Intrinsic weaknesses of IDSs to malicious adversarial attacks and their mitigation

Hassan Chaitou, Thomas Robert, Jean Leneutre, Laurent Pautet

Intrinsic weaknesses of IDSs to malicious adversarial attacks and their mitigation

Hassan Chaitou, Thomas Robert, Jean Leneutre, and Laurent Pautet

LTCI, Télécom Paris, Institut Polytechnique de Paris, Paris, France
{hassan.chaitou, thomas.robert, jean.leneutre, laurent.pautet}@telecom-paris.fr

Abstract. Intrusion Detection Systems (IDS) are essential tools to protect network security from malicious traffic. IDS have recently made significant advancements in their detection capabilities through deep learning algorithms compared to conventional approaches. However, these algorithms are vulnerable to meta-attacks, also known as adversarial evasion attacks, which are attacks that improve already existing attacks, specifically their ability to evade detection. Deep learning-based IDS, in particular, are particularly susceptible to adversarial evasion attacks that use Generative Adversarial Networks (GAN). Nonetheless, well-known strategies have been proposed to cope with this threat. However, these countermeasures lack robustness and predictability, and their performance can be either remarkable or poor. Such robustness issues have been identified even without adversarial evasion attacks, and mitigation strategies have been provided. This paper identifies and formalizes threats to the robustness of IDSs against adversarial evasion attacks. These threats are enabled by flaws in the dataset's structure and content rather than its representativeness. In addition, we propose a method for enhancing the performance of adversarial training by directing it to focus on the best evasion candidates samples within a dataset. We find that GAN adversarial attack evasion capabilities are significantly reduced when our method is used to strengthen the IDS.

Keywords: Adversarial machine learning · GAN · Intrusion Detection System · Sensitivity analysis.

1 Introduction

A network intrusion detection system (NIDS) plays an important role in protecting networks by monitoring the state of the network activity. A NIDS can be designed in two ways: signature-based IDS and machine learning (ML)-based IDS. In the first one, an expert identifies solutions to common problems by collecting a large database of signatures for well-known attacks, and then the IDS scans the network traffic to see whether it matches one of the attack signatures or not. This approach, however, has a number of downsides, including a large number of rules to handle, limited detection capabilities, and high maintenance costs. Therefore, many researchers have focused on designing IDSs that rely on machine learning techniques to address the above problems (ML-based IDS). It is generally accepted that such IDSs perform better in detecting variations of known attacks and, in some cases, unknown attacks.

We focus on IDSs that monitor network activity by observing packets, events, or connections. As a result, whenever such observations are made, the IDS is expected to determine whether it is an attack or normal activity (its class). Such IDS have a parametric model that must be optimized to predict the class of the observed activity with the highest attack detection rate and the lowest false alarm rate.

This optimization process is divided into three stages: collecting field data, extracting features, cleaning and preprocessing them to make them usable by the model, and iteratively modifying model parameters to improve detection and false positive rates. The collected field data should contain labelled examples of both attacks and normal activities in a low-level representation that combines scalar and categorical values. The

