MANDA: On Adversarial Example Detection for Network Intrusion Detection System

Ning Wang, *Student Member, IEEE*, Yimin Chen, *Member, IEEE* Yang Xiao, *Student Member, IEEE* Yang Hu, Wenjing Lou, *Fellow, IEEE* and Y. Thomas Hou, *Fellow, IEEE*

Abstract—With the rapid advancement in machine learning (ML), ML-based Intrusion Detection Systems (IDSs) are widely deployed to protect networks from various attacks. 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 of which class it is misclassified into; and 2) AEs tend to be close to the decision boundary 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 both binary IDS and multi-class IDS on two datasets (NSL-KDD and CICIDS) under three state-of-the-art AE attacks. Our experimental results show that MANDA achieves high true-positive rate (98.41%) with a 5% false-positive rate.

Index Terms—Adversarial example (AE), AE detection, intrusion detection system

对入侵流量的小扰动（例如，稍微增加数据包到达间隔时间）

认为对抗样本接近原始流形

对抗样本还接近决策边界

1 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. An intrusion detection system (IDS) plays a significant role in safeguarding networks from malicious attacks [2]. There are mainly two types of IDS: signature-based detection [3] and anomaly-based detection [4]. Signature-based detection schemes work by extracting the traffic signature and comparing it 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 normal traffic model. With the advancement in ML in recent years, 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 these days, ML-based methods show much greater potential as they require little or no prior knowledge to work on emerging novel attacks.

基于签名的检测[3]和基于异常的检测[4]

机器学习用于训练代表良性的“规范”型，然后评估流量的可信度

ML technologies have seen great success in domains such as computer vision and natural language processing [5], [6], [7]. While applying to network intrusion detection, state-of-the-art IDSs usually employ advanced neural networks (e.g., LSTM) and learning schemes (e.g., meta-learning and active learning). An important security attack common to almost all machine learning models is the adversarial example (AE) attack [8], [9]. In such an attack, the adversary is able to craft a sample, often by applying small perturbations, which can mislead a well-trained model to output an arbitrary label other than its true label with a high probability. For an IDS, an attacker can launch AE attacks to significantly increase the false-positive rate and false-negative rate, rendering the IDS practically useless.

对于IDS，攻击者可以发动AE攻击，显着提高误报率和漏报率，使IDS几乎无用。

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. [10] leveraged a generative adversarial network (GAN) to transform original malicious traffic into adversarial traffic to fool the IDS. Wu et al. [11] employed deep reinforcement learning (DRL) to automatically and adaptively generate adversarial traffic flow to deceive the detection model. Rigaki et al. [12] utilized a GAN to adapt the Command and Control (C2) channel of malicious traffic to mimic a legitimate application's traffic (e.g., the Facebook chat network traffic), and therefore evaded the IDS. Shu et al. [13] 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 [10], [11], [12], [13], [14] 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 [15] before proceeding to the IDS. Defense schemes in the first category [16], [17] 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.

本文做的是2，但是我的工作属于1

In this paper, we propose MANDA, a MANifold and

RNN

- *N. Wang, Y. Xiao, and Y. T. Hou are with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061 USA (E-mail: {ning18,xiaoy,thou}@vt.edu);*
- *Y. Chen, Y. Hu, and W. Lou are with the Department of Computer Science, Virginia Tech, Blacksburg, VA 24061 USA (E-mail: {yiminchen,yanghu,wjlou}@vt.edu)*

The preliminary version of this paper was published in IEEE INFOCOM 2021 [1].

Decision boundary-based AE detection scheme for ML-based IDS. It is observed that the benign or malicious traffic events usually reside in a low-dimensional manifold (i.e., a cluster) embedded in the ambient feature space. An ML-based IDS aims to learn a decision boundary that discriminates malicious network traffic from benign network traffic. To explain the intuitions behind MANDA clearly, we use an AE generated from a malicious network event for an example. The AE fools the IDS model (i.e., evades the IDS) by traversing the decision boundary of the IDS model. To preserve the malicious property of the 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 still highly likely to discriminate it into the manifold of 'malicious' samples. This motivates us to leverage such inconsistency between the IDS decision boundary and the manifolds to detect an AE. In addition, given that AEs are usually closer to the decision boundary of the IDS model than normal samples, it is expected that when small noise is added, the classification result of an AE is more likely to change than that of a clean sample. This motivates us to use such changes of IDS classification results to detect an AE.

We further demonstrate that MANDA is also effective on multi-class IDS. A multi-class IDS differentiates multiple types of intrusion from benign network traffic with a single model, while a two-class IDS only detects one type of intrusion. As discussed above, MANDA is composed of two building blocks: a manifold-based (Manifold) method and a decision-boundary-based (DB) method. Manifold shows similar performance in multi-class intrusion detection as is shown in two-class IDS. In order to improve the performance of DB in multi-class IDS, we propose to adjust noise magnitude based on the different inter-class distances. 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) to achieve effective AE detection regardless of which AE attack is used.
- We demonstrate that Manifold generalizes well to multi-class IDS. We improve the performance of DB on multi-class IDS by customizing the noise magnitude for each decision boundary according to the inter-class distance and achieve effective AE detection on multi-class IDS.
- Our experimental results show that MANDA achieves 98.41% true-positive rate (TPR) with 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. On the CICIDS dataset, MANDA achieves as high as 98.50% TPR with 5% FPR under CW attack. We also demonstrate that MANDA outperforms Artifact [18], a state-of-the-art solution on AE detector, on both IDS task and image classification task.

2 Background and Related Work

This section reviews the previous work most related to our paper, including recent intrusion attacks on IDS and adversarial examples in deep learning. No prior work focuses on defense mechanisms against AE attacks on IDS to the best of our knowledge.

2.1 Network Intrusion Attacks

The information technology infrastructures, including the Internet, telecommunication networks, computer systems, and embedded industrial processors, are subject to various network intrusion attacks. A network probing attack searches for network vulnerabilities by scanning the network's connections (e.g., port scanning) to launch further attacks. Another type of network intrusion attack, 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. Unlike APT attacks, a distributed denial of service (DDoS) attack [20] 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 [21].

The recent development in machine learning has enabled new and powerful ML-based IDSs [20], [22]. At the same time, the rapid progress in adversarial machine learning brought out a novel network intrusion attack, i.e., adversarial example attack, to evade an ML-based IDS [10], [11], [12], [13], [14], [23], [24]. Xu et al. [14] proposed a general method to find evasive variants for a target classifier automatically. 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. [23] 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. [11] employed deep reinforcement learning (DRL) to automatically generate adversarial traffic flow to deceive a target detection model. In this attack, the reinforcement learning agent updates the adversarial samples based on the feedback from the target model, which can adapt to the

change of the temporal and spatial features of the traffic flows.

Besides those works aiming to find a perturbation vector for each specific input, some other works focus on finding a perturbation generator model. [24] designed a perturbation generator (a generative adversarial network) model aiming to generate perturbations that can be applied in real-time on live traffic. The blind adversarial perturbations can reduce the accuracy of state-of-the-art website fingerprinting by 90% by only adding 10% bandwidth overhead. Lin et al. [10] proposed IDSGAN, which leverages a generative adversarial network (GAN) to transform original malicious traffic into adversarial traffic instances so as to mislead the IDS to classify it as benign. Rigaki et al. [12] 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. [13] employed active learning and GAN to launch the adversarial example attack on an ML-based IDS and showed the great capability to attack an IDS using only limited prior knowledge. In sum, it is clear that AE attacks are posing a real threat to today's ML-based IDS considering the cost is ultra-low, and AE attacks themselves are constantly evolving.

