks, a distributed denial of service (DDoS) attack tries to disrupt network operation by exhausting network resources but usually with no further goals. A recent prominent example of a DDoS attack is the Mirai botnet, which took down hundreds of websites including Twitter, Netflix, Reddit, and GitHub for several hours in October 2016. Today, Mirai mutations are generated daily and they can continue to proliferate and inflict real damage to networks [20].

The recent development in machine learning has enabled new and powerful ML-based IDSs. At the same time, the rapid progress in adversarial machine learning brought out a novel network intrusion attack, i.e. adversarial example attack, to screw up a ML-based IDS. Lin et al. [9] proposed IDSGAN which leverages a GAN to transform original malicious traffic into adversarial traffic instances so as to mislead the IDS to classify it as benign. Xu et al. [13] proposed a general method to automatically find evasive variants for a target classifier. Their method first uses genetic programming techniques to manipulate a malicious sample and then obtains its variant that preserves malicious behavior but is classified as benign by the classifier. They demonstrated its effectiveness in two popular PDF malware classifiers. Apruzzese et al. [21] studied realistic adversarial example attacks performed on IDS with a focus on identifying botnet traffic by ML classifiers. Their results highlight the effectiveness of adversarial examples on all botnet detection classifiers they explored. Wu et al. [10] employed deep reinforcement learning (DRL) to generate adversarial traffic flows to deceive a target detection model automatically. In this attack, the reinforcement learning agent updates the adversarial samples based on the feedback from the target model, which is able to adapt to the change of the temporal and spatial features of the traffic flows. Rigaki et al. [11] utilized a GAN to modify the Command and Control (C2) channel of malicious traffic so as to mimic that of a legitimate application (e.g. Facebook traffic) and evade detection. Shu et al. [12] employed active learning and GAN to launch the adversarial example attack on a ML-based IDS and showed great capability to attack an IDS using only limited prior knowledge. In sum, it is clear that AE attacks are posing real threat to today's ML-based IDS considering the cost is ultra low and AE attacks themselves are constantly evolving.

B. Adversarial Example

Adversarial example (AE) has become one of the most important research topic of machine learning in the past few years. It was first proposed in [7] that the classification result of a machine learning model on an arbitrary input sample could change dramatically by just applying intentionally crafted imperceptible perturbations. Such a perturbed sample is called an AE. Research on AE involves two main directions: how to generate AEs (i.e., AE attacks) and how to deal with AEs (i.e., AE defenses).

AE attacks. Multiple ways to modify a sample into an AE have been proposed which lead to multiple AE attacks. Many AE generation methods compute the gradient of the model's loss with respect to the input. To lead to a misclassification, the fast gradient sign method (FGSM) moves the current input image along the direction that maximizes the loss. Basic iterative method (BIM) extends FGSM to an iterative scheme by applying multiple times of gradient sign with small steps. The Jacobian-based saliency map attack (JSMA) seeks the top features that contribute to misclassification when applying a fixed distortion and only perturbs the selected pixels until a misclassification is arrived. Carlini and Wagner proposed optimization-based attacks to find a successful AE with the smallest distortion. Among all known AE attacks, FGSM, JSMA, and CW are the most referenced ones.

AE detection. Most defenses for AE attacks are specifically designed for image input in computer vision research. For example, image processing techniques such as reducing color depth [22], [23], reducing image size [24], [25], increasing resolution [26], and rotation or shifting [27] have been developed to detect AEs. Apparently, those methods are designed for images, they are either not applicable or not effective when applying to network traffic-based IDS.

Detection schemes based on statistical testing are not limited to image input and thus can be applied for IDS. Generally, the hypothesis of these AE detectors is that statistical characteristics of AEs and clean data are different and thus distinguishable. Grosse et al. [28] applied the kernel-based two-sample test to distinguish AEs from clean data. Song et al. [29] leveraged generative models to decide whether an input sample was drawn from the same distribution of clean data. Zheng et al. [30] used a Gaussian Mixture Model (GMM) to approximate the hidden layer distribution so as to reject samples with hidden states lying in the low density regions of the distribution. Feinman et al. [17] employed kernel-based density estimation to detect AEs. The drawback of the scheme in [17], [28] is that it requires sufficient AEs to build the detector, similar to defense schemes based on adversarial training [31], [32].

To the best of our knowledge, we are the first to design AE detection schemes for ML-based IDS. We compare our scheme to one of the state-of-the-art statistical testing schemes [17] in IDS and also show its applicability for tasks with image input.

III. System Model and Threat Model

This paper proposes a novel AE detector for ML-based IDS. We first describe our system model and threat model in this section.

A. Notations

First let us clarify two types of 'detection' tasks in our paper. The first is 'intrusion detection' which is our target application scenario. In this paper, we consider an intrusion detection system, in short IDS, that relies on machine learning techniques to detect abnormal/malicious network events. In the remaining part, we refer to the classification model of an IDS (shown in Fig. 1) as the `IDS model`, of which the purpose is to decide *whether an input sample of network events is an intrusion or not.* The second is 'AE detection' which is our research goal. An IDS model is subject to AE attacks. The proposed AE detector, i.e. `MANDA`, is placed in front of the intrusion detection module so as to detect and reject adversarial examples before they are fed into the IDS model. So the purpose of the AE detector is to decide *whether an input sample is an AE or not.* With the clarification, positive and negative samples of the two, i.e., IDS model and `MANDA`, can be defined as follows.

For IDS model

- The input to the IDS is a sequence of network packets. Let us call it a network event. An `malicious input` then refers to a network event that is generated by a malicious attack, such as a DDoS attack or a Botnet traffic. Ideally the IDS will classify a malicious input as positive, otherwise a false negative will occur which means a malicious input has evaded the detection. The synonyms for "malicious input" include "malicious data", "malicious traffic", or simply "intrusion".
- We use `benign traffic` to denote network events generated from normal network applications. Ideally the IDS will classify all benign traffic as negative samples. Sometimes 'benign traffic' are also referred to as 'benign data' , 'benign input', or 'normal traffic'.