second step, network data analysis, consists of packet preparation and feature extraction up to the feature conversion to scalar types only (called preprocessing). Eventually, it converts the low-level representation of network activities into vectors of values known as "samples," a term borrowed from the general Machine Learning community. The last step is the actual training step, which aims to find the best model parameters for the highest detection rate and lowest false alarms based on the training data. Therefore, the quantity and quality of data used to train the parametric function are critical factors in determining IDS detection performance. Many works highlight the quality of datasets that may require to be sufficiently representative [13].
The studied system consists of two sides, the attacker and defender sides, with the attacker responsible for performing attacks and attempting to evade detection and the defender responsible for training and deploying an efficient IDS. In this context, an attacker can benefit from "evasion attack," a well-known approach to evade detection for IDSs. The evasion attack makes the above mentioned training pipeline vulnerable to adversarial samples. [30]. Crafting adversarial attack samples consists of transforming attacks that are unaware of an IDS being used into attacks specifically designed to avoid the IDS while having a malicious impact. Such transformations are possible using Generative Adversarial Networks [9] among other generative approaches. On the other hand, a defender employs countermeasures against evasion attacks mainly by injecting adversarial samples into the IDS's training dataset. Consequently, the training pipeline is extended by this training dataset enhancement. Interestingly, the literature shows that such a dataset enhancement is typically based on the same method, namely "adversarial training." However, despite the fact that the method appears to be similar at first glance, in practice, the result of this enhancement can vary a lot in terms of countermeasure performances. In the literature, it has been hypothesized that this variation can be partly explained by how the defender and attacker define and use the extent to which attack can be altered [4].
In this work, we reuse and extend the dataset issues of regular training pipelines to analyze adversarial training ones on well-understood models [10] and datasets. Moreover, it allows us to understand why some approaches perform perfectly on some datasets but not on others or even exhibit rather unpredictable behavior on the same dataset.
Processes and criteria for improving and assessing the quality of training pipelines and the datasets on which they rely have been defined, according to [8]. Three criteria can be retained: i) If one wants to predict their class accurately, they should avoid underrepresented types of activities, ii) Train distinct binary models to separate pairs of classes if possible (e.g., normal type vs. attack type, for each attack class), iii) Avoid dataset issues with the same sample associated with different classes.
One example of the point iii issue is storing the incorrect class in the collected data. However, a more complicated situation arises when the dataset appears to contain the same sample associated with two distinct classes. It would result in a dead end in terms of determining whether or not a sample is an attack without impairing false alarms or detection capabilities.
To the best of our knowledge, such criteria are not taken into account when expanding the training dataset with examples of evasion attacks. Therefore, the following are our research objectives:

Question 1 Can the dataset consistency issue intuitively introduced above be relevant as it is for adversarial training datasets? If not, why not, and what should be done about it?

Question 2 Can consistency issues for adversarial training datasets be responsible for IDS performance issues against evasion attacks (qualitatively)?

Question 3 What are the potential solutions to consistency issues on adversarial training datasets?

Question 4 To what extent, quantitatively, do these issues really pose threats to IDS performance without or with countermeasures applied?

Section 2 describes the system architectures as well as the IDS regular training pipelines. Section 3 explains and formalizes the notion of the adversarial neighborhood of attack. Section 4 formally defines the main

consequence of being able to generate adversarial samples that remain attacks without testing. Furthermore, it introduces the threats this situation entails and presents mitigation strategies. Section 5 presents the evaluation metrics besides the experimentation descriptions. Sections 6 and 7 present the experimental assessment of the threats we identified and the performance of the mitigation strategies.
This paper is an extension of the one published in the SECRYPT proceedings [5].
In [5], we only considered one adversarial neighborhood definition, assuming that a single perfect definition exists. This paper, on the other hand, assumes that each attack generator has its own definition of neighborhood. This is demonstrated by examining two datasets, the NSL-KDD, and the CIC-IDS2017, with three adversarial neighborhood definitions. Furthermore, this paper refined the measures on the best evasion attack candidate (BEAC) to demonstrate quantitatively that the BEAC set elements poses a severe threat to the IDS detection performance.

2 Systems under review and regular training pipelines

This section recalls the system architecture described in this paper as well as the main steps in the IDS training pipeline. It recalls the vocabulary and formalizes the concepts needed to describe IDS training pipeline and the consistency issue mentioned above in its training dataset.