2.2 Adversarial Example

Adversarial example (AE) has become one of the most important research topics in the past few years. It was first proposed in [8] 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: generating AEs (i.e., AE attacks) and dealing with AEs (i.e., AE defenses).

AE attacks. Multiple ways to modify a sample into an AE have been proposed, which lead to various AE attacks. Many AE generation methods compute the gradient of the model's loss for the input. The fast gradient sign method (FGSM) [9] moves the current input image along the direction that maximizes the loss to lead to misclassification. The basic iterative method (BIM) [25] applies FGSM iteratively with small steps. The Jacobian-based saliency map attack (JSMA) [26] seeks the top features that contribute to misclassification when applying a fixed distortion. It only perturbs the selected pixels until a misclassification is achieved. Carlini and Wagner [27] proposed optimization-based attacks (i.e., CW attacks) to find a successful AE with the smallest distortion. FGSM, JSMA, and CW are the most referenced among all known AE attacks.

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 [28], [29], reducing image size [30], [31], increasing resolution [32], and rotation or shifting [33] have been developed to detect AEs. Those methods are designed for images, and they are either not applicable or not effective when applied 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, these AE detectors hypothesize that statistical characteristics of AEs and clean data are different and thus distinguishable. Grosse et al. [34] applied the kernel-based two-sample test to distinguish AEs from clean data. Song et al. [35] leveraged generative models to decide whether an input sample was drawn from the same distribution of clean data. Zheng et al. [36] 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. [18] employed kernel-based density estimation to detect AEs. The drawback of the scheme in [18], [34] is that it requires an abundance of AEs to build the detector, similar to defense schemes based on adversarial training [37], [38].

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 [18] in IDS and also show its applicability for tasks with image input.

3 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.

3.1 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 an instance of 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' is also referred to as 'benign data', 'benign input', or 'normal traffic'.

TABLE 1: Symbol definition.

Symbol	Definition
$x \in \mathbb{R}^n$	input feature vector
$\mathcal{F}, \theta$	IDS model and Model parameter
$y \in \mathbb{R}^m$	output classification probability vector ($y = \mathcal{F}(\theta, x)$)
$c(x)$	the final output label, $c(x) = \text{argmax}_i y[i]$
x'	the feature-space AE generated based on x
η	Adversarial perturbation, $x' = x + \eta$
z, z'	corresponds to x and x' in the problem space
$J(\theta, x)$	loss of $\mathcal{F}$ on x
S_{diff}	differentiable feature set
$S_{non-diff}$	non-differentiable feature set
S_{func}	the functional feature set
$S_{non-func}$	the non-functional feature set
L_2	the L_2 distance between x and x'
p	the upper bound change ratio of a feature
R_i	the range of the i-th feature
f_{ji}	the correlation function from j-th feature to i-th feature
S^i_{corr}	the feature set that correlated to the i-th feature

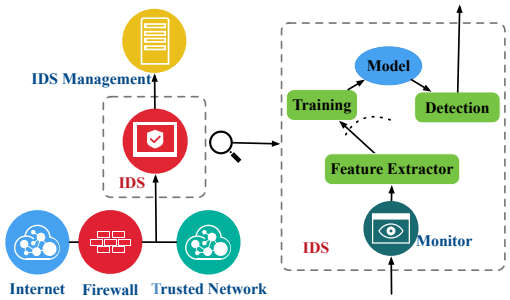


Fig. 1: System model of ML-based IDS.

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 intentionally crafted to fool the IDS model, i.e., either evading the IDS detection or creating false positives. The goal of 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.

To emphasize, 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 purposes, we always stick to the former case across the paper. In experiments, we explore both cases.

3.2 System Model

A typical architecture of an ML-based IDS is shown in Fig. 1. Usually, IDS is a passive infrastructure that rarely interferes with the network traffic under monitoring. An IDS sniffs the internal interface of the firewall in a read-only mode and sends alerts to an IDS management server via a read-and-write network interface [39], [40]. As Fig. 1 shows, an ML-based IDS is composed of the following modules [4]:

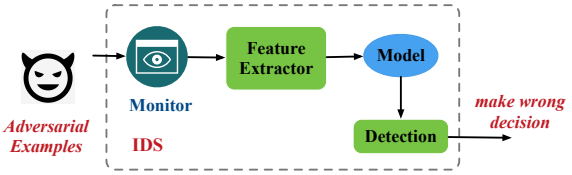


Fig. 2: Attack model of AE generating attacks.

- Network Traffic Monitor keeps tracking the ongoing network traffic of a communication and networking system.
- Feature Extractor processes the raw traffic data into 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 the 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 the IDS model.

3.3 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 as shown in Fig. 2. The attacker aims to fool the IDS to either classify a malicious traffic instance as benign (i.e., a False Negative) or classify a benign one as malicious (i.e., a False Positive). 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 on 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.

This paper considers the white-box attack to the IDS system, which is the most powerful 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. We use this powerful threat model to demonstrate the effectiveness of our defense. We also assume an attacker has black-box access to the 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 $y[i], i = 1, ..., m$ 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., $c(x) = \arg\max_i y[i]$. The

objective of AE generation is to search for $x' = x + \eta$ (η denotes a small perturbation) such that $c(x') \neq c(x)$. We further assume $J(\theta, x)$ is the loss function on an input x. The symbols used in the paper are also shown in Table 1.

Among the good amount of AE attacks proposed in the literature, we choose the four most representative and effective ones 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. [9] proposed a fast gradient sign method (FGSM) to generate AEs. Specifically, an attacker computes the gradient of the loss function with respect to x. It then moves the current x along the direction that maximizes $J(\theta, x)$. We can represent the generated AE as:
$$x' = x + \epsilon \text{sign} \nabla_x J(\theta, x).$$
2) BIM: Kurakin et al. [25] proposed a basic iterative method (BIM) which is an extension of FGSM. This attack applies FGSM multiple times with small steps. At each time of applying FGSM, BIM 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 [26], Papernot et al. proposed the Jacobian-based saliency map attack (JSMA), which applies iterative computation to seek features which contribute more to mis-classification 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:
$$S(x,t)[i] = \begin{cases} 0, \text{if } \frac{\partial f_t(x)}{\partial x_i} < 0 \text{ or } \sum_{j \neq t} \frac{\partial f_j(x)}{\partial x_i} > 0 \\ \frac{\partial f_t(x)}{\partial x_i} \left| \sum_{j \neq t} \frac{\partial f_j(x)}{\partial x_i} \right|, \text{otherwise.} \end{cases}$$
4) CW attack In [27], 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:
$$\min \|x' - x\| + c \cdot \max\{\max\{f_i(x') : i \neq t\} - f_t(x'), -\kappa\}$$
$$\text{subject to } x' \in [0,1]^n$$

It is worth noting that another well-known AE attack, the projected gradient descent (PGD) attack [41], is essentially the same with the BIM attack. We opt for the latter for convenience. In most cases, the CW attack outperforms the other three methods in terms of effectiveness and distortion.