For `MANDA`

- The input to the `MANDA` system is also sequences of network packets, i.e. network events. We use `adversarial examples` (AEs) to refer to the inputs that have been intentionally crafted to fool the IDS model, i.e. either to evade the IDS detection or to create false positives. The goal of the `MANDA` is to classify those AEs correctly. We also use 'adversarial input' or 'adversarial sample' to refer to an AE.
- We use `clean example` to refer to the network traffic instances without adversarial perturbations. The `MANDA` system is expected to classify a clean example as negative. A clean example can be either a malicious traffic instance or a benign traffic instance. We also use 'clean input' or 'clean sample' to refer to it.

One thing to emphasize is that an AE can be an input crafted from a malicious input and classified as 'benign' or one crafted from a benign input and classified as 'malicious'. For illustration purpose, we always stick to the former case across the paper. In experiments, we explore both cases.

B. System Model

A typical architecture of a ML-based IDS is shown in Fig. 1. Usually, IDS is a passive infrastructure which rarely interferes with the network traffic under monitoring. An IDS sniffs the internal interface of the firewall in a read-only mode and

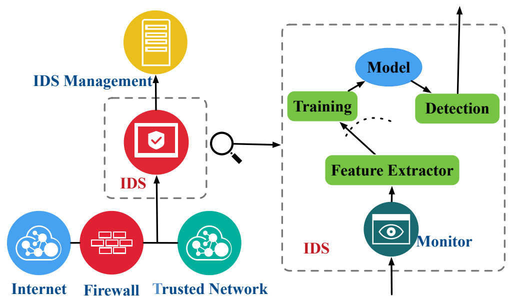

Fig. 1: System model of ML-based IDS.

sends alerts to an IDS management server via a read-and-write network interface [33], [34]. As Fig. 1 shows, a ML-based IDS is composed of the following modules [3]:

- `Network Traffic Monitor` keeps tracking the ongoing network traffic of a communication and networking system.
- `Feature Extractor` processes the raw traffic data as feature vectors in a pre-defined form.
- `Training Phase.` In the training phase, an ML model is trained with both benign and malicious traffic instances. We refer to the ML model as IDS model.
- `Detection Phase.` In the detection phase, processed runtime traffic instances are fed into the learned model. An alert will be generated if an input instance is classified as positive by IDS model.

C. Threat Model

In this paper, we focus on AE attacks in which an attacker aims to mislead the IDS model by slightly modifying its traffic flow, e.g., enlarging or shortening the length of payload slightly as shown in Fig. 2. The goal of the attacker is to let the IDS model to either misclassify a malicious traffic instance as benign (i.e., increase FN) or misclassify a benign one as malicious (i.e., increase FP). In either case, successful AE attacks may render the IDS model less effective or practically useless. Depending on the prior knowledge known to the attacker, there are three types of attacks to ML-based systems: white-box attack, gray-box attack, and black-box attack.

- `An white-box adversarial attacker` knows both the architecture and weights of the IDS model.
- `An gray-box adversarial attacker` knows the IDS model architecture but not the weights. She is able to query the model while trying to reduce the number of queries to avoid being suspicious.
- `An black-box adversarial attacker` has no information about the architecture and weights of the IDS model. She is able to query the model while trying to reduce the number of queries to avoid being suspicious.

In this paper, we consider white-box attack to the IDS system, which is the most powerful one among the three types from the attacker's perspective. This will allow attackers to craft adversarial network instances in the most effective and stealthy manner in order to defeat the IDS system. However, we assume that the attacker has no knowledge of our proposed

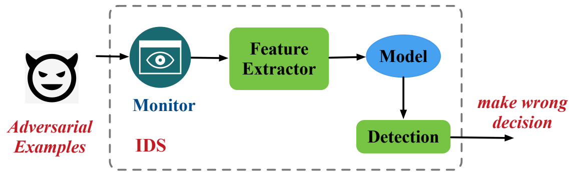

Fig. 2: Attack model of adversarial example generating attacks.

AE detection method. The general AE generation process can be summarized as follows. Let $\mathcal{F}$ be a m-class classifier with model parameter θ. The model maps the input ($x \in \mathbb{R}^n$) to the output ($y \in \mathbb{R}^m$), i.e., $y = \mathcal{F}(\theta, x)$. Note that $y[i], i = 1, \cdot, m$ denotes the probability that x belongs to i-th class, and $y[1]+...+y[m] = 1$. The predicted label of x, $c(x)$, is the class with the largest $y[i]$, i.e., $c(x) = \arg\max y$. The objective of AE generation is to search for $x' = x + \eta$ (η denotes a small perturbation) such that $c(x') \neq c(x)$. Assume $J(\theta, x)$ is the loss function of $\mathcal{F}(\theta)$ on an input x. Despite that there are quite a few AE attacks in the literature, we choose four most representative and effective attacks to address in this paper. In what follows we provide a brief review of the four attacks and highlight the techniques used in each of them.

1) `FGSM`: Goodfellow et al. [8] proposed a fast gradient sign method (FGSM) to generate AEs. Specifically, an attacker computes the gradient of y with respect to x. It then moves the current x along the direction that maximizes $J(\theta, x, y)$. The generated AE can be written as:
$$x' = x + \epsilon \text{sign} \nabla_x J(\theta, x).$$
2) `BIM`: Kurakin et al. [35] proposed a basic iterative method (BIM) which is an extension of FGSM. This attack applies fast gradient sign multiple times with small steps and clips the pixel value of intermediate results of each step to ensure that the generated AE is in the ϵ-neighborhood of the original input. ϵ is a global parameter to bound the distance between x and its AE.
$$x_0^{'} = x$$
$$x_i^{'} = \text{clip}_{x,\epsilon}(x_{i-1}^{'} + \alpha \text{sign} \nabla_x J(\theta, x_{i-1}^{'})), i \geq 1.$$
3) `JSMA`: In [36], Papernot et al. proposed jacobian-based saliency map attack (JSMA), which applies iterative computation to seek features which contribute more for misclassification in each step. For an input x, the prediction confidence on j-th class is denoted by $f_j(x)$. To generated an AE for a target class t, the applied perturbation needs to satisfy two requirements simultaneously: a) $f_t(x)$ increases and b) $f_j(x), \forall j \neq t$ decreases. The adversarial saliency map (for i-th feature) is defined as:
$$S(x,t)[i] = \begin{cases} 0, \text{if } \frac{\partial f_t(x)}{\partial x_i} < 0 \text{ or } \sum_{j \neq t} \frac{\partial f_j(x)}{\partial x_i} > 0 \\ \frac{\partial f_t(x)}{\partial x_i} | \sum_{j \neq t} \frac{\partial f_j(x)}{\partial x_i} > 0 |, \text{otherwise.} \end{cases}$$
4) `CW attack` In [37], Carlini and Wagner introduced optimization-based attacks for generating AEs. Three distance metrics – L_0, L_2, and L_∞ distance – are used to evaluate the distortion of an AE from its original input.

