A Deeper Analysis of Adversarial Examples in Intrusion Detection

Mohamed Amine Merzouk[1,2(✉)], Frédéric Cuppens[3], Nora Boulahia-Cuppens[3], and Reda Yaich[4]

[1] Ecole Nationale Supérieure d'Informatique, Algiers, Algeria
fm_merzouk@esi.dz
[2] IMT Atlantique, Rennes, France
mohamed-amine.merzouk@imt-atlantique.fr
[3] Polytechnique Montréal, Montréal, Canada
{frederic.cuppens,nora.boulahia-cuppens}@polymtl.ca
[4] IRT SystemX, Plaiseau, France
reda.yaich@irt-systemx.fr

Abstract. During the last decade, machine learning algorithms have massively integrated the defense arsenal made available to security professionals, especially for intrusion detection. However, and despite the progress made in this area, machine learning models have been found to be vulnerable to slightly modified data samples called adversarial examples. Thereby, a small and well-computed perturbation may allow adversaries to evade intrusion detection systems. Numerous works have already successfully applied adversarial examples to network intrusion detection datasets. Yet little attention was given so far to the practicality of these examples in the implementation of end-to-end network attacks. In this paper, we study the applicability of network attacks based on adversarial examples in real networks. We minutely analyze adversarial examples generated with state-of-the-art algorithms to evaluate their consistency based on several criteria. Our results show a large proportion of invalid examples that are unlikely to lead to real attacks.

Keywords: Adversarial machine learning · Adversarial examples · Intrusion detection · Evasion attacks

1 Introduction

The importance of Artificial Intelligence (AI) and particularly Machine Learning (ML) in cybersecurity cannot be overstated. The synergistic integration of the two disciplines is considered by most specialists as one of the most profitable advances in cybersecurity [30]. Indeed, a description of the current threats landscape is sufficient to understand the interest of cybersecurity professionals from industry and academia in AI and ML. However, it is still relatively simple to deliberately mislead ML models, by means of what is commonly called Adversarial Machine Learning (AdvML) attacks.

J. Garcia-Alfaro et al. (Eds.): CRiSIS 2020, LNCS 12528, pp. 67–84, 2021.
https://doi.org/10.1007/978-3-030-68887-5_4

AdvML research community has been very active in the last few years to illustrate the fragility of ML models with regards to Adversarial Examples (AEs) [19]. AEs are data samples to which a small and deliberate perturbation is added to maliciously influence the output of an ML model towards erroneous predictions. For instance, applied to intrusion detection models, an attacker could transform an instance, that was originally classified as an attack, to be misleadingly classified as a benign entry. Thus allowing adversaries to evade intrusion detection systems.

Contribution: This article tries to answer the question: *Do we really need to worry about AEs in intrusion detection?* In order to do so, we investigate the practicality of AEs generated by state-of-the-art approaches in the execution of end-to-end cyberattacks on computer networks. We provide a comprehensive literature review of research initiatives addressing the vulnerability of intrusion detection systems to AEs. We also design and train an intrusion detection model on the well-known NSL-KDD dataset [33] and generate AEs against it with the most prominent methods. Furthermore, we evaluate their impact on the model and their distance from the original examples. Through an in-depth analysis of the network features of these examples, we identify several criteria that invalidate these attacks. In practice, the outcomes of our work are three-fold:

- An automated environment to train an intrusion detection model and analyse the impact and the consistency of AEs generated against it. The environment is available through a public repository[1] to stimulate further investigations.
- An implementation of Carlini and Wagner [6] L_0-*attack* that is made available to the community through the well-established open-source library Adversarial Robustness Toolbox [23].
- A description of validity criteria that can be used to discard unpractical AEs (cf. Sect. 6).

Paper Organization: Section 2 defines some basic background about artificial neural networks, adversarial machine learning, and the algorithms we use to generate AEs. In Sect. 3 we propose a literature review of research initiatives applying AdvML to intrusion detection. Section 4 details the experimental methodology we followed. In Sect. 5 we present and discuss the results of our experiments and analyze the generated AEs. Section 6 introduces the list of validity criteria for practical AEs. Finally, concluding remarks and future works are detailed in Sect. 7.

2 Background

In this section, we present some background knowledge on artificial neural networks, adversarial machine learning and adversarial examples.

[1] https://github.com/mamerzouk/adversarial_analysis.

2.1 Artificial Neural Networks

An Artificial Neural Network (ANN) is a machine learning model represented as a function $f(\cdot)$ that takes a data sample $x \in \mathcal{R}^n$ as an input and outputs a prediction l. ANNs are made of interconnected layers of neurons (perceptrons): small units that compute the sum of their inputs, weighted by the model parameters θ, and pass it through a non-linear activation function to produce an output (activation); this output is then propagated to be the input the neurons of the next layer.

The parameters of a neural network are randomly initialized and optimized through training. The loss function $J_\theta(x, l)$ estimates the error of the model by computing the difference between its output and the correct label. The gradient of the loss function with respect to the parameters $\nabla J_\theta(x, l)$ is then used by algorithms like Stochastic Gradient Descent or Adam to optimize the parameters in order to minimize the loss.

2.2 Adversarial Machine Learning

Adversarial Machine Learning (AdvML) is a recent research discipline that aims to evaluate and improve the robustness of machine learning models against malicious manipulations. The extensive literature in this field reports a wide variety of attacks that fall into four categories [5]:

- *Poisoning attacks* are achieved before the training phase by introducing perturbations among the training data to generate a corrupted model [4].
- *Evasion attacks* happen after the model is trained. They are used to manipulate the input data of a model to provoke erroneous predictions [3].
- *Extraction attacks* try to steal the parameters of a remote model in order to reproduce its behavior or rob confidential information [13].
- *Inversion attacks* abuse a model to extort sensitive information learned from the training data [9].