4 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, i.e., pertaining to network packets with physical meanings. 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 identifying 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.

4.1 Problem-Space AE Attack on IDS

This section demonstrates how we generate AEs on IDS in problem-space. The concept of problem-space attack arises from the need for generating physically meaningful, domain-specific attack instances instead of only tweaking adversarial samples in a feature space [42]. The problem-space of an IDS is comprised of all possible traffic instances in the form of sequences of network packets. In contrast, the feature-space of an IDS is comprised of all possible feature vectors in the form of numerical entries representing packet length, packet inter-arrival time, etc. Prior AE generation algorithms [25], [26], [27] only focus on image inputs where the problem-space and the feature-space AEs are the same, i.e., a vector of pixels. The problem-space AE attacks 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 the problem space. First, we generate a feature-space AE x' from a clean input x. Second, we design a mapping function to project x' to the problem-space and obtain the ultimate problem-space AE z'. The corresponding representation of x in problem-space is denoted by z.

Traditional inverse mapping techniques [43], [44] are ineffective in mapping instances from feature-space to problem-space in our problem context because the mapping is neither invertible nor differentiable. We modify the AE generation approaches by nullifying the perturbations on non-differentiable features to make the mapping differentiable. We use $\boldsymbol{x}[i]$ and $\boldsymbol{x}'[i]$ denotes i-th feature of $\boldsymbol{x}$ and $\boldsymbol{x}'$ respectively. The features are divided into two parts: $\{S_{diff}, S_{non-diff}\}$. We force that $x'[i] = x[i]$ for $i \in S_{non-diff}$. Hence, we get $z'[i] = z[i]$ for $i \in S_{non-diff}$. After getting rid of non-differentiable features, we map the differentiable features of x back to z. In an IDS, non-differentiable features are categorical features, e.g., 'protocol type', 'service type', etc. Nullifying perturbations on such non-differentiable features helps to get a meaningful AE. On the other hand, we also demand that an AE preserves its original functionality, e.g., an intrusion flow maintains its attacking capability; a benign traffic flow retains its benign functionality. We refer to staying meaningful and maintaining the original functionality as preserving intrinsic property. We discuss how to help keep the intrinsic property below.

We observe that an attack may lose its malicious property if the attacker is allowed to manipulate features of a network traffic flow arbitrarily. Another observation is that an AE generated by the popular adversarial generation methods does not try to maintain potential data correlations between feature pairs. Such AE will be easily filtered by

以 IDS 问题空间为例，“端口号”和“使用的网络服务”是 IDS 的功能特征。 区分功能特征的一种简单方法是默认将具有分类值的特征设置为功能特征。 例如，采用三个可能的分类值（即“icmp”、“tcp”和“udp”）的“协议类型”是一个功能特性。 请注意，需要领域专业知识来确定最终的功能特征集，因为非分类特征也可以是功能性的

configuring data correlations as basic filtering rules. We refer to either scenario as losing AE's intrinsic properties. We summarize some general guidelines by adding several constraints on the AE generation to alleviate the problem. Compared to unconstrained AE generation, our AE generation method is more likely to preserve the properties and validity of an instance. Readers are referred to the detailed study in [45], [46] on preserving the properties of a network traffic flow in AE generation. These guidelines apply to various problem spaces, including but are not limited to IDS.

1) Differentiate the functional features S_{func} from non-functional features $S_{non-func}$. Try not to change the value of the functional features. Here the functional features are the features that directly impact the functionality of the instance. Take IDS problem space as an example, the 'port number' and 'network service used' are functional features for IDS. A simple way to differentiate functional features is to set features with categorical values as functional features by default. For example, the 'protocol type' that takes three possible categorical values (i.e., 'icmp', 'tcp' and 'udp') is a functional feature. Note that domain expertise is needed to determine the ultimate set of functional features as non-categorical features can also be functional.
2) Make sure that the value of a modified feature is still a valid one. The value of the modified feature needs to be in the range if the feature value is continuous. Features with discrete values should remain in the set of valid values after modification.
3) Keep a small perturbation magnitude for the non-functional features. The intrinsic properties of an instance may change because of non-functional features of which the perturbation magnitude is large enough.
4) Make a consistent modification on features that are correlated to each other. It is not realistic to assume that all features are independent of each other. Adding perturbations independently on correlated features may harm the validity of an instance.

The guidelines help to preserve an instance's intrinsic properties and validity at our best. We name the AE that follows the guidelines as a problem-space AE. We try to preserve the properties of adversarially generated examples by adding multiple constraints (i.e., guidelines). Note that it is still possible that a problem-space AE deviates from its desired properties or becomes invalid after perturbation. In our paper, compared to feature-space AE generation, we include the following restrictions for problem-space AE generation:

$$
\begin{aligned}
&\|\boldsymbol{x'} - \boldsymbol{x}\|_2 \leq L_2, \\
&\|\boldsymbol{x'}[i] - \boldsymbol{x}[i]\| \leq p * R_i, \text{ if } i \in (S_{diff} \cap S_{non-func}), \\
&\boldsymbol{x'}[i] = \boldsymbol{x}[i], \text{ if } i \in (S_{non-diff} \cup S_{func}), \\
&\boldsymbol{x'}[\boldsymbol{i}] = f_{ji}(\boldsymbol{x'}[j]), j \in S^i_{corr},
\end{aligned}
$$

where $\boldsymbol{x'}$ is an AE generated from $\boldsymbol{x}$, R_i denotes its range, L_2 denotes the maximum L_2 distance between $\boldsymbol{x}$ and $\boldsymbol{x'}$. S_{diff} corresponds to the set of differentiable features and p the maximum change ratio. Only the differentiable and non-functional features are eligible to modify. S^i_{corr} denotes indexes of features that are correlated to feature i and f_{ji} is the correlation function from feature j to feature i.

首先，我们深入研究流形学习以促进对我们的 AE 分类方案的理解。 流形学习假设输入数据位于或靠近嵌入环境空间的低维流形［47］。

4.2 Properties of AE

First, we dive into `manifold learning` to facilitate the understanding of our AE categorization scheme. Manifold learning assumes that input data reside on or close to a low-dimensional manifold embedded in the ambient space [47]. For example, a plane is a 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 [48]. Most manifold learning methods focus on data representation as in [49], [50], [51]. 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).

Next, we 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 $c(x) = \arg\max \mathcal{F}(\theta, x)$. $c(x) = 1$ indicates that x is classified as malicious while $c(x) = 0$ indicates that x is classified as benign. We use $x' = x + \eta$ to denote an AE generated from x. There are two attack scenarios: 1) $c(x) = 1$ (the true label of x is also 1) while the goal of AE attack is $c(x') = 0$; and 2) $c(x) = 0$ (the true label of x is also 0) while the goal of AE attack is $c(x') = 1$. All the analysis on the first attack scenario is also applicable to the second attack scenario as our defense does not depend on the value of $c(x)$. In what follows, we use the first attack scenario for consistent illustration purposes (we consider AEs of both scenarios in our experiments). *We assume that x' needs to keep the essential property of its true class, i.e., 'malicious'.* In other words, 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' would not be an AE, rather a new instance of 'benign' class that yields no attack impact.