The AE generation process is formulated as:

$$\min \|x' - x\| + c \cdot \max\{\max\{f_i(x') : i \neq t\} - f_t(x'), -\kappa\}$$
$$\text{subject to } x' \in [0,1]^n$$

In most cases, CW attack outperforms the other three methods in terms of effectiveness and distortion.

IV. The MANDA System

In this section, we present the design of MANDA, the proposed AE detector for ML-based IDS, and explain the rationale behind each design choice. The valid input to an IDS system is real network traffic flows in the problem-space. Therefore, the generated AE should also lie in the same problem-space of IDS. We adapt existing feature-space AE generation algorithms to problem-space algorithms in order to generate AEs that can map back to valid real network events. The key insight for detecting AEs is to identify the discrepancy between true benign samples and AEs. Such an intuition motivates us to investigate AE's position to the decision-boundary of the IDS model and its position in the traffic manifolds formed by training samples.

A. Problem-Space AE Attack for IDS

In this section, we demonstrate how we generate AEs for IDS in problem-space. The problem space of an IDS is all possible traffic instances in the form of sequences of network packets while its feature space is all possible feature vectors in the form of numerical entry representing packet length, packet inter-arrival time, etc. Prior AE generation algorithms [35]–[37] are for image inputs where the problem space and feature space are the same, i.e. a vector of pixels. The problem-space AE attacks specifically on network flow-based IDS have not been investigated yet. Our work is to fill this gap.

It takes two steps to generate an AE in problem-space. First, we generate a feature-space AE x' from a clean input x. Second, we design a mapping function to project x' back to problem-space and obtain the ultimate problem-space AE z'. The corresponding representation of x in problem-space is denoted by z.

Traditional inverse feature-mapping techniques are not adequate to project a feature-space instance to problem-space because the mapping itself is neither invertible nor differentiable in our problem context. Our approach is to force the mapping to be differentiable by nullifying the perturbations on non-differentiable features. Specifically, x', the crafted feature-space AE, is composed of two parts: $\{x^{'}_{diff}, x^{'}_{non-diff}\}$. We force that $x^{'}_{non-diff} = x_{non-diff}$. From this we go straight to letting $z^{'}_{non-diff} = z_{non-diff}$. After getting rid of $x^{'}_{non-diff}$, we backpropagate the gradient of $\mathcal{F}(x_{diff})$ from x to z such that z's change in problem-space follows negative gradient. In IDS, non-differentiable features are categorical features like 'protocol type', 'service type', and so on. We also conjecture that nullification of such non-differentiable features helps to maintain intrinsic properties of the clean data. Obviously, an AE attack still fails if its AEs fool the IDS model but do not maintain intrinsic properties of the clean data.

Compared to feature-space AE generation, we need to include the following restrictions for problem-space AE generation.

$$\|\boldsymbol{x'} - \boldsymbol{x}\|_2 \leq \epsilon,$$
$$\|\boldsymbol{x'}[\boldsymbol{i}] - \boldsymbol{x}[\boldsymbol{i}]\| \leq p * R_i, \text{ if } i \in S_{diff},$$
$$\boldsymbol{x'}[\boldsymbol{i}] = \boldsymbol{x}[\boldsymbol{i}], \text{ if } i \in S_{non-diff},$$

where $\boldsymbol{x'}$ is an AE generated from $\boldsymbol{x}$. $\boldsymbol{x}[i]$ denotes $\boldsymbol{x}$'s i-th feature and R_i denotes its range. ϵ denotes the maximum perturbation applied on $\boldsymbol{x'}$. S_{diff} corresponds to the set of differentiable features and p the maximum change ratio on S_{diff}. Only the differentiable features are eligible to modify.

B. Properties of AE

First let's talk about `manifold learning` for better understanding of our AE categorization scheme. The assumption of manifold learning is that input data reside on or close to a low-dimensional manifold embedded in the ambient space [38]. For example, a plane is the manifold revealed by a group of three-dimensional data if these data points lie in a plane (a flat, 2-dimensional surface). Manifold learning refers to the process of automatically learning the geometric and topological properties of a given manifold [39]. Most manifold learning methods focus on data representation as in [40]–[42]. Let $\mathcal{M}$ be a manifold model. Formally, we refer to the inference on an input x as 'manifold evaluation', denoted by $\mathcal{M}(x)$. Similar to machine learning, the output of manifold evaluation is also a vector showing the probabilities of x locating in each sub-manifold (i.e., each class).

Let's look back to AE for IDS again. The IDS model, $\mathcal{F}$, maps an input sample x to a confidence vector $y \in \{p_0, p_1\}$. The final predicted class is $c(x) = \arg\max \mathcal{F}(x)$. $c(x) = 1$ indicates that x is classified as malicious while $c(x) = 0$ indicates that x is classified as benign. Assume $c(x) = 1$ (the true label of x is also 1) and $x' = x + \eta$ is an AE generated from x. The goal of AE attack is $c(x') = 0$. *We assume that x' needs to keep the essential property of its true class, i.e., 'malicious'.* This translates to that although x' is closer to 'benign' class in the eye of the IDS model, it does not totally comply with the property of 'benign' class. Otherwise, x' is not an AE rather a new instance of 'benign' class.