In our work, we focus on evasion attacks; there are two main types of them: *Untargeted attacks* that aim to cause the model to make erroneous predictions regardless of the output result and *Targeted attacks* that are much complex as they intend to orient the erroneous outcome towards a specific result. The two approaches are equivalent in binary classification (between two classes).

2.3 Adversarial Examples

Adversarial Examples (AEs) are inputs deliberately crafted to fool machine learning models. An AE x' is generally based on a clean example x to which a well-computed and minimal perturbation η is added. This perturbation must be sufficiently important to misclassify the sample in the false class l' instead of the correct class l, while limiting changes to maintain malicious functionality and minimize effort [34]. These changes are communally measured with distance metrics like L_p norms. We described below the three most used L_p norms.

- L_0 measures the number of perturbed features i such as $x_i \neq x'_i$.
- L_2 measures the Euclidean distance between two samples $\sqrt{\sum_{i=1}^{n} (x_i - x'_i)^2}$.
- L_∞ measures the maximum perturbation applied to any data feature.

The problem of finding AEs with minimal perturbation regarding an L_p norm is formulated in Eq. 1. In targeted attacks, l' is known in advance, while in untargeted attacks, l' can be any class other than the correct class l. We also assume that the features must stay in a limited interval we refer to as I.

$$\text{minimize } \|x - x'\|_p \text{ such that } f(x') = l', f(x) \neq l', x' \in I^n \tag{1}$$

This problem being too complex to solve, Szegedy *et al.* [32] reformulated it in Eq. 2, where c is a positive constant minimized by line-search. They then used the box-constrained L-BFGS optimization method to solve it.

$$\text{minimize } c \cdot \|x - x'\|_2 + J_\theta(x', l') \text{ such that } x' \in I^n \tag{2}$$

2.4 Adversarial Examples Generation Methods

We present some of the ground-breaking methods for the generation of AEs. These methods will later be used in our experiments. Readers interested in a detailed survey on AEs can refer to Yuan *et al.* [38].

Fast Gradient Sign Method (FGSM) was introduced by Goodfellow *et al.* [11] to allow the generation of AEs much faster than L-BFGS. It uses the concept of backpropagation but updates the inputs instead of the parameters. Thus, the sign of the gradient of the loss function with respect to the inputs is used to guide the perturbation ϵ (positive or negative), as shown in Eq. 3.

$$x' = x + \epsilon \cdot \text{sign}(\nabla J_\theta(x, l)) \tag{3}$$

Basic Iterative Method (BIM) was introduced by Kurakin *et al.* [16] and consists of applying FGSM in many iterations with a small perturbation magnitude. The advantage of BIM is that it adapts the perturbation to each iteration, the more iterations it does, the finer the perturbation is. In addition, BIM applies a clipping method, shown in Eq. 4, for every iteration to avoid getting feature values out of the interval I (considered [0, 1] in the equation).

$$\text{Clip}_{x,\xi}\{x'\} = \min\{1, x + \xi, \max\{0, x - \epsilon, x'\}\} \tag{4}$$

DeepFool was introduced by Moosavi-Dezfooli *et al.* [21]. This method looks for the closest distance from a normal example to the classification boundary it must cross to be misclassified. This distance is the perturbation applied to the example. Since it only looks for the closest distance to a different class, no matter which class, this attack is untargeted. This method originally optimizes the L_2 norm since it uses the Euclidean distance. The author overcame the obstacle of non-linearity in high dimensionality by using an iterative attack with linear approximation. In the case of binary differentiable classifiers, the perturbation

is approximated attractively by considering f linear around x_i. The minimal perturbation is then computed by Eq. 5. Despite its efficiency, DeepFool provides only a coarse approximation of the optimal perturbation vectors.

$$\text{argmin}_{\eta_i} \|\eta_i\|_2 \text{ such that } f(x_i) + \nabla f(x_i)^T \cdot \eta_i = 0 \tag{5}$$

Jacobian-based Saliency Map Attack (JSMA) was introduced by Papernot *et al.* [24]. Unlike previous methods, this method tries to minimize the number of perturbed features in order to create AEs with minimal L_0 norm. It starts with an empty set of features and chooses a new feature to perturb in each iteration. It iterates and adds perturbation until the example becomes adversarial or until it reaches another stop criterion. JSMA starts by computing the Jacobian matrix shown in Eq. 6. It is the matrix of the derivatives of each output logit with respect to each feature.

$$J_F(x) = \frac{\partial F(x)}{\partial x} = \left[\frac{\partial F_j(x)}{\partial x_i}\right]_{i \times j} \tag{6}$$

The Jacobian matrix estimates the contribution of each feature to each class. In order to prioritize the most salient attributes, a saliency map is built on the basis of the Jacobian matrix. The attribute with the highest saliency value for the targeted class is chosen to be perturbed in the current iteration.

Carlini&Wagner's attack (C&W) was introduced in Carlini and Wagner [6] as an efficient method to defeat existing defense techniques. The authors first reformulated Eq. 1 as an appropriate optimization instance.

$$\text{minimize } \|\eta\|_p + c \cdot g(x + \eta) \text{ such that } x + \eta \in I^n \tag{7}$$

In Eq. 7, g is an objective function such that $f(x + \eta) = l'$ if and only if $g(x + \eta) \leq 0$. Thus, the two constraints become a single term to minimize. A positive constant c is chosen by binary search to scale the minimization problem. In [6] three methods are introduced for the optimization of each of the L_0, L_2 and L_∞ distance norms:

- *L_2-attack* optimizes Eq. 7 with $p = 2$ using an optimization function to find AEs. It is the main method of the Carlini&Wagner attack, the other methods are based on this one.
- *L_∞-attacks* is an iterative attack, because the L_∞ distance norm is not fully differentiable, and thus optimization algorithms are not efficient. The first term of Eq. 7 is replaced by a new penalty that estimates the L_∞ norm.
- *L_0-attacks* is also iterative since the L_0 norm is not differentiable. In each iteration it applies the *L_2-attack*, it identifies the feature that contributes the least to the AEs using the gradient of the objective function and it fixes its value. The algorithm stops when the remaining subset of features is insufficient to construct AEs.