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:

- `Case A`: *x' is close to the manifold of the 'malicious' class but far from the manifold of the 'benign' class.* This frequently occurs when malicious and benign manifolds 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 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 mutually exclusive. They may coexist for the same IDS model at different segments of the decision boundary.

Note that if x' is far away from the manifold of the 'malicious' class and close to that of the 'benign' class, x' is not a successful AE because x' breaks the above assumption

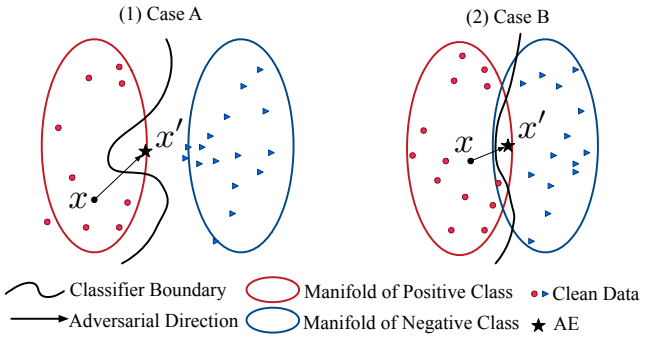


Fig. 3: The illustration of AEs in the 2-D view.

of *AE needs to keep the essential property of its true class*. The same argument applies to the case that x' is far from both manifolds.

4.3 MANDA

Here we introduce our AE detection scheme, MANDA, which is based on the above AE categorization. We assume that the IDS model has high accuracy on clean data since it makes less sense to discuss detecting AEs already misclassified by the IDS model. A clean input needs to traverse the decision boundary of the IDS model to be an AE. As mentioned in Section 1, MANDA consists of two components, Manifold and DB. Manifold combines both x's classification and manifold evaluation results to detect AEs. If the two outputs of x are inconsistent, x is highly likely to be an AE. DB examines 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 (and thus $c(x)$) changes frequently, x is highly likely to be an AE. We want to emphasize that both Manifold and DB can be used as a stand-alone AE detection scheme while MANDA combines them together for better performance.

4.3.1 Manifold

For Case A in Fig. 3, an AE lies in (or near) the manifold of the 'malicious' class but far from the 'benign' class. Meanwhile, the output label from the IDS model on the AE is 'benign'. Therefore, the IDS model results and manifold evaluation results on the same input are inconsistent. On the contrary, results for a clean input tend to be consistent. Thus 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 (i.e., 'malicious') and negative (i.e., 'benign') class, we employ a transductive learning model proposed by Zhou et al. in [52]. The learning method explores the intrinsic structure revealed collectively by a group of labeled and unlabeled data points. It guarantees both local consistency and global consistency of known data points. In other words, (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 concerning the intrinsic data structure. We use this learning method to obtain the manifold for each class in this paper.

Algorithm 1 Score-Compute() for Criterion 1 & 2

Input: input $x \in \mathbb{R}^n$, IDS model $\mathcal{F}(\theta)$, learned manifold $\mathcal{M}$
Output: $score_1, score_2$

$p \leftarrow \mathcal{M}(x)$ # Confidence vector of manifold evaluation
$q \leftarrow \mathcal{F}(\theta, x)$ # Classifier output
$score_1 \leftarrow \|p\| + \|q\| - \|p+q\|$ # Criterion 1
for $i = 0$ to N **do**
  $x_i = x + \mathcal{N}(0, \sigma^2)$
  $p_i \leftarrow \mathcal{F}(\theta, x_i)$
end for
$score_2 \leftarrow \frac{1}{N}\sum_{i=1}^{N}\|p_i\| - \frac{1}{N}\left\|\sum_{i=1}^{N} p_i\right\|$ # Criterion 2
return $score_1, score_2$

Algorithm 2 Manifold

Input: input $x \in \mathbb{R}^n$, IDS model $\mathcal{F}(\theta)$, learned manifold $\mathcal{M}$, threshold τ_1
Output: $isAdversarial \in \{False, True\}$

$score_1, \sim \leftarrow$ **Score-Compute**$(x, \mathcal{F}, \mathcal{M})$
if ($score_1 > \tau_1$) **then**
  $isAdversarial \leftarrow True$
end if
return $isAdversarial$

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 comparing the confidence vector from the manifold evaluation and the one from the IDS model as shown in Algorithm 1. Next, we compare $score_1$ to an optimal threshold τ_1 to decide whether an input sample is an AE or not (see Algorithm 2). The threshold τ_1 is selected by examining the statistics of clean data points' scores. In the evaluation section, we select the 95 percentile of the clean data points' scores as τ_1.

4.3.2 DB

Different from Case A, the two manifolds of 'malicious' and 'benign' classes 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 (or benign) input and x' is x's corresponding AE, i.e., $c(x)=1$ (or 0), $x'=x+\eta$, $c(x')=0$ (or 1); 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. This criterion can falsely conclude a clean input close to the boundary as an AE. Due to the curse of dimensionality, very few correctly classified clean inputs are close to the boundary [53]. 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 the IDS model's

Algorithm 3 DB

Input: input $x \in \mathbb{R}^n$, IDS model $\mathcal{F}(\theta)$, learned manifold $\mathcal{M}$, threshold τ_2
Output: $isAdversarial \in \{False, True\}$

$\sim, score_2 \leftarrow$ **Score-Compute**$(x, \mathcal{F}, \mathcal{M})$
if ($score_2 > \tau_2$) **then**
  $isAdversarial \leftarrow True$
end if
return $isAdversarial$

output when the input is applied with a small additive perturbation. For a near-boundary example, such a small perturbation may cause it to traverse the decision boundary. Consequently, the outputs of the 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 the IDS model is evaluated as the variance of the confidence vector $\mathcal{F}(\theta, x_i)$ of input $x_i = x + \mathcal{N}(0, \sigma^2), (i \in \mathbb{N}, i \leq N)$:

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

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.*

We illustrate DB method in Algorithm 3. DB first computes $score_2$ using Eq. 1. Next, DB compares $score_2$ to an optimal threshold τ_2 to decide whether an input sample is an AE or not. Note that the threshold τ_2 is selected similarly as τ_1.

4.3.3 MANDA (Manifold+DB)

MANDA is designed to make the best of Manifold and DB for AE detection. We first generate some AEs using the clean training dataset. We mix these AEs with the clean training inputs together and use $\mathbf{X}$ to denote the mixed inputs. Next, MANDA obtains $[score_1, score_2]$ of each input in $\mathbf{X}$ by executing **Algorithm 1**. We attach label 1 to $[score_1, score_2]$ if the input is an AE and label 0 if the input is clean. Next, we create a new dataset $([\mathbf{S_1}, \mathbf{S_2}], \mathbf{Y_{adv}})$ where $[\mathbf{S_1}, \mathbf{S_2}]$ represents the set of score for each input. Finally, MANDA trains a logistic regression model on the new dataset and uses it for AE detection. Algorithm 4 illustrates the process of MANDA.