Assuming that *AE needs to keep the essential property of its true class* (which is confirmed by our experimental results), successful AEs can be categorized into two cases. Again, we use an AE given by $c(x) = 1$ and $c(x') = 0$ for illustration purpose, i.e., x is a malicious sample and x' is recognized as a benign sample. Notice that we also consider AEs with $c(x) = 0$ and $c(x') = 1$ in our experiments.

- `Case A`: *x' is close to the manifold of the 'malicious' class but far from the manifold of the 'benign' class.* This occurs frequently when malicious and benign manifold are fully separable from each other.
- `Case B`: *x' is close to both manifolds.* This happens when both manifolds are close to each other, or even overlapping (i.e., cannot separate from each other perfectly).

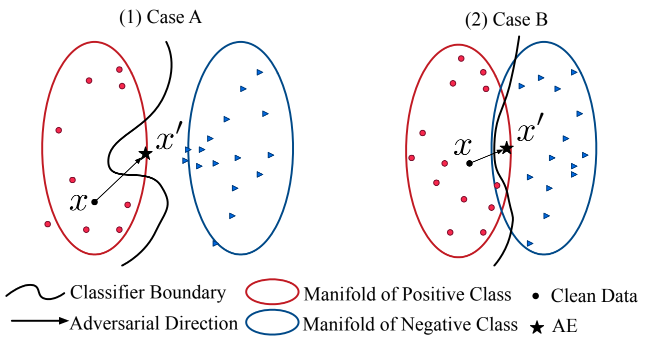

Fig. 3: The illustration of AEs in the 2-D view.

Fig. 3 illustrates `Case A` and `Case B` for two-dimensional data. In practice, the dimension of a manifold depends on the dimension and distribution of input samples. It is noted that the two cases (i.e., `Case A` and `Case B`) are not non-coexistent. They may exist for the same IDS model at different segments of the decision boundary.

Note that if x' is far away from the manifold of 'malicious' class and close to that of 'benign' class, x' is not a successful AE because x' breaks the above assumption of *AE needs to keep the essential property of its true class*. The same happens to the case that x' is far away from both manifolds.

C. MANDA

Here we introduce our AE detection scheme, MANDA, which is based on the above AE categorization. Assume that the IDS model has high classification rate on clean data since it makes much less sense to discuss detecting AEs already misclassified by the IDS model. Obviously, a clean input needs to traverse the decision boundary of the IDS model to be an AE. As mentioned in Section I, MANDA consists of two components denoted by Manifold and DB, respectively. Manifold combines both x's classification and manifold evaluation results together to detect AEs. If the two outputs of x are inconsistent, x is very likely to be an AE. DB explores whether an input x is near the decision boundary of the IDS model to detect AE. Specifically, if we add small Gaussian noise to x and the corresponding y (thus $c(x)$) changes frequently, x is very likely to be an AE. We want to emphasize that both Manifold and DB can be used as stand-alone AE detection scheme while MANDA combines them together for better performance.

1) Manifold: For `Case A` in Fig. 3, an AE lies in or near the manifold of 'malicious' class but far from 'benign' class. Meanwhile, the output label from the IDS model on the AE is 'benign' class. Therefore, results of the IDS model and manifold evaluation on the same input are inconsistent. On the contrary, results for a clean input tends to be consistent. Intuitively, we can use the inconsistency between the IDS model and manifold evaluation as a criterion for AE detection.

In order to capture the data manifold for positive and negative class, we employ a transductive learning model proposed by Zhou et al. in [43]. The learning method explores the intrinsic structure revealed collectively by a group of labeled

Algorithm 1 Score Computed for Criterion 1 & 2

```
Input: input $x \in \mathbb{R}^n$, IDS model $\mathcal{F}(\theta)$, learned manifold $\mathcal{M}$
Output: $score_1, score_2$
1: $p \leftarrow \mathcal{M}(x)$  # Confidence vector of manifold evaluation
2: $q \leftarrow \mathcal{F}(\theta, x)$  # Classifier output
3: $score_1 \leftarrow \|p\| + \|q\| - \|p+q\|$        # Criterion 1
4: for $i = 0$ to $N$ do
5:     $x_i = x + \mathcal{N}(0, \sigma^2)$
6:     $p_i \leftarrow \mathcal{F}(\theta, x_i)$
7: end for
8: $score_2 \leftarrow \frac{1}{N}\sum_{i=1}^{N}\|p_i\| - \frac{1}{N}\left\|\sum_{i=1}^{N} p_i\right\|$  # Criterion 2
9: return $score_1, score_2$
```

and unlabeled data points. It guarantees both local consistency and global consistency of known data points. This means that (1) nearby points are likely to have the same label; and (2) points on the same structure (typically referred to as a manifold) are likely to have the same label. The learned manifold evaluation model is sufficiently smooth with regards to intrinsic data structure. We use this learning method to obtain the manifold for each class in this paper.

Detection Criterion 1 *We conclude an input as an AE if an inconsistency occurs between manifold evaluation and IDS model.*

For implementation, we first compute $score_1$ by combining results from manifold evaluation and the IDS model as shown in Algorithm 1. Next, we compare $score_1$ to an optimal threshold to decide whether an input sample is an AE or not. See Algorithm 2.

2) DB: Different from `Case A`, the two manifolds of 'malicious' and 'benign' class are not fully separable in `Case B`. Here we have the following proposition:

PROPOSITION 1: *In IDS, if*

(i) *the two manifolds of 'malicious' and 'benign' classes are not fully separable but most instances are still distinguishable;*
(ii) *IDS classifier $\mathcal{F}$ is with optimized accuracy;*
(iii) *x is a clean malicious input and x' is x's corresponding AE, i.e., $c(x)=1$, $x'=x+\eta$, $c(x')=0$; and*
(iv) *x' keeps the essential property of 'malicious' class.*

Then, x' is very close to the decision boundary of $\mathcal{F}$ with high probability.

For `Case B` in Fig. 3, an AE should be very sensitive to small perturbations. Based on Proposition 1, we use whether an input is close to the decision boundary as a secondary criterion for AE detection. Note that this criterion can falsely conclude a clean input close to boundary as an AE. Due to the curse of dimensionality very few clean inputs that are correctly classified are close to the boundary [44]. Our experimental results also verify such a hypothesis.