3 Literature Review

While often chiefly presented as a challenge for AI, AdvML has rapidly attracted the attention of the security research community. This is particularly evident when we analyze the extensive literature devoted to attacking techniques used to evade ML-based intrusion detection and malware detection models [19]. In this section, we review the research initiatives applying pioneer approaches to network intrusion detection.

Rigaki and Elragal [28] first explored the applicability of AEs on deep learning based intrusion detection models and their transferability to other machine learning models. Their experiments were performed using FGSM [11] and JSMA [24]. Wang [35] extended their work by testing DeepFool [21] and the three C&W attacks [6]. The author also discussed the contribution of each feature to the AEs and gave some guidelines on how these features could be manipulated by an adversary. Warzyński and Kołaczek [36] has also used FGSM [11] and successfully misclassified all attack samples as normal traffic. However, the attack parameters and the distance norms have not been reported.

Unlike previous works, Yang *et al.* [37] assumed a black-box attack scenario where the adversary only knows the output of the model (label or confidence). Three different black-box algorithms were evaluated: Transferring AEs generated on a substitute model using C&W [6], Zeroth Order Optimization (ZOO) [7] and Generative Adversarial Nets (GANs) [10]. Lin *et al.* [17] introduced IDSGAN, a framework based on GANs to generate AEs that can deceive a black-box intrusion detection system.

It is worth noticing that the experiments of all the previously mentioned works used a Multi-Layer Perceptron (MLP) neural network trained on the NSL-KDD dataset [33].

Martin *et al.* [18] applied the main attack methods to six different classifiers. They used NSL-KDD [33] and CICIDS2017 [31], a more recent dataset. They showed the robustness of different models before and after re-training them with AEs. Peng *et al.* [26] proposed an improved boundary-based method to craft AEs for DoS attacks, they also used CICIDS2017 [31].

Ibitoye *et al.* [12] compared the performance of Self-normalizing Neural Networks (SNNs) [14] with traditional Feed-forward Neural Networks (FNNs) for intrusion detection on the BoT-IoT dataset [15]. Their results show that FNNs outperform SNNs based on multiple performance metrics, while SNNs demonstrate better resilience against AEs. AbouKhamis *et al.* [1] used a min-max (or saddle-point) approach to train a model against AEs generated using variants of FGSM on the NSW-NB 15 dataset [22]. Principal Component Analysis (PCA) was applied to the dataset to evaluate its impact on the robustness of the model. Clements *et al.* [8] were able to efficiently fool an intrusion detection model by modifying 1.38 features on average. Alhajjar *et al.* [2] explored the use of evolutionary computation and GANs to generate AEs against network intrusion detection models. Piplai *et al.* [27] showed that even intrusion detection models trained with AEs can still be fooled.

Moisejevs [20] proposed a survey on adversarial attacks and defenses in intrusion detection, and Martins *et al.* [19] provided a systematic review on adversarial machine learning applied to intrusion and malware scenarios.

Despite the large number of works addressing adversarial attacks against intrusion detection, little attention was paid to the consistency of the generated AEs. In fact, even if these attacks can fool detection models, they do not represent a real threat if they cannot be implemented. The work reported in this article tries to provide a deeper analysis of the AEs to evaluate whether they can practically lead to the implementation of end-to-end network attacks. As far as we know, no other research initiative presents a such deep analysis to derive comprehensive validity criteria for adversarial attacks (cf. Sect. 6).

4 Experimentation Approach and Settings

In order to evaluate the impact of different adversarial attacks and the consistency of the generated AEs in intrusion detection, we set up a methodical experimentation approach. In this section, we describe our approach, starting from the choice of the dataset and the pre-processing techniques applied to it. Then we present the target ML model, discuss its design and its training. We finally introduce the AEs generation methods and their parameters.

4.1 Dataset and Pre-processing

With all the attention paid to intrusion detection in recent years, several interesting datasets have emerged. Ring *et al.* [29] presented a detailed survey of network intrusion detection datasets, they evaluated 34 datasets based on 15 properties they identified. In order to allow proper comparison with related works, all our experiments are performed using the NSL-KDD dataset [33]. Indeed, despite some drawbacks like its age, NSL-KDD remains the most widely used dataset in the intrusion detection literature.

In terms of pre-processing, we use One-Hot-Encoding to transform categorical features into a vector of binary features. For instance, in NSL-KDD, the `Protocol-type` feature can take three values: `TCP`, `UDP` and `ICMP`. When applying One-Hot-Encoding, this feature is represented by three different binary features and its values can be : $(1, 0, 0)$, $(0, 1, 0)$ or $(0, 0, 1)$. Only one binary feature can hold the value 1 since the instance belongs to a single category. One-Hot-Encoding pre-processing is paramount, particularly for neural network models, as they require numerical features. By applying it, the features count of our dataset rose from 41 to 120.

In addition, we removed the 20th feature `Num-out-bound-cmds` that only held the value 0. Min-Max normalization was also used to scale the values in the range [0, 1] to prevent features with large value ranges from influencing the classification.