Algorithm 4 MANDA

Input: IDS model $\mathcal{F}(\theta)$, learned manifold $\mathcal{M}$, test data point $x_{test} \in \mathbb{R}^n$, input dataset $\mathbf{X}$, AE flag $\mathbf{Y_{adv}}$
Output: $isAdversarial \in \{False, True\}$

if training **then**
  $\mathbf{S_1}, \mathbf{S_2} \leftarrow$ **Score-Compute**$(\mathbf{X}, \mathcal{F}, \mathcal{M})$
  $model \leftarrow$ LogistcRegression$(\mathbf{S_1}, \mathbf{S_2}, \mathbf{Y_{adv}})$
else
  $score_1, score_2 \leftarrow$ **Score-Compute**$(x_{test}, \mathcal{F}, \mathcal{M})$
  $isAdversarial \leftarrow model(score_1, score_2)$
end if
return $isAdversarial$

5 Experimental Results

5.1 Datasets 互联网流量数据集 NSL-KDD

NSL-KDD. We use the internet traffic dataset, NSL-KDD [54] (also used in AE attacks in IDS [10], but [9] does 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 in the testing 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. Our experiments 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), further processed into 121 numerical features as an input-space (feature-space) vector.

CICIDS2017. CICIDS2017 dataset [55] contains benign traffic and 12 up-to-date attacks, which resembles the real-world Packet Capture (PCAPs) data. The flow-based dataset is extracted from the PCAPs files using CICFlowMeter[1] according to the time stamp, source, and destination IPs, source and destination ports, protocols. There are 2,416,775 data records after primary preprocessing (e.g., remove NaN). Each data record, corresponding to a traffic flow, contains 78 features and is attached with a label indicating the attack type or benign. We randomly split the dataset into a training set containing 1,933,420 data records and a test set containing 483,355 data records.

MNIST. We also evaluate our approach on an image dataset, MNIST [56], 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.

图像数据集 MNIST

5.2 Experiment Settings and Evaluation Metrics

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×12, a GeForce RTX 2080 Ti GPU, and Ubuntu 18.04.3 LTS. The IDS model is a multilayer 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 [57]. We implement four AE attacks including FGSM, BIM, CW (the

1. https://github.com/ahlashkari/CICFlowMeter

只评估过了Dos大类下的攻击

三层网络，一层输入层，一层输出，一层隐藏层

内在特征、内容特征、基于时间的特征和基于主机的特征

实现了四种攻击，但只用了前三种，why?

Perturbation restriction (%)	FGSM		BIM		CW	
	Acc (%)	L_2	Acc (%)	L_2	Acc (%)	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 2: IDS Accuracy under AE Attacks on the NSL-KDD Dataset.

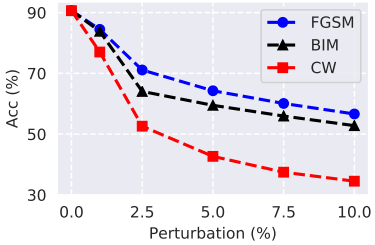


Fig. 4: IDS model accuracy from TABLE 2

针对测试集中原本能被正确分类的样本生成了相应的对抗样本

L_2-norm version) and JSMA (cf. Section 3.3). We adapt the first three to the problem space of IDS. We generate AEs on the test samples that the IDS model correctly classifies in each experiment. 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 [18], the same as in [15], [53]. Artifact is proposed by Feinman et al. in [18] and becomes one of the state-of-the-art AE detection schemes. Different from MANDA, Artifact uses kernel density estimation (KDE) and Bayesian neural network uncertainty as two criteria to detect AEs.

1：1检测

对比方案18

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

We count the True Positives (TPs), False Positives (FPs), True Negatives (TNs), and False Negatives (FNs) of AE detection as evaluation metrics defined as follows.

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

Then we have TPR $= \frac{\#TPs}{\#TPs+\#FNs}$, FPR $= \frac{\#FPs}{\#TNs+\#FPs}$. 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.

5.3 Results of IDS

5.3.1 AE Attacks on NSL-KDD

We show the classification accuracy of the IDS model under FGSM, BIM and CW attacks in TABLE 2. We draw two main conclusions from the experimental results. First, the larger the 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 4.1). TABLE 2 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 2) is higher than that of problem-space attack (i.e., any row other than the last in TABLE 2) because the latter faces more restrictions in generating AEs (See Section 4.1). Problem-space AE attacks can still cause a very low accuracy of the 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 [58]. TABLE 3 shows the results of our problem-space AEs generated from the 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 3: Transferability of 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

5.3.2 Detection Performance on NSL-KDD

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 4 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 4. 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 is pointed out in Section 4.3, our DB detection method is to detect those AEs which Manifold 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, making them great candidates for fast online detection.

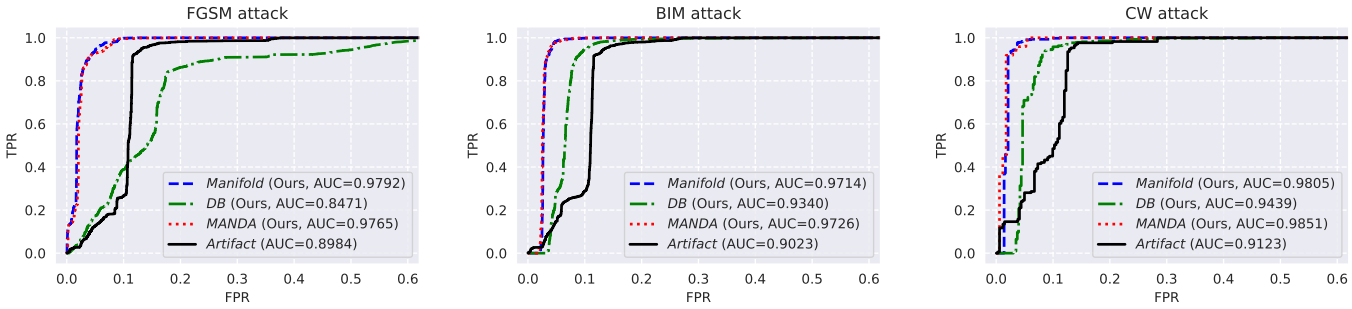
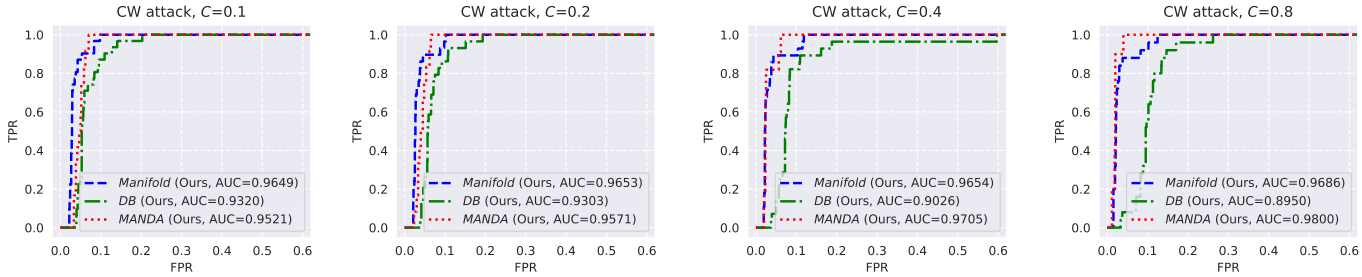


Fig. 5: ROC curve of AE detection methods under FGSM, BIM and CW attacks on NSL-KDD.