DB needs to evaluate whether an input is a near-boundary example in high-dimensional space. We achieve this goal by evaluating the uncertainty of IDS model output when an input is applied with small additive perturbation. For a near-boundary example, such a small perturbation may cause it to

Algorithm 2 `Manifold`

Input: input $x \in \mathbb{R}^n$, model parameters θ, threshold τ
Output: $isAdversarial \in \{False, True\}$

```
1: $score_1, score_2 \leftarrow$ **Score-Compute**$(x, \theta, \tau)$ # Algorithm 1
2: if $(score_1 > \tau_1)$ then
3:     $isAdversarial \leftarrow True$
4: end if
5: return $isAdversarial$
```

traverse the decision boundary. Consequently, the predictions of IDS model become very unstable when an input is applied with perturbations. Conversely, a small perturbation on an input away from the boundary will hardly lead to such a change. We compute model uncertainty on an input with additive Gaussian perturbation $\mathcal{N}(0, \sigma^2)$ in Algorithm 1. For an input x, the uncertainty of output from IDS model is evaluated as the variance of the confidence vector $\mathcal{F}(x_i)$ of $x_i = x + \mathcal{N}(0, \sigma^2), (i \in \mathbb{N}, i \leq N)$:

$$score_2 = \frac{1}{N}\sum_{i=1}^{N}\|\mathcal{F}(\theta, x_i)\| - \frac{1}{N}\left\|\sum_{i=1}^{N}\mathcal{F}(\theta, x_i)\right\|.$$

Similar to `Manifold`, `DB` uses an optimal threshold for $score_2$ to decide whether x is an AE or not. Due to space limit, we do not include the pseudocode of `DB` here.

Detection Criterion 2 *We conclude an input with high model uncertainty on small perturbations as an AE.*

3) `MANDA` (`Manifold+DB`): To combine `Manifold` and `DB` together for AE detection, `MANDA` first obtains $[score_1, score_2]$ of each x in training set from them. Denote x's label in training set as $label(x)$. Next, `MANDA` transforms each $[x, label(x)]$ into $[score_1, score_2, label(x)]$ and obtains a new training set. Finally, `MANDA` trains a logistic regression model on the new dataset and uses it for AE detection. Algorithm 3 details the whole process of `MANDA`.

V. Experimental Results

A. Datasets

1) NSL-KDD: We use the internet traffic dataset, NSL-KDD [45] (also used in AE attacks in IDS [9], but [9] dose not consider problem-space validity), for our evaluation. In NSL-KDD, each sample contains four groups of entries including Intrinsic Characteristics, Content Characteristics, Time-based Characteristics, and Host-based Characteristics. There are four categories of intrusion: DoS, Probing, Remote-to-Local (R2L), and User-to-Root (U2R) of which each contains more attack sub-categories. There are 24 sub-categories of attacks in the training set and 38 sub-categories of attacks are in test set (i.e., 14 sub-categories of attacks are unseen in the training set). There are 125,973 training records and 22,544 testing records. In our experiments, we only show the evaluations on an IDS model for discriminating DoS attacks from normal traffic since the results for the other three attacks are similar. The total number of entries for each record is 41 (in problem-space) which are further processed into 121 numerical features as an input-space (feature-space) vector.

Algorithm 3 `MANDA`

Input: θ, $x_{test} \in \mathbb{R}^n$, training data $\mathbf{X}, \mathbf{Y_{adv}}$
Output: $isAdversarial \in \{False, True\}$

```
1: if training then
2:     $\mathbf{S_1}, \mathbf{S_2} \leftarrow$ **Score-Compute**$(\mathbf{X}, \theta, \tau)$ # Algorithm 1
3:     $model \leftarrow$ LogistcRegression$(\mathbf{S_1}, \mathbf{S_2}, \mathbf{Y_{adv}})$
4: else
5:     $score_1, score_2 \leftarrow$ **Score-Compute**$(x_{test}, \theta, \tau)$
6:     $isAdversarial \leftarrow model(score_1, score_2)$
7: end if
8: return $isAdversarial$
```

2) MNIST: We also evaluate our approach on an image dataset, MNIST [46], to demonstrate its applicability. The images in MNIST are handwritten digits from 0 to 9. The corresponding digit of an image is used as its label. Each class has 6,000 training samples and 1,000 test samples. Therefore, the whole MNIST dataset has 60,000 training samples and 10,000 test samples. All the images have the same size of 28×28 and are in grey-level.

B. Experiment Settings

We implemented the problem-space attacks and `MANDA` in TensorFlow. We ran all the experiments on a server equipped with an Intel Core i7-8700K CPU 3.70GHz$\times$12, a GeForce RTX 2080 Ti GPU, and Ubuntu 18.04.3 LTS. The IDS model is a muti-layer perceptron (MLP) composed of one input layer, one hidden layer with 50 neurons and one output layer. For completeness, we also implemented other models for IDS including Logistic Regression (LGR), K-Nearest Neighbors (KNN), Naive Bayes classifier for multivariate Bernoulli (BNB), Decision Tree Classifier (DTC) and Support Vector Machine (SVM) from scikit-learn library [47]. We implement four AE attacks including `FGSM`, `BIM`, `CW` (the L_2-norm version) and `JSMA` (cf. Section III-C) and adapt the first three to problem-space of IDS. In each experiment, we generate AEs on the test samples that are correctly classified by the IDS model. Note here that we do not generate AEs for misclassified test samples. Next, we combine the successful AEs and the same number of clean data points (randomly selected) together as a mixed dataset, on which we run all detection algorithms. The benchmark for comparison is `Artifact` [17], the same as in [14], [44]. `Artifact` is proposed by Feinman et al. in [17] and becomes one of the state-of-the-art AE detection scheme. Different from `MANDA`, `Artifact` uses kernel density estimation (KDE) and Bayesian neural network uncertainty as two criteria to detect AEs.