2.1 Systems architecture

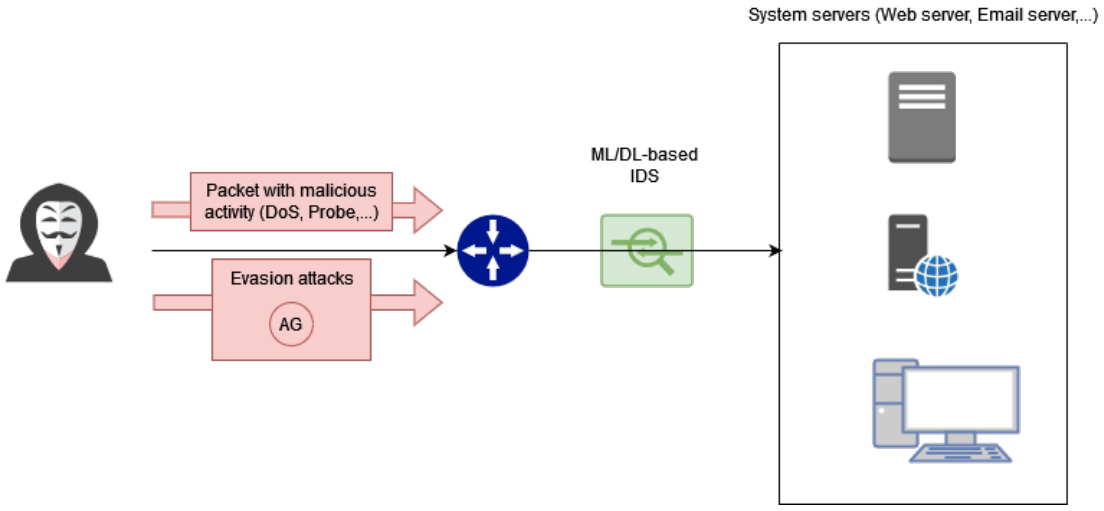

Fig. 1. Overview of the system architecture

As shown in figure 1, an attacker is interested in sending to the system packets with malicious payload to bypass the defense mechanism (e.g., IDS) and ensure that the malicious payload has the intended impact on the targeted system behind the IDS. However, when the defense mechanism detects the malicious activity, it drops the attacker's packets.

In this case, the attacker can launch a meta-attack, which consists in modifying the given malicious activity detected by the IDS into an activity that can bypass the defense system. The authors of [21] propose a meta-attack approach that aims to modify a malicious activity so that it is different to the IDS leading it to consider it as a normal activity. This meta-attack is exactly an evasion attack. In order to automate evasion attacks, an "attack generator" (AG) is defined. It is a function, denoted *ag* that should select for an attack the best changes to apply. The result of those changes on the activity is usually called a mutation.
The defender, on the other hand, relies on ML-based IDS to protect its network from packets with malicious payloads and potentially evasion capabilities. Set of examples of normal and attack activities contains usually no example of evasion. Then an IDS train with such data cannot a priori detect the attack on which the meta-attack, the evasion attack, has been applied. In this situation, the defender must also employ an *ag* to generate several mutations of the malicious activities in order to have examples of evasion attacks. This dataset enhancement aims to improve the efficiency of IDS detection against attacks without and with evasion capabilities without compromising other important IDS performance metrics. For instance, it should not reduce the availability of the normal activity.
In the next subsection, we formalize and explain the different procedures and phases of the training pipeline in detail.

2.2 Regular training pipeline in details

In order to explore the potential issues with the quality of datasets, we need to start by understanding the regular training pipeline for ML-based IDS.
The first step in the training pipeline is to collect the network raw data. The network raw data refers to data provided by network sensors, as depicted in Figure 2. The network sensors represent the observation capabilities of network activity from routers, firewalls or host machines. This data is organized and merged into observation units corresponding to an element of network activity, called a sample. An IDS incorporating a classifier for attack detection is fed with samples of data collected by sensors. As depicted in figure 2, the sample goes through various processes before being processed by a classifier that determines for each sample whether it belongs to a normal or attack class.

Definition 1 (Raw sample). *A* raw sample *is a tuple of n values respectively of types* T_1, ..., T_n. *The raw sample type* T_R *is* $T_R = T_1 \times ... \times T_n$.