TABLE 4: AE Detection Performance on the NSL-KDD Dataset

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` [18]	12.60	96.63	0.8984	14.78	96.31	0.9023	28.07	97.66	0.9123



(a) (b) (c) (d)

Fig. 6: ROC curves of AE detection under *Adaptive* CW attacks on the NSL-KDD dataset.

5.3.3 Adaptive AE Attack

We further assume a stronger threat model where an attacker knows the defense mechanism. With this knowledge, an attacker can launch an adaptive attack to generate adversarial examples that are far from the decision boundary. Theoretically, the decision-boundary-based detection component (`DB`) will fail to detect this type of adaptive attack since the confident AE will be far from the decision boundary with high probability. However, the manifold-based detection component (`Manifold`) will still be effective in AE detection. The reason is as follows. Firstly, the generated AE should maintain its original properties regardless of its confidence. Therefore, the manifold evaluation is highly likely to predict this AE to its original class. On the other hand, the target classifier will classify it as a class other than the original one. We can still find an inconsistency between the manifold evaluation and classifier prediction. In this way, the manifold-based detection component will successfully detect this AE. We further experimentally evaluate this type of adaptive adversarial attack as follows.

We implement the adaptive adversarial attack based on the CW attack. We enable an adjustable confidence level of AEs in this attack. Specifically, we use the difference between the confidence in the adversarial class and confidence in the ground truth class to represent the strength of AEs. And we denote this metric as C. We show `MANDA`'s performance against AEs with different Cs. Figure 6(a) 6(b) 6(c) 6(d) show the AE detection results when $C = 10\%, 20\%, 40\%,$ and 80% respectively. `MANDA` and its two components, i.e., `DB` and `Manifold`, are all included in the figures. We can see that the performance of `DB` degrades with the increase of AE confidence as expected. However, `Manifold` shows better detection performance with the increasing AE confidence. The reason is that the inconsistency between the victim model prediction and the manifold evaluation increases with C. `MANDA` consistently shows high detection rates under all C values, indicating that `MANDA` is effective against this type of adaptive attack.

5.3.4 AE Attacks on CICIDS

Compared to the NSL-KDD dataset, we build a multiple-class IDS model on the CICIDS dataset instead of a two-class model. The IDS model is more powerful as it can not only detect a malicious incoming traffic flow but also recognize its attack type. However, this also makes AE detection for a multi-class IDS much more challenging due to that each type of attack tends to hide its maliciousness in its own way. In order to make a meaningful attack, we only consider the attack scenario where an attacker tries to evade the detection (i.e., to be benign) but not the scenario where an attacker pretends to be another attack type or a normal one pretends to be malicious. The accuracy of IDS on the CICIDS dataset under no adversarial attack is 98.2% in our evaluation.

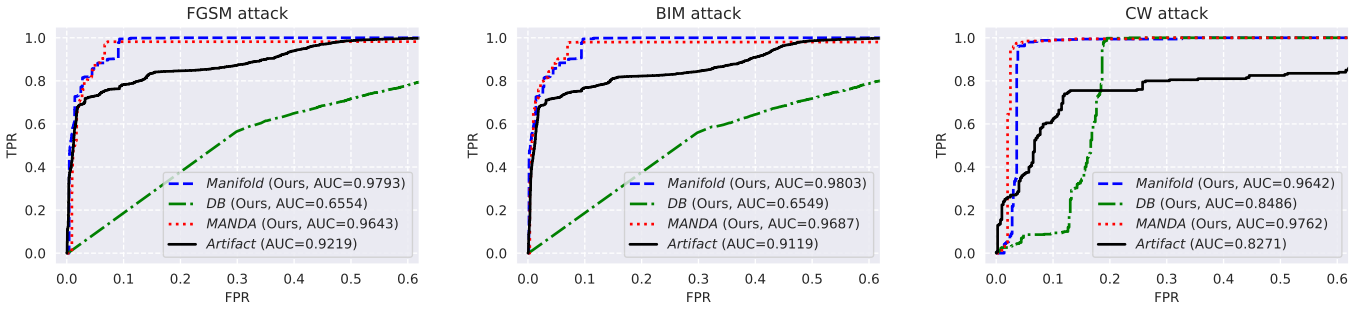


Fig. 7: ROC curve of AE detection under FGSM, BIM and CW attacks on CICIDS. The victim model is multi-class IDS.

TABLE 5: ASR on IDS for the CICIDS Dataset.

Perturbation restriction (%)	FGSM	BIM	CW
1.0	72.8	87.8	100.0
2.5	83.6	90.0	100.0
5.0	85.4	90.3	100.0
7.5	86.0	90.4	100.0
10.0	86.6	92.7	100.0
None	99.7	99.7	100.0

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

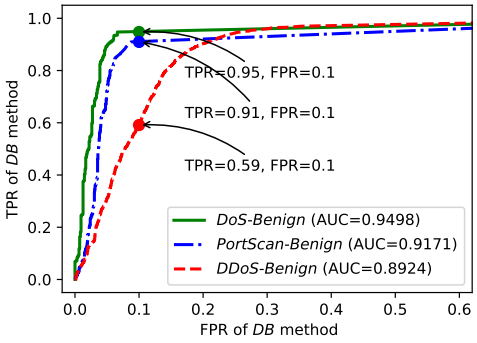


Fig. 8: The ROC curve of DB detection method on AEs generated by FGSM. The victim models are three two-class IDSs: DoS Intrusion denotes a two-class IDS that differentiate DoS attack from benign. Similar with PortScan and DDoS.

We show the effectiveness of AE attacks including FGSM, BIM and CW on the IDS model. To give a clearer view on the attack effectiveness, TABLE 5 shows the Attack Success Rate (ASR) instead of the IDS accuracy under attacks as TABLE 2 does. The two evaluation metrics, ASR and IDS accuracy, are correlated by $\text{Acc}_{under-attack} = (1 - \text{ASR}) * \text{Acc}_{no-attack}$ where $\text{Acc}_{under-attack}$ denotes the IDS accuracy under AE attacks and $\text{Acc}_{no-attack}$ denotes the original accuracy of IDS without attack. For instance, with ASR $= 90\%$, we can derive that the IDS model accuracy drops to $98.2\% * (1 - 90\%) = 9.8\%$.

We observe that the ASR on the CICIDS dataset is higher than that on the NSL-KDD dataset. One possible reason is that the IDS models designed for the two datasets rely on different feature engineering methods. As discussed above, another reason might be that the AE generation for a multiple-class model is easier than a binary model. The observation that AE attacks successfully disrupt two IDSs built on different feature engineering methods demonstrates that AE attacks are powerful against IDS.

TABLE 6: FPR of AE Detection When TPR Reaches 0.95 on the CICIDS Dataset.

Detection Method	FGSM	BIM	CW
Manifold	0.091	0.094	0.038
DB	0.828	0.827	0.186
MANDA	**0.067**	**0.070**	**0.025**
Artifact	0.419	0.439	0.653

5.3.5 Detection Performance on CICIDS

We evaluate the performance of MANDA on the CICIDS2017 dataset and compare it with the performance of Artifact. For better understanding, the preliminaries of the multi-class ML model are first given.