On MNIST dataset, we use a convolutional neural network (CNN) rather than the above MLP as the target model for AE attacks. The CNN model comprises 4 convolutional layers with ReLU activation, followed by 2 fully-connected layers.

C. Evaluation Metrics

We compute True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) of AE detection as evaluation metrics defined as follows.

Perturbation restriction (%)	FGSM Acc (%)	FGSM L_2	BIM Acc (%)	BIM L_2	CW Acc (%)	CW L_2
0	90.64	0	90.64	0	90.64	0
1.0	84.48	1.40	83.84	1.54	77.02	1.08
2.5	71.10	1.51	64.05	1.58	52.61	1.45
5.0	64.27	1.58	59.48	1.58	42.68	1.58
7.5	60.09	1.67	55.95	1.62	37.47	1.68
10.0	56.63	1.79	52.79	1.67	34.51	1.67
None	2.42	2.57	0.08	1.53	0.00	0.96

[a]Perturbation is only applied to a subset of features.

TABLE I: IDS model accuracy under AE attacks with different perturbations

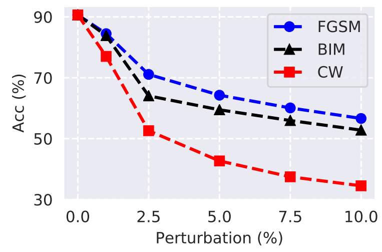

Fig. 4: IDS model accuracy from TABLE I

- TP: a test sample is an AE and detected as an AE.
- FP: a test sample is a clean sample but detected as an AE.
- TN: a test sample is a clean sample and detected as clean.
- FN: a test sample is an AE but detected as clean.

Then we have TPR $= \frac{TP}{TP+FN}$, FPR $= \frac{FP}{TN+FP}$. The receiver operating characteristics (ROC) curve is created by plotting the TPR against the FPR at various threshold settings. The AUC-ROC score is defined as the area under the ROC curve, and we use AUC to denote it in the following.

D. Results of IDS

1) AE Attacks in Problem-Space: We show the classification accuracy of the IDS model under `FGSM`, `BIM` and `CW` attack in Table I. We draw two main conclusions from the experimental results. First, the larger perturbation AE attacks use (i.e., p), the more powerful AE attacks are, and hence the less accurate the targeted IDS model becomes. Recall that p is the maximum change ratio on each feature of the modifiable feature set S_{diff} (in Section IV-A). Table I shows that model accuracy drops from 90.64% with $p = 0$ to 34.51% with $p = 10\%$ under `CW` attack (the strongest attack investigated). Second, the success attack rate of feature-space attack (i.e., the last row in Table I) is higher than that of problem-space attack (i.e., any row other than the last in Table I) because the latter faces more restrictions in generating AEs (See Section IV-A). Yet problem-space AE attacks can still result in very low accuracy of targeted IDS model with larger p. In what follows, we stick to $p = 5\%$ and explore the detection performance of our proposed approach under AE attacks.

It is known that AEs generated from one target model can transfer to other models. Table II shows the results of our problem-space AEs generated from MLP model and then applied to models including LGR, KNN, BNB, SVM and DTC. The significantly decreased accuracy confirms that our problem-space AE generation scheme maintains the capability of AE to transfer to different models.

TABLE II: IDS model accuracy under AE attacks

Models	Acc (%)	Acc (%) after attack FGSM	BIM	CW	Overall
LGR	89	15	70	52	45
KNN	91	33	38	51	40
BNB	87	84	86	26	65
SVM	89	22	26	25	24
DTC	84	79	79	73	77

2) Detection Performance: Here we show the results of our AE detection schemes including `Manifold`, `DB`, and `MANDA`.

First, we show ROC curves of detecting `FGSM`, `BIM`, and `CW` attacks in Fig. 5. We can see that `Manifold` and `MANDA` achieve similar ROC results which outperform both `DB` and `Artifact`. Such results inspire us that most AEs can be successfully detected by the inconsistency between IDS model and manifold model. We further compute the AUC and TPR (with a fixed FPR) in Table III to better show the detection performance. From the table, we can see that the best AUC score under `FGSM` attack is achieved by `Manifold`. For `BIM` attack and `CW` attack, `MANDA` outperforms the other detection methods with 0.9726 and 0.9851 AUC, respectively. We also show TPR under 5% FPR and 15% FPR in Table III. Our `Manifold` achieves the best TPR under `FGSM` attack with either 5% FPR or 15% FPR, while `MANDA` outperforms the other methods under `CW` attack with either 5% FPR or 15% FPR. For the `BIM` attack, `Manifold` achieves the highest TPR with 5% FPR, while `MANDA` works best with 15% FPR.

In sum, `Manifold` and `MANDA` achieve excellent AE detection performance for IDS models: they achieve both high AUC score and TPR. As pointed out in Section IV-C, our `DB` detection method is to detect those AEs which are not close to either manifold and `Manifold` thus fails to detect. Therefore, `MANDA` is able to detect more AEs while at the risk of more false positive samples. `Manifold`, `DB`, and `MANDA` are able to detect an AE in around 0.26 millisecond, rendering them great candidates for fast online detection.

E. Results on MNIST

We also apply `MANDA` to the MNIST dataset to evaluate their performance. The results below imply that the proposed approach can be used for other application scenario as well. When dealing with image input, we add a feature extractor to convert an low-level image input (i.e., a pixel vector) into a high-level feature vector for dimensionality reduction. The details are as follows.

1) Feature Extraction: The feature extractor is a convolutional auto-encoder comprising three convolutional layers with $32, 64, 64$ filters as the encoder and three convolutional layers with $64, 64, 32$ filters as the decoder. We first train the auto-encoder model and then include only the trained encoder into the main image classification model for MNIST. Comparing with the original input size of 28×28, the extracted high-level features are in a smaller size of 64×1.