Since the main concern of our study is evasion attacks against intrusion detection (classifying attacks as normal traffic), for our experiments, we only

consider the attack samples in the test set of NSL-KDD. Also, the dataset has been processed in order to regroup all the attack types into a single label. The classification will only be between two classes: normal and malicious (binary classification problem), which makes the targeted and untargeted attacks equivalent in our scenario.

4.2 Target Model Design and Training

Similarly to most of the previous works, the target model used in our experiments is a Multi-Layer Perceptron (MLP). It has 2 hidden layers of 256 neurons and a Softmax output layer with 2 neurons (Since we have a binary classification problem). The neurons of the hidden layers use the Rectified Linear Unit activation function (ReLU). The loss is computed using the Cross-Entropy Loss function. The model is trained for 1000 epochs using the Adam optimizer to adjust the parameters with a learning rate of 0.001. The model is made as simple and as close as possible to the models used in similar work in order to allow realistic comparisons. Thus, no regularization has been applied to avoid introducing any bias. Neural networks in our experiments are implemented using the open-source machine learning library Pytorch [25] on the programming platform Google Colaboratory.

4.3 Adversarial Attacks Models

In our experiments, the attacks are implemented using the open-source library Adversarial Robustness Toolbox (ART) [23]. The L_0-*attack* of Carlini&Wagner was not available, so we undertook its implementation to enrich ART. The parameters used for each attack are described below. Default parameters are preferred and no clipping was applied, since only few studies specify the parameters used in their experiments. The complete implementation of our experiments can be found on: https://github.com/mamerzouk/adversarial_analysis.

Fast Gradient Sign Method. For our experiments, we apply FGSM as defined in [11]. The gradient of the loss, with respect to the original class, is added to the examples, which makes it untargeted. The perturbation is applied in one single step (no iterations). The maximum perturbation magnitude ϵ is set to 0.1 and the batch size is set to 128.

Basic Iterative Method. For our experiments, we apply BIM with the same parameters as FGSM. We do not specify a target, the attack is thus untargeted. We set the number of iterations to 100 and the magnitude of the perturbation for each iteration to 0.001. This way, the maximum magnitude of the perturbation cannot exceed 0.1. We also set the batch size to 128.

DeepFool: In our experiment, we use DeepFool with a magnitude of 10^{-6} over 100 iterations and a batch size of 128. DeepFool is untargeted by definition and optimizes the L_2 norm.

Carlini&Wagner: The C&W attacks were applied in an untargeted way and with no minimum confidence imposed. The learning rate for the optimization algorithm was set to 0.01, and the batch size was set to 128. The rest of the parameters are kept in the ART default values. Since ART did not contain an implementation of Carlini&Wagner L_0-*attack*, we implemented this attack and made the code available in the experiment notebook.

Jacobian-Based Saliency Map Attack: We allow JSMA to perturb 100% of the features. We apply a perturbation of 0.1 in each iteration. The batch size is set to 128. Since JSMA is a targeted attack, if no target is specified, the implementation of ART randomly chooses a target from the false classes.

5 Evaluation of the Perturbation Potential

In this section, we present the results of our experiments and evaluate the generated AEs. Table 1 shows the accuracy of the model on the AEs generated by each method, along with the mean and maximum of each distance metric.

Table 1. Detection rate and distance metrics of different methods.

Methods	Detection	L_0 norm		L_2 norm		L_∞ norm	
		Mean	Max	Mean	Max	Mean	Max
Clean	75.1188%	0	0	0	0	0	0
FGSM	24.8811%	121	121	1.2099	1.2099	0.1	0.1
BIM	24.8811%	120.9543	121	0.9936	1.1578	0.1	0.1
DeepFool	25.1305%	120.9979	121	0.0177	0.1792	0.0469	0.1772
C&W L_2	22.7382%	13.8185	22	1.1977	7.2848	0.5078	1.4739
C&W L_∞	28.1306%	13.0478	43	0.5832	3.0571	0.2138	0.3
C&W L_0	24.1175%	3.7126	21	2.5272	22.1609	0.9099	2.4803
JSMA	24.8811%	2.0804	4	0.075	0.5	0.1729	0.5

We observe in Table 1 that before perturbing the data, the trained model achieved 75.11% detection rate on attack samples. These results are consistent with state-of-the-art performance on NSL-KDD. More details on the performance of the model can be found in the publicly available notebook.

Table 1 also shows that all the attacks had an impact on the detection rate of the model. Almost all of them considerably decreased the accuracy to around 24%, which represents a 68% decrease.

The similarity in the degradation caused by different methods allows an unbiased evaluation since the differences in the distance metrics are highlighted. These metrics demonstrate the various behavior of each method concerning the perturbation. Thus, we can understand how the attacks perturb the data differently to achieve, approximately, the same result. In the following subsections, we

Table 2. Samples of feature values from AEs of different methods.

Methods	`IRC`	`Telnet`	`Logged-in`	`Same-srv-rate`
FGSM	−0.1	0.1	−0.1	1.1
BIM	0.1	−0.1	−0.1	1.1
DeepFool	−0.0172	−0.0051	−0.0034	−0.0007
C&W L_2	0	1.4425	1.8378	2.3132
C&W L_∞	1.28	0	0.7274	1.3
C&W L_0	0	1.8155	0.8127	2.0155
JSMA	0.5	1	0	0

analyze the results of each method, and we examine the consistency of generated adversarial examples.

5.1 Fast Gradient Sign Method

We observe in Table 1 that FGSM has an important impact on the detection rate of the model, it decreases the detection rate to 24.88%. Among all the experimented algorithms, FGSM was the fastest. It has the lowest maximum L_∞ distance, which is the same as the mean L_∞ distance. This absence of variance is due to the fact that FGSM perturbs with the same amount all the features of all the examples. The objective is to spread the perturbation on the whole feature space with minimal perturbation magnitude (slightly perturb all the features instead of heavily perturb few features).