In a multi-class problem, there exists a decision boundary between every two classes. A decision boundary is a locus of points on which a posteriori probabilities are the same, and it can be a point, line, plane, hyperplane, solid, hypersolid, curved surface, or curved hypersurface [59]. The definition of decision boundary is as follows:

Definition 1. For a n-class model $\mathcal{F}$ with model parameter θ, a decision boundary between class i and j ($i, j \leq n$) is defined as

$$D_{ij} : \{\mathbf{x} | \mathcal{F}(\theta, \mathbf{x})[i] = \mathcal{F}(\theta, \mathbf{x})[j]\}$$

where $\mathcal{F}(\theta, \mathbf{x})$ is the confidence vector, and $\mathcal{F}(\theta, \mathbf{x})[i]$ is the likelihood that the model predicts $\mathbf{x}$ as class i.

The inter-class distance between class i and j can be evaluated by the averaged Euclidean distance between a set of data points from class i (denoted by c_m) and a set of data points from class j (denoted by c_n),

$$l_{ij} = \frac{1}{|c_m||c_n|} \sum_{\mathbf{x}_a \in c_m} \sum_{\mathbf{x}_b \in c_n} \|\mathbf{x}_a - \mathbf{x}_b\|_2.$$

First, we show ROC curves of detecting FGSM, BIM, and CW attacks in Fig. 7. Similar as on NSL-KDD, Manifold and MANDA outperform Artifact on CICIDS. We observe that the AE detection performance of DB on the CICIDS dataset is not comparable to other detection methods. The reason is that we employ a constant Gaussian noise with σ^2 variance in the current design of DB. The constant noise magnitude is not able to evaluate the closeness of data points to their corresponding decision boundaries since the inter-class distance varies from one to another.

We analyze the performance of DB on multi-class IDS by a toy example. Assume that we have two data points $\mathbf{x}_a$ and $\mathbf{x}_b$. $\mathbf{x}_a$ is most likely from Class 1 or Class 2, and the inter-class distance is $l_{12} = 1$. The L_2 norm distance

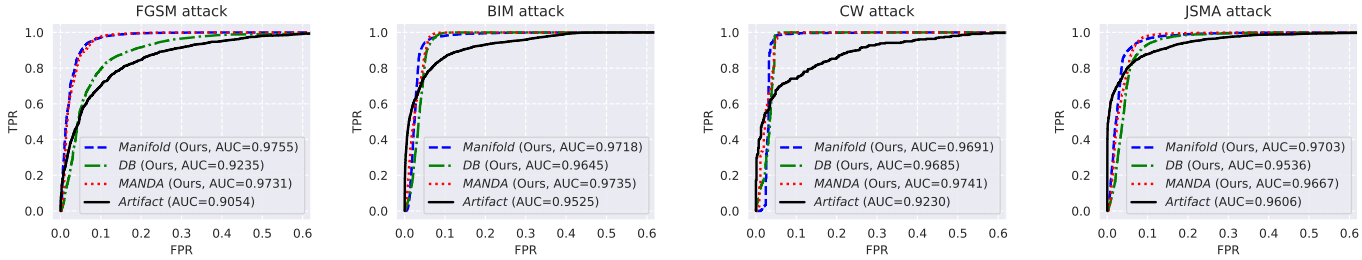


Fig. 9: ROC curve of AE detection methods under FGSM, BIM, CW, and JSMA attacks on the MNIST dataset.

from $\mathbf{x}_a$ to decision boundary D_{12} is 0.8. We would say $\mathbf{x}_a$ is not close to D_{12} since the ratio $0.8/1$ is large. $\mathbf{x}_b$ is most likely from Class 3 or Class 4, and $l_{34} = 100$. The L_2 norm distance from $\mathbf{x}_b$ to decision boundary D_{34} is also 0.8. $\mathbf{x}_b$ is believed to be close to D_{34} since $0.8/100$ is small. When we employ DB using a Gaussian noise with $\sigma = 2$ to evaluate the closeness of the two data points to decision boundaries, we would highly likely get that both $\mathbf{x}_a$ and $\mathbf{x}_b$ are close to decision boundaries. In this way, we made an FP on $\mathbf{x}_a$. When we decrease the σ to 0.2, we eliminate the FP on $\mathbf{x}_a$. However, we would make an FN on $\mathbf{x}_b$. Therefore, we need to customize the noise scale according to the relative position of a data point to different decision boundaries, which is a potential future work. On the other hand, Manifold is not sensitive to the inter-class distance so that we can still achieve good detection performance with Manifold.

We further verify our hypothesis by exploring the performance of DB when the victim IDS is a two-class IDS on the same dataset (i.e., CICIDS) in Fig. 8. By comparing Fig. 8 with Fig. 7, we can see that DB achieves much better detection performance when the victim model is a two-class IDS rather than a multi-class IDS.

We further compute the minimum FPR of different detection methods when they achieve 0.95 TPR in TABLE 6. From the table, we can see that MANDA achieves 0.067 FPR under FGSM attack. Other AE detection methods result in higher FPR while achieving the same level of TPR. For BIM attack and CW attack, MANDA achieves the smallest FPR among all detection methods. The FPR of DB is larger than all the other AE detection methods, indicating that it is not suitable to be used alone. However, we can improve the AE detection performance of Manifold by incorporating DB to Manifold as shown in TABLE 6. We note that employing noise with different variances in DB as discussed above would improve the detection performance of DB.

In sum, Manifold and MANDA achieve excellent AE detection performance for IDS models: they achieve both high AUC scores and TPR. As pointed out in Section 4.3, our DB detection method is best at detecting 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.

5.4 Results on MNIST

We also apply MANDA to the MNIST dataset to evaluate their performance, to demonstrate that the proposed approach

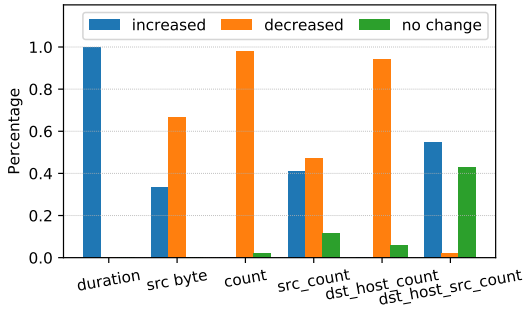


Fig. 10: The percentage of AEs that a selected feature (duration, src_byte, etc) value is increased or decreased.

can potentially be used for other application scenarios. We add a feature extractor to convert an original image input (i.e., a pixel vector) into a low-dimensional feature vector when dealing with an image input. 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. Compared with the original input size of 28×28, the size of the extracted high-level features is smaller (i.e., 64×1).

5.4.1 Detection Performance

We show the results of AE detection of the main model under FGSM, BIM, JSMA and CW attacks in Fig. 9. 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 detecting adversarial tampering of images. We will continue to explore more application scenarios in the future work.

6 Lessons Learned from AEs against IDS

One interesting observation in our experiments is that the AE generation in the problem space of IDS exhibits some common patterns, which sheds some insights in evading IDS.

We compare AEs to the corresponding clean traffic flows and summarize the modification patterns. For each feature, there are two possible modifications: either increasing or decreasing the value of the feature. Different types of attacks need different modification patterns so as to evade IDS.