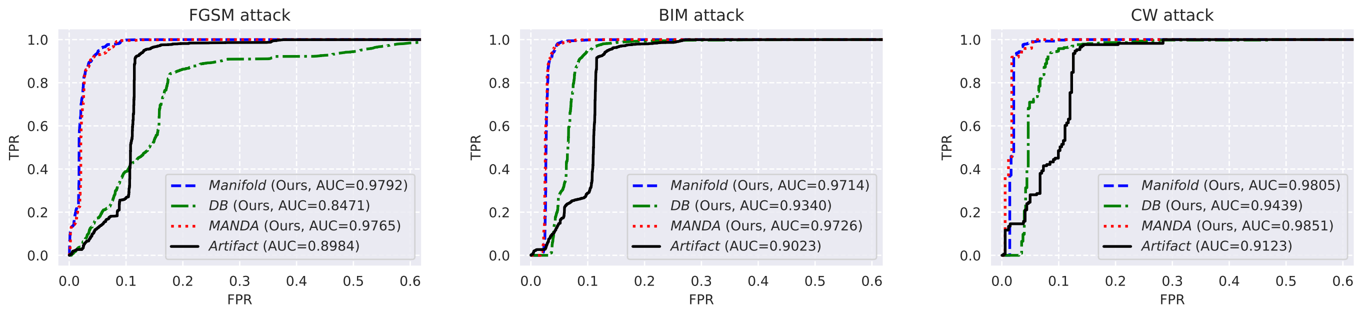

Fig. 5: ROC curve of AE detection methods under FGSM, BIM and CW attacks on NSL-KDD.

TABLE III: AE Detection Performance on IDS

Detection Method	FGSM			BIM			CW		
	TPR(%)		AUC-ROC	TPR(%)		AUC-ROC	TPR(%)		AUC-ROC
	FPR=5%	FPR=15%		FPR=5%	FPR=15%		FPR=5%	FPR=15%	
Manifold (Ours)	**94.04**	**100.00**	**0.9792**	**98.41**	99.98	0.9714	**98.38**	**100.00**	0.9805
DB (Ours)	17.27	53.57	0.8471	27.91	98.62	0.9340	71.00	97.91	0.9439
MANDA (Ours)	92.88	99.89	0.9765	98.04	**100.00**	**0.9726**	95.93	**100.00**	**0.9851**
Artifact [17]	12.60	96.63	0.8984	14.78	96.31	0.9023	28.07	97.66	0.9123

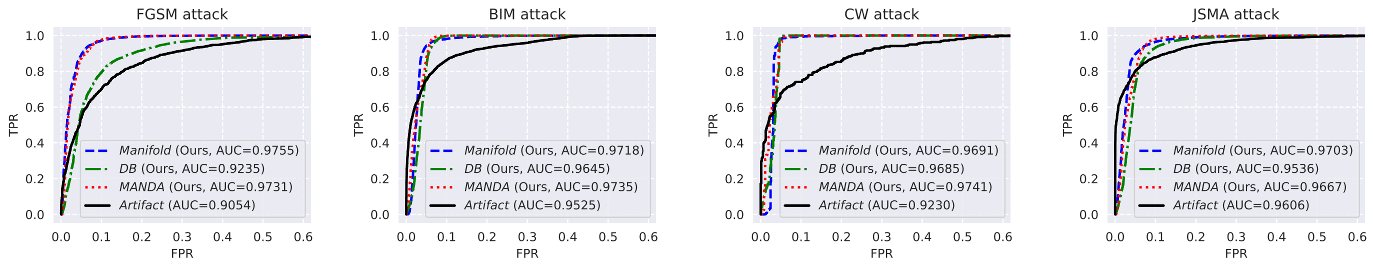

Fig. 6: ROC curve of AE detection methods under FGSM, BIM, CW, and JSMA attacks on the MNIST dataset.

2) Detection Performance: We show the results of AE detection of the main model under FGSM, BIM, JSMA and CW attacks in Fig. 6. With a fixed 5% FPR, MANDA achieves 0.89, 0.94, 0.90, and 0.99 TPR under FGSM, BIM, JSMA, and CW attack, respectively. Compared to Artifact, MANDA improves TPR under FGSM, BIM, JSMA, and CW attack by 0.36, 0.19, 0.10, and 0.31, respectively. Such results confirm that MANDA is effective not only for network intrusion detection but also for other application scenarios. We will continue to explore more application scenarios in the future work.

F. Discussion

For intrusion detection, an intrusion event is usually a malicious activity accessing a network component (e.g. gateway) in the form of a sequence of Internet packets. The problem-space adversarial examples generated in this paper are not directly the sequences of attacking packets yet. Our goal here is to understand the strategies and limits to which an attacker can reshape the attack traffic in order to evade the detection (i.e., changes to high-level features such as packet inter-arrival time, protocol type, etc.). This would provide important guidance for ultimate adversarial example generation at packet sequence level. We perceive that the module to generate packets that lead to the desired high-level features is a parallel research topic and hence is not the focus of our study here.

VI. Conclusion

In this paper, we examine three recent AE attacks against ML-based IDSs. The results confirm that the problem-space AE attacks are an effective disruption to the IDSs as it allows malicious events to escape with high probability. We identify common features of successful AEs, and based on which we design an effective and accurate AE detector, MANDA. The MANDA system takes on a novel design that exploits inconsistency between manifold evaluation and IDS model inference and evaluates model uncertainty on small perturbations to differentiate AEs from clean network traffic. Our evaluation of MANDA using the NSL-KDD dataset shows that MANDA outperforms the state-of-the-art statistical test model (i.e., Artifact) by achieving higher AUC score and higher true-positive rate with 5% false-positive rate. MANDA also performs well when evaluating using the MNIST dataset which implies that the detector may be applied to other domains, e.g. the computer vision area.

VII. Acknowledgment

This work was supported in part by the Office of Naval Research under grant N00014-19-1-2621, the National Science Foundation under grants CNS-1837519, and the Virginia Commonwealth Cyber Initiative (CCI).

References

[1] H.-J. Liao, C.-H. R. Lin, *et al.*, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[2] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Trans. on Information and System Security (TISSEC)*, vol. 3, no. 4, pp. 262–294, 2000.

[3] P. Garcia-Teodoro, J. Diaz-Verdejo, *et al.*, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *computers & security*, vol. 28, no. 1-2, pp. 18–28, 2009.