However, this method leads to indiscriminate perturbation of all the features. The mean and maximum of the L_0 norm, which refers to the number of perturbed features, is equal to the total number of features. This is consistent with the results of the Fig. 1, a heat map of the percentage of AEs perturbing each feature, that shows that all the features are perturbed in 100% of the AEs generated by FGSM.

This property of FGSM might be problematic for binary features: Since the perturbation applied is always equal to 0.1, it cannot change the value of a binary feature from one state to the other. For example, Table 2 shows an adversarial example generated by FGSM that puts the value of the binary feature `Telnet` to 0.1. This value invalidates the data sample, making it not practically possible to implement. This observation is valid for 100% of the AEs generated by FGSM, as shown in the Table 3.

Categorical features are also impacted by FGSM: Using One-Hot-Encoding transformed every categorical feature into multiple binary features. Only one of the binary features generated from the same categorical feature can hold the value 1, all the others must hold the value 0. However, FGSM perturbs all these binary features, which consequently activates multiple categories at the same time. We can see in Table 2 that the features `IRC` and `Telnet` which are derived from the category `Service` are both perturbed by FGSM. Since an

instance cannot belong to multiple categories, this perturbation invalidates the data sample. Table 3 shows that this observation is also valid for all the AEs generated by FGSM.

FGSM perturbs all the data samples following the sign of the gradient. This perturbation does not consider the definition domain of the feature. Thus, without a clipping function, the perturbation might put the value of a feature below its minimum or above its maximum. The example shown in Table 2 puts the value of `Same-srv-rate`, which is the proportion of connection to the same service among the connection aggregated in `count`, to 1.1. This value is not possible since the maximum proportion is 1. We can also see that `Logged-in` has a negative value -0.1. This value has no interpretation in a real network, so this adversarial example cannot be implemented. As well as all the other examples generated by FGSM according to Table 3.

FGSM was designed to generate AEs very quickly. It uses the simple idea of propagating the gradient of the loss all the way back to the inputs. This method is useful for adversarial training [11] since it allows the fast generation of AEs to re-train the model. However, it spreads the perturbation on all the features to minimise the L_∞ norm. This might be useful for unstructured data like images where features (pixels) do not hold a semantic value. But in the case of heavily structured data like network records, FGSM generates inconsistent values and breaks the semantic links between the features.

5.2 Basic Iterative Method (BIM)

As shown in Table 1, BIM has the same impact as FGSM with slightly better mean distance norms. The maximums are the same, except the L_2 norm which has a smaller maximum for BIM. The difference between the two can be explained by the finer optimization method of BIM that applies small FGSM steps in each iteration. This leads to smaller norm distances.

However, BIM perturbs the features the same way FGSM does. It also inherits all its disadvantages. Table 2 shows that AEs generated by BIM share the same properties as FGSM. Without clipping, the values of the features get out of their definition domain, as `Telnet`, it puts non-binary values on binary features like `Logged-in` and it activates multiple categories of `Service`. These criteria are present in 100% of the AEs generated by BIM, as shown in Table 3, and are sufficient to invalidate them.

5.3 DeepFool

Table 1 shows that DeepFool performs almost as well as the other methods. Since the objective of DeepFool is to optimize the L_2 norm, it has the smallest mean Euclidean distance. It is also noteworthy that DeepFool shows the best mean L_∞ norm and a slightly larger maximum L_∞ than FGSM and BIM.

Just like FGSM or BIM, the mean L_0 norm is almost equal to the total number of features. This demonstrates that DeepFool perturbs all the features of practically all instances. Figure 1 supports the results of the L_0 norm. It shows

Table 3. Proportion of invalidation criteria in AEs of different methods.

Methods	Out-of-range values	Non-binary values	Multiple categories
Clean	0%	0%	0%
FGSM	100%	100%	100%
BIM	100%	100%	100%
DeepFool	100%	100%	100%
C&W L_2	94.7089%	99.9688%	0%
C&W L_∞	80.5345%	90.0802%	0.8493%
C&W L_0	63.5393%	54.0559%	0.1636%
JSMA	0.0155%	67.2952%	67.2796%

indeed that the vast majority of features are perturbed on more than 99% of instances.

Despite its good results, DeepFool stays a method that only focuses on the Euclidean distance. It does not optimize the number of perturbed features; rather, it perturbs a large number of features in practically all the instances to minimize the L_2 norm.

Table 2 shows examples where DeepFool generates non-binary values on binary features like `Logged-in`. It activates multiple categories of `Service` as `IRC` and `Telnet`. It also generates out-of-range values, as for `Same-server-rate`, which is a proportion and cannot be negative. These properties are found in 100% of generated AEs, according to Table 3. Besides, the simultaneous perturbation of all the features may damage the semantic links between them. In the case of network data, this leads to inconsistent samples that cannot be implemented.

5.4 Carlini and Wagner

***L_2*-Attack:** As shown in Table 1, Carlini&Wagner *L_2-attack* reduces the detection rate of the model to 22.73%, which is the lowest detection rate recorded. Though it is supposed to optimize the L_2 norm, it has one of the highest mean and maximum Euclidean distance. The L_∞ norm is also high compared to previous methods. However, the *L_2-attack* of Carlini&Wagner does not perturb all the features, the L_0 norm has a mean of 13.81 features and a maximum of 22 features.