We take the DoS attack from the NSL-KDD dataset as an example to show its modification pattern. We randomly select 200 instances of DoS traffic flow and apply the AE attack on the selected traffic flow. Among all the 41 features, we only allow modifying 6 features as shown in Fig. 10. The meaning of each feature is available [60]. Among 200 generated AEs, 51 AEs succeed in evading IDS. We summarize the likelihood that a feature is increased or decreased to achieve a successful evasion attack in Fig. 10. From Fig. 10 we can see that an attacker is able to evade the detection by increasing the connection duration, decreasing the number of connections per second, etc. Such results provide guidelines to design evasion attacks.

7 Conclusions

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 and the CICIDS dataset shows that `MANDA` outperforms the state-of-the-art statistical test model (i.e., `Artifact`) by achieving higher AUC scores and higher true-positive rate with a 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.

Acknowledgments 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 CNS-1916902, and the Virginia Commonwealth Cyber Initiative (CCI).

References

[1] N. Wang, Y. Chen, Y. Hu, W. Lou, and Y. T. Hou, "Manda: On adversarial example detection for network intrusion detection system," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pp. 1–10, IEEE, 2021.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

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

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

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

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

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

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

[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] B. Wang, Y. Zheng, W. Lou, and Y. T. Hou, "Ddos attack protection in the era of cloud computing and software-defined networking," *Computer Networks*, vol. 81, pp. 308–319, 2015.

[21] C. Kolias, G. Kambourakis, and et al., "Ddos in the iot: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[22] N. Wang, Y. Chen, Y. Hu, W. Lou, and Y. T. Hou, "Feco: Boosting intrusion detection capability in iot networks via contrastive learning," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, IEEE, 2022.

[23] 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.

[24] M. Nasr, A. Bahramali, and A. Houmansadr, "Defeating dnn-based traffic analysis systems in real-time with blind adversarial perturbations," in *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

[25] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[26] 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.

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

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

[29] 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.

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

[31] 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.

[32] 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.

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

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

[35] 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.

[36] 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 (NeurIPS)*, pp. 7913–7922, 2018.

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

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

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

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

[41] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[42] F. Pierazzi, F. Pendlebury, J. Cortellazzi, and L. Cavallaro, "Intriguing properties of adversarial ml attacks in the problem space," in *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 1332–1349, IEEE, 2020.

[43] S. Liao and Y. Zhao, "On the method of directly defining inverse mapping for nonlinear differential equations," *Numerical Algorithms*, vol. 72, no. 4, pp. 989–1020, 2016.

[44] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," *arXiv preprint arXiv:1605.09782*, 2016.

[45] X. Peng, W. Huang, and Z. Shi, "Adversarial attack against dos intrusion detection: An improved boundary-based method," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 1288–1295, IEEE, 2019.

[46] Y. Tian, Y. Wang, E. Tong, W. Niu, L. Chang, Q. A. Chen, G. Li, and J. Liu, "Exploring data correlation between feature pairs for generating constraint-based adversarial examples," in *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 430–437, IEEE, 2020.

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

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

[49] 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.

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

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," in *ICISSP*, pp. 108–116, 2018.

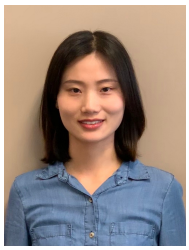
[56] Y. LeCun, L. Bottou, *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

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

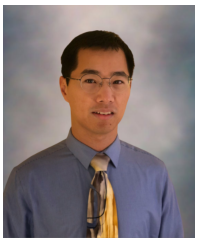
[58] N. Papernot, P. McDaniel, *et al.*, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[59] C. Lee and D. A. Landgrebe, "Feature extraction based on decision boundaries," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 15, no. 4, pp. 388–400, 1993.

[60] L. Dhanabal and S. Shantharajah, "A study on nsl-kdd dataset for intrusion detection system based on classification algorithms," *Int. Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, pp. 446–452, 2015.

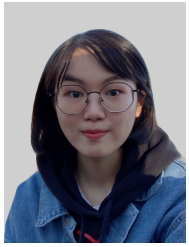


Ning Wang received her B.E. degree in communication engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2015, where she also received her M.S. degree in electronics and communication engineering in 2018. Since Fall 2018, she has been working towards the Ph.D. degree in the Department of Electrical and Computer Engineering at Virginia Tech, Blacksburg, VA, USA. Her research interests include anomaly detection, adversarial machine learning, and differential privacy.



Yimin (Ian) Chen is currently a postdoctoral associate in the Department of Computer Science at Virginia Tech. He received his Ph.D. from Arizona State University in 2018. His PhD research focused on security and privacy in mobile computing. Before that, he received his B.S. from the School of Electronics Engineering and Computer Science at Peking University in 2010 and his M.Phil. from the Department of Electrical Engineering at Chinese University of Hong Kong in 2013. His research interests encompass security and privacy in machine/meta learning and networked systems.

Yang Xiao is currently pursuing the Ph.D. degree with the Bradley Department of Electrical and Computer Engineering at Virginia Tech, supervised by Prof. Wenjing Lou. He received his B.S. degree from the School of Electrical and Information Engineering at Shanghai Jiao Tong University and M.S. degree from the Electrical Engineering and Computer Science Department at University of Michigan, Ann Arbor. His research interests lie in network security, blockchain and distributed system security, and wireless network security.



Yang Hu is a Ph.D. student in Computer Science at Virginia Tech. She received her B.S. degree in Computer Science from Beihang University, Beijing, China, in 2019.

Wenjing Lou (F'15) is the W. C. English Endowed Professor of Computer Science at Virginia Tech and a Fellow of the IEEE. Her research interests cover many topics in the cybersecurity field, with her current research interest focusing on wireless network security, trustworthy AI, blockchain, and security and privacy problems in the Internet of Things (IoT) systems. Prof. Lou is a highly cited researcher by the Web of Science Group. She received the Virginia Tech Alumni Award for Research Excellence in 2018. She received the INFOCOM Test-of-Time paper award in 2020. She was the TPC chair for IEEE INFOCOM 2019 and ACM WiSec 2020. She was the Steering Committee Chair for IEEE CNS conference from 2013 to 2020. She is currently a steering committee member of IEEE INFOCOM and IEEE Transactions on Mobile Computing. She served as a program director at the US National Science Foundation (NSF) from 2014 to 2017.

Y. Thomas Hou (F'14) is Bradley Distinguished Professor of Electrical and Computer Engineering at Virginia Tech, Blacksburg, VA, USA, which he joined in 2002. His current research focuses on developing innovative solutions to complex science and engineering problems arising from wireless and mobile networks. He is also interested in wireless security. He has published over 300 papers in IEEE/ACM journals and conferences. His papers were recognized by eight best paper awards from IEEE and ACM. He holds five U.S. patents. He authored/co-authored two graduate textbooks: *Applied Optimization Methods for Wireless Networks* (Cambridge University Press, 2014) and *Cognitive Radio Communications and Networks: Principles and Practices* (Academic Press/Elsevier, 2009). Prof. Hou was named an IEEE Fellow for contributions to modeling and optimization of wireless networks. He was/is on the editorial boards of a number of IEEE and ACM transactions and journals. He was Steering Committee Chair of IEEE INFOCOM conference and was a member of the IEEE Communications Society Board of Governors. He was also a Distinguished Lecturer of the IEEE Communications Society.