[4] M. Ren, A. Pokrovsky, *et al.*, "Sbnet: Sparse blocks network for fast inference," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition(CVPR)*, pp. 8711–8720, 2018.

[5] W. Xiong, L. Wu, *et al.*, "The microsoft 2017 conversational speech recognition system," in *IEEE int. conf. on acoustics, speech and signal processing (ICASSP)*, pp. 5934–5938, IEEE, 2018.

[6] M. Johnson, M. Schuster, *et al.*, "Google's multilingual neural machine translation system: Enabling zero-shot translation," *Trans. of the Association for Computational Linguistics*, vol. 5, pp. 339–351, 2017.

[7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[8] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[9] Z. Lin, Y. Shi, *et al.*, "Idsgan: Generative adversarial networks for attack generation against intrusion detection," *arXiv preprint arXiv:1809.02077*, 2018.

[10] D. Wu, B. Fang, J. Wang, Q. Liu, and X. Cui, "Evading machine learning botnet detection models via deep reinforcement learning," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2019.

[11] M. Rigaki and S. Garcia, "Bringing a gan to a knife-fight: Adapting malware communication to avoid detection," in *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 70–75, IEEE, 2018.

[12] D. Shu, N. O. Leslie, C. A. Kamhoua, and C. S. Tucker, "Generative adversarial attacks against intrusion detection systems using active learning," in *Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning*, pp. 1–6, 2020.

[13] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers," in *Proceedings of the 2016 network and distributed systems symposium*, vol. 10, 2016.

[14] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proc.10th ACM Workshop on Artificial Intelligence and Security*, pp. 3–14, 2017.

[15] N. Papernot, P. McDaniel, *et al.*, "Practical black-box attacks against deep learning systems using adversarial examples," *arXiv preprint arXiv:1602.02697*, vol. 1, no. 2, p. 3, 2016.

[16] F. Tramèr, A. Kurakin, *et al.*, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[17] R. Feinman, R. R. Curtin, *et al.*, "Detecting adversarial samples from artifacts," *arXiv preprint arXiv:1703.00410*, 2017.

[18] E. Bou-Harb, M. Debbabi, *et al.*, "Cyber scanning: a comprehensive survey," *Ieee communications surveys & tutorials*, vol. 16, no. 3, pp. 1496–1519, 2013.

[19] P. Hu, H. Li, *et al.*, "Dynamic defense strategy against advanced persistent threat with insiders," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 747–755, IEEE, 2015.

[20] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "Ddos in the iot: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[21] G. Apruzzese, M. Colajanni, and M. Marchetti, "Evaluating the effectiveness of adversarial attacks against botnet detectors," in *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*, pp. 1–8, IEEE, 2019.

[22] W. Xu, D. Evans, *et al.*, "Feature squeezing: Detecting adversarial examples in deep neural networks," *Proc. Network and Distributed System Security Symposium*, 2018.

[23] B. Liang, H. Li, *et al.*, "Detecting adversarial image examples in deep neural networks with adaptive noise reduction," *IEEE Trans. on Dependable and Secure Computing*, 2018.

[24] D. Hendrycks and K. Gimpel, "Early methods for detecting adversarial images," in *Int. Conf. on Learning Representations (ICLR)*, 2017.

[25] X. Li and F. Li, "Adversarial examples detection in deep networks with convolutional filter statistics," in *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*, pp. 5764–5772, 2017.

[26] A. Mustafa, S. H. Khan, *et al.*, "Image super-resolution as a defense against adversarial attacks," *IEEE Trans. on Image Processing*, vol. 29, pp. 1711–1724, 2019.

[27] S. Tian, G. Yang, *et al.*, "Detecting adversarial examples through image transformation," in *32ed AAAI Conf. on Artificial Intelligence*, 2018.

[28] K. Grosse, P. Manoharan, *et al.*, "On the (statistical) detection of adversarial examples," *arXiv preprint arXiv:1702.06280*, 2017.

[29] Y. Song, T. Kim, *et al.*, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," in *Int. Conf. on Learning Representations (ICLR)*, 2018.

[30] Z. Zheng and P. Hong, "Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks," in *Advances in Neural Information Processing Systems*, pp. 7913–7922, 2018.

[31] J. H. Metzen, T. Genewein, *et al.*, "On detecting adversarial perturbations," in *Int. Conf. on Learning Representations (ICLR)*, 2017.

[32] Z. Gong, W. Wang, *et al.*, "Adversarial and clean data are not twins," *arXiv preprint arXiv:1704.04960*, 2017.

[33] J. D. Burton, *Cisco security professional's guide to secure intrusion detection systems*. Syngress Publ., 2003.

[34] E. Conrad, S. Misenar, *et al.*, *Eleventh Hour CISSP®: Study Guide*. Syngress, 2016.

[35] A. Kurakin, I. Goodfellow, *et al.*, "Adversarial examples in the physical world," in *Int. Conf. on Learning Representations (ICLR 17)*, 2017.

[36] N. Papernot, P. McDaniel, *et al.*, "The limitations of deep learning in adversarial settings," in *IEEE European Symp. on Security and Privacy (EuroS&P)*, pp. 372–387, IEEE, 2016.

[37] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP 17)*, pp. 39–57, IEEE, 2017.

[38] T. Lin and H. Zha, "Riemannian manifold learning," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 30, no. 5, pp. 796–809, 2008.

[39] R. Wang and X. Chen, "Manifold discriminant analysis," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 429–436, IEEE, 2009.

[40] J. B. Tenenbaum, V. De Silva, *et al.*, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[41] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[42] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in neural information processing systems(NeurIPS)*, pp. 585–591, 2002.

[43] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in neural information processing systems*, pp. 321–328, 2004.

[44] S. Hu, T. Yu, *et al.*, "A new defense against adversarial images: Turning a weakness into a strength," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1635–1646, 2019.

[45] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *IEEE symp. on computational intelligence for security and defense applications (CISDA)*, pp. 1–6, IEEE, 2009.

[46] Y. LeCun, L. Bottou, *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[47] F. Pedregosa, G. Varoquaux, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.