From the samples shown in Table 2, we can see how the *L_2-attack* introduces non-binary values like 1.8378 for `Logged-in`. Because of the large magnitude of the perturbation, some features are pushed out of their definition range. For example `Same-server-rate` is pushed to 2.3132 when it should not exceed 1. However, unlike other methods, Carlini&Wagner does not activate multiple categories of the same categorical feature on any data sample. Despite this interesting result, the two first properties make the AEs generated by the *L_2-attack* not applicable to real-world network traffic.

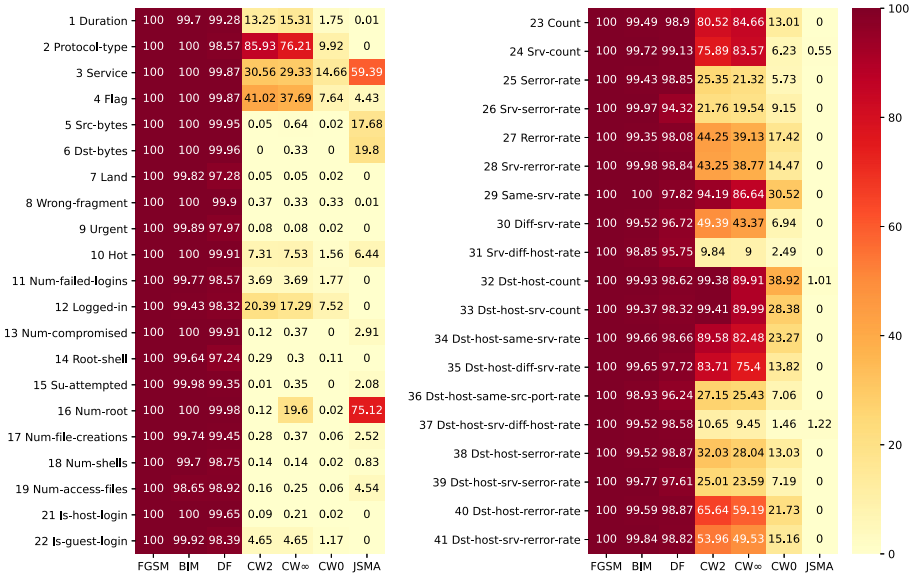

Fig. 1. Heat map of the proportion of AEs perturbing each feature

***L_∞*-Attack:** Our results show that Carlini&Wagner L_∞-*attack*, with the used parameters, was the less efficient method on NSL-KDD. It decreased the accuracy to 28.13%. Even though this method is supposed to optimize the L_∞ norm, its mean and maximum perturbation values are larger than FGSM, BIM or DeepFool. However, the AEs generated by this method showed relatively small L_0 norm values. Only 12.71 features were perturbed in average with a maximum of 43 perturbed features.

Despite all this, the L_∞-*Attack* of Carlini&Wagner presents insufficiency that prevents its use in network data. First, its lower impact on the accuracy reduces the number of feasible adversarial attacks. Even if the perturbation is not spread on a large number of features, Table 2 shows that this attack perturbs some binary features with a non-binary value, it is the case for `Logged-in`. It also puts out-of-range values on features like `Same-server-rate`. But it does not activate multiple categories on more then 0.8% of its AEs.

***L_0*-Attack:** As shown in Table 1, the L_0-*attack* of Carlini&Wagner had strong impact on the detection rate by only perturbing 3.7 features on average and a maximum of 21 features. However, these results are explained by the L_2 and L_∞ norms that are excessively large, by far the highest among all the methods. The Euclidean distance reached 22.16, and the maximum perturbation was up to 2.48 and 0.9 on average. These metrics are extremely high and make the L_0-*attack* of Carlini&Wagner unsuited for network data.

We can see in Table 2 that even if it only perturbs a few features, there is still inconsistency in the data. Binary features like `Logged-in` hold non-binary values.

Also, some features reach very large values, like `Telnet` that was set to 1.81 when its maximum should be 1. However, our results showed that Carlini&Wagner L_0-*attack*, just like other Carlini&Wagner attacks, almost never perturbs multiple categories of the same categorical feature. It focuses its perturbation on the actual category of the instance. This result holds true for the three categorical features `Protocol-type`, `Service` and `Flag`.

5.5 Jacobian-Based Saliency Map Attack

The Jacobian-based Saliency Map Attacks decreased the accuracy to 24.88%, which is the same score as FGSM and BIM. This finding was observed in several executions.

JSMA showed the best L_0, it only perturbed 2.08 features on average and a maximum of 4 features. The average Euclidean distance was around 0.07, and the maximum was 0.5. These are the second-best L_2 norms after DeepFool. The mean L_∞ norm was better than all Carlini&Wagner attacks but the maximum L_∞ reached 0.5, the third-highest after C&W L_0-*attack* and C&W L_2-*attack*.

JSMA certainly shows the most interesting results for a network data application. Unfortunately, Table 2 shows that even AEs generated by JSMA have inconsistency problems. Binary features like `Telnet` are perturbed with non-binary values. Multiple categories of the same categorical feature are activated, is the case for `IRC` and `Telnet`. These two criteria were found in, respectively, 67.29% and 67.27% of the AEs generated by JSMA. Thus, many examples may be disqualified. However, only 0.01% of the examples have out-of-range values, it can be explained by Fig. 1 that shows that JSMA focuses its perturbation on features like `Num-root` or `Src-bytes` and `Dst-bytes` that can reach high values.

6 Criteria for Valid End-To-End Adversarial Attacks

The results presented previously demonstrate the high perturbation potential of adversarial examples on ML-based intrusion detection systems. However, when we perform an in-depth analysis of the data samples generated by the different methods, one can legitimately question the practicality of these samples when it comes to performing real end-to-end cyberattacks. Our results showed that a large portion of the perturbation that was applied to network traffic features invalidate the original network session, making the derived attack hard, if not impossible, to execute in real environments. We summarize below the main invalidation criteria we have identified in our research. This is a non-exhaustive list that can be extended with other criteria.

Non-binary Values: Binary features indicate the presence or the absence of a property in the data, they can only hold the values 0 or 1. Since these features are often important to identify intrusion, AEs generation methods focus on perturbing them. Thus introducing a value between 0 and 1. These values are inconsistent for binary features and cannot be implemented in real network traffic. We have seen examples where the binary feature `Logged-in` was set to 0.72.

Multiple Categories Membership: Categorical features have been converted into binary features to make it possible to use them as inputs for neural networks. One-Hot-Encoding was used to create a binary feature for each instance of the categorical feature. Thus, only one instance can hold the value 1, while all the others must be set to 0. Generation methods often perturb these features by activating multiple categories. Which, even if it is recognized as an attack by the neural network, cannot be implemented in real network traffic. We gave as an example the feature `Service` which cannot be `IRC` and `Telnet` at the same time.

Out-of-Range Values: Every attribute of the network traffic has a limited range of values it can take. But since generation methods apply the perturbation until they reach the adversarial boundary, some features might be pushed out of their definition interval, which generates inconsistent values that cannot be

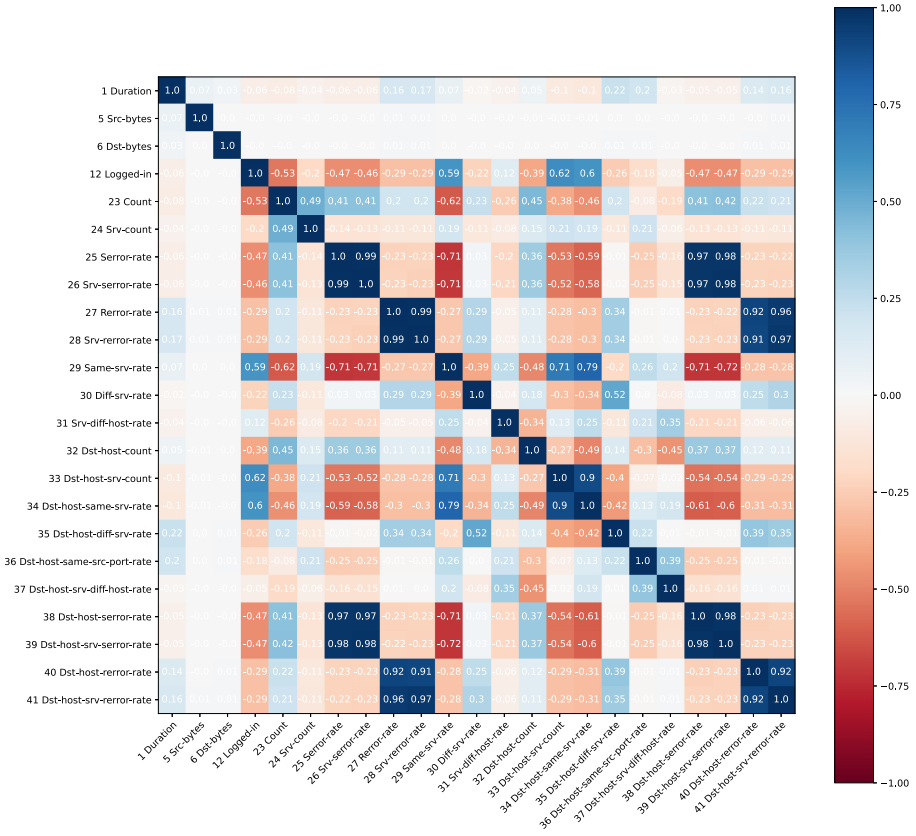

Fig. 2. Heat map of the correlation matrix of NSL-KDD numerical features. (Color figure online)

implemented. As we saw in Table 2, `Same-server-rate` was set to 2.31 by C&W L_2-*attack* when it is a proportion that should not exceed 1. In Table 3, we consider the minimum and maximum values found in the testing set to compute the proportion of AEs containing out-of-range values.

Semantic Links: In contrast to unstructured data like images, network data hold semantic links between features. These links create dependencies that must be kept to ensure the consistency of the traffic. The generation methods do not consider these semantic links and apply an arbitrary perturbation that often breaks them and generates incoherent samples. Unlike other invalidation criteria that are present in Table 3, semantic links are hard to identify. This is due to the fact that there are no explicit rules to express these links. To better illustrate our findings, we computed the heat map of the correlation matrix between the numerical attributes of NSL-KDD illustrated in Fig. 2. The intensity of shades of blue (resp. red) indicates the level of positive (resp. negative) pairwise correlation between the attributes. For example, we can notice a strong positive correlation between `Dst-host-rerror-rate` and `Dst-host-srv-rerror-rate`, or between `Srv-serror-rate` and `Rerror-rate`.

7 Conclusion and Future Work

In this paper, we have discussed the applicability of adversarial examples in network intrusion detection. Through a literature review, we have noticed that little consideration was given to the validity of AEs with respect to network traffic structure and constraints.

We have filled that gap by analyzing AEs generated by state-of-the-art algorithms and identifying key criteria that invalidate them. These criteria include values outpacing the definition domain, assignment of non-binary values to binary features, belonging to multiple contradictory categories, and breaking semantic links between features.

Though the described criteria are sufficient to invalidate AEs, they do not guarantee their validity. Thus, future work should focus on a formal description of network constraints that must be fulfilled in order to validate an attack example. More recent datasets should be used in order to study the perturbation potential of adversarial attacks on different data types. Finally, the vulnerability of intrusion detection models should be proven on real networks with end-to-end attack scenarios.

Acknowledgment. This research was partly funded by the European Union's Horizon 2020 research and innovation program under the Secure Collaborative Intelligent Industrial Automation (SeCoIIA) project, grant agreement No 871967 and IRT SystemX projects (Exploratory research and PFS).

References

1. Abou Khamis, R., Shafiq, O., Matrawy, A.: Investigating resistance of deep learning-based IDS against adversaries using min-max optimization. arXiv preprint:1910.14107 (2019)
2. Alhajjar, E., Maxwell, P., Bastian, N.D.: Adversarial machine learning in network intrusion detection systems. arXiv preprint:2004.11898 (2020)
3. Biggio, B., et al.: Evasion attacks against machine learning at test time. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013. LNCS (LNAI), vol. 8190, pp. 387–402. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40994-3_25
4. Biggio, B., Nelson, B., Laskov, P.: Poisoning attacks against support vector machines. arXiv preprint:1206.6389 (2012)
5. Biggio, B., Roli, F.: Wild patterns: Ten years after the rise of adversarial machine learning. Pattern Recognition (2018)
6. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy. IEEE (2017)
7. Chen, P.Y., Zhang, H., Sharma, Y., Yi, J., Hsieh, C.J.: Zoo: zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In: 10th ACM Workshop on Artificial Intelligence and Security (2017)
8. Clements, J., Yang, Y., Sharma, A., Hu, H., Lao, Y.: Rallying adversarial techniques against deep learning for network security. arXiv preprint:1903.11688 (2019)
9. Fredrikson, M., Jha, S., Ristenpart, T.: Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (2015)
10. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems (2014)
11. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint:1412.6572 (2014)
12. Ibitoye, O., Shafiq, O., Matrawy, A.: Analyzing adversarial attacks against deep learning for intrusion detection in IoT networks. In: IEEE Global Communications Conference (GLOBECOM) (2019)
13. Jagielski, M., Carlini, N., Berthelot, D., Kurakin, A., Papernot, N.: High accuracy and high fidelity extraction of neural networks. In: 29th USENIX Security Symposium (2020)
14. Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S.: Self-normalizing neural networks. In: Advances in Neural Information Processing Systems (2017)
15. Koroniotis, N., Moustafa, N., Sitnikova, E., Turnbull, B.: Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset. arXiv preprint:1811.00701 (2018)
16. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial machine learning at scale. arXiv preprint:1611.01236 (2016)
17. Lin, Z., Shi, Y., Xue, Z.: IDSGAN: generative adversarial networks for attack generation against intrusion detection. arXiv preprint:1809.02077 (2018)
18. Martins, N., Cruz, J.M., Cruz, T., Abreu, P.H.: Analyzing the footprint of classifiers in adversarial denial of service contexts. In: EPIA Conference on Artificial Intelligence (2019)
19. Martins, N., Cruz, J.M., Cruz, T., Abreu, P.H.: Adversarial machine learning applied to intrusion and malware scenarios: a systematic review. IEEE Access **8**, 35403–35419 (2020)

20. Moisejevs, I.: Adversarial attacks and defenses in intrusion detection systems: A survey. Int. J. Artif. Intell. Expert Syst. (IJAE) **8**(3), 44–62 (2019)
21. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: DeepFool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016)
22. Moustafa, N., Slay, J.: UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: Military Communications and Information Systems Conference (MilCIS) (2015)
23. Nicolae, M.I., et al.: Adversarial robustness toolbox v1.2.0. arXiv preprint:1807.01069 (2018)
24. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: IEEE European Symposium on Security and Privacy (EuroS&P) (2016)
25. Paszke, A., et al.: Pytorch: an imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems (2019)
26. Peng, X., Huang, W., Shi, Z.: Adversarial attack against dos intrusion detection: an improved boundary-based method. In: IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI) (2019)
27. Piplai, A., Chukkapalli, S.S.L., Joshi, A.: Nattack! adversarial attacks to bypass a gan based classifier trained to detect network intrusion. arXiv preprint:2002.08527 (2020)
28. Rigaki, M., Elragal, A.: Adversarial deep learning against intrusion detection classifiers. In: NATO IST-152 Workshop on Intelligent Autonomous Agents for Cyber Defence and Resilience (2017)
29. Ring, M., Wunderlich, S., Scheuring, D., Landes, D., Hotho, A.: A survey of network-based intrusion detection data sets. Comput. Secur. **86**, 147–167 (2019)
30. Rosenberg, I., Shabtai, A., Elovici, Y., Rokach, L.: Adversarial learning in the cyber security domain. arXiv preprint:2007.02407 (2020)
31. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: 4th International Conference on Information Systems Security and Privacy (2018)
32. Szegedy, C., et al.: Intriguing properties of neural networks. arXiv preprint:1312.6199 (2013)
33. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the KDD cup 99 data set. In: IEEE Symposium on Computational Intelligence for Security and Defense Applications (2009)
34. Vorobeychik, Y., Kantarcioglu, M.: Adversarial machine learning. Synthesis Lectures on Artificial Intelligence and Machine Learning (2018)
35. Wang, Z.: Deep learning-based intrusion detection with adversaries. IEEE Access (2018)
36. Warzyński, A., Kołaczek, G.: Intrusion detection systems vulnerability on adversarial examples. In: Innovations in Intelligent Systems and Applications (2018)
37. Yang, K., Liu, J., Zhang, C., Fang, Y.: Adversarial examples against the deep learning based network intrusion detection systems. In: IEEE Military Communications Conference (MILCOM) (2018)
38. Yuan, X., He, P., Zhu, Q., Li, X.: Adversarial examples: attacks and defenses for deep learning. IEEE Trans. Neural Netw. Learn. Syst. **30**, 2805–2824 (2019)