The PCAP format is the type of raw sample usually used: it is detailed enough that the activity can even be replayed from the raw sample. Basically, raw samples are expected to be enough detailed to be able to decide whether a sample corresponds to normal activity or attack. Let L be the set of possible labels: $L = \{normal, attack\}$.

Definition 2 (Raw labeled sample and dataset). *A* raw labeled sample *is a couple* (x, y) *where* $x \in T_r$ *and* $y \in L$. *A* raw labeled dataset *is a set of raw labeled samples.*

The raw sample type often relies on non-numeric types to capture metadata about packets or application behaviors. Therefore, the values of raw samples can be of very diverse types (e.g., binary, categorical, numeric, strings). It is extremely difficult to feed a classifier with such data without first transforming all these types into scalar normalized values. This step is called pre-processing. Let call $prep$ the function that produces IDS scalar inputs from raw samples R. Therefore, $prep$ function takes elements of T_R and produces a vector of values q in $[0,1]$, we now use T_P such that $T_P = [0,1]^q$.

Definition 3 (Preprocessed samples and labeled dataset). *For any raw sample x, $prep(x)$ is a* preprocessed sample*, and for any raw labeled dataset D, $prep(D) = \{\{(prep(x_i), y_i) | (x_i, y_i) \in D\}\}$, is the corresponding* preprocessed labeled dataset.

To apply *prep* to a labeled dataset, one has to apply it to the raw sample portion of each labeled sample. In the Machine Learning community, a dimension of a sample is called a feature in the feature space. A labeled sample is said to be an *attack sample* if its label is *attack*, and a *normal sample* if it is *normal*. Each labeled dataset can be split in two subsets respectively called *normal traffic* and *attack traffic*.

Definition 4 (Normal and attack traffic of D). *Given a labeled dataset D (raw or preprocessed), the* normal traffic *of D denoted $N(D)$, and the* attack traffic *of D denoted $A(D)$ are defined as follow:*

$$N(D) = \{(x_i, y_i) | (x_i, y_i) \in D, y_i = normal\}$$
$$A(D) = \{(x_i, y_i) | (x_i, y_i) \in D, y_i = attack\}$$

A labeled dataset is necessary when training a classifier, or defining its parameters; such a dataset is called a training dataset.

As stated in section 1, the most important quality criteria are the absence of label issues on a sample and the balance in the proportion of sample types (i.e., to avoid under-representation of large sample classes).

Datasets may contain problematic samples due to sample collection problems or too high-level observations. Beyond wrongly labelled samples, we point out another type of labelling issues that affect also the IDS.

Definition 5 (Contradictory samples). *Two labelled samples (x_1, y_1) and (x_2, y_2) are* contradictory samples *if and only if (iff) $x_1 = x_2$ and $y_1 \neq y_2$.*

Such pairs of samples if present in a training dataset are problematic as they contradict each other.

Definition 6 (Contradictory set of D). *The contradictory set of D, denoted $CS(D)$ is the set of all contradictory samples contained in D.*

A classifier is a parametric model that must be configured, and the training process is responsible for determining the appropriate parameters. Usually it relies on minimizing the gap between the label produced by the model (predicted one) and the label stored in the dataset. Yet, in case of contradictory samples, one can consider that there is no good answer for such samples. Indeed, the class predicted by the IDS contradicts at least one of the samples. In such a case, the IDS is always partly wrong.
The regular training pipeline is vulnerable to adversarial evasion attacks. Therefore, the following section shows how malicious adversarial samples can evade detection and how enriching the dataset with adversarial samples affects the IDS training pipeline.

3 Detailed analysis of attack generators

This section explains how attack generators found in the literature appear at first sight to rely on the same principles. Then, it presents the internal details of their design and highlights two elements that are often redefined for each application and dataset. The first of these parameters is formalized with the notion of an attack sample's adversarial neighborhood. As the attack generator also relies deeply on a random seed, we explain how it can affect its performances. Finally, the section presents a list of such neighborhoods used in the security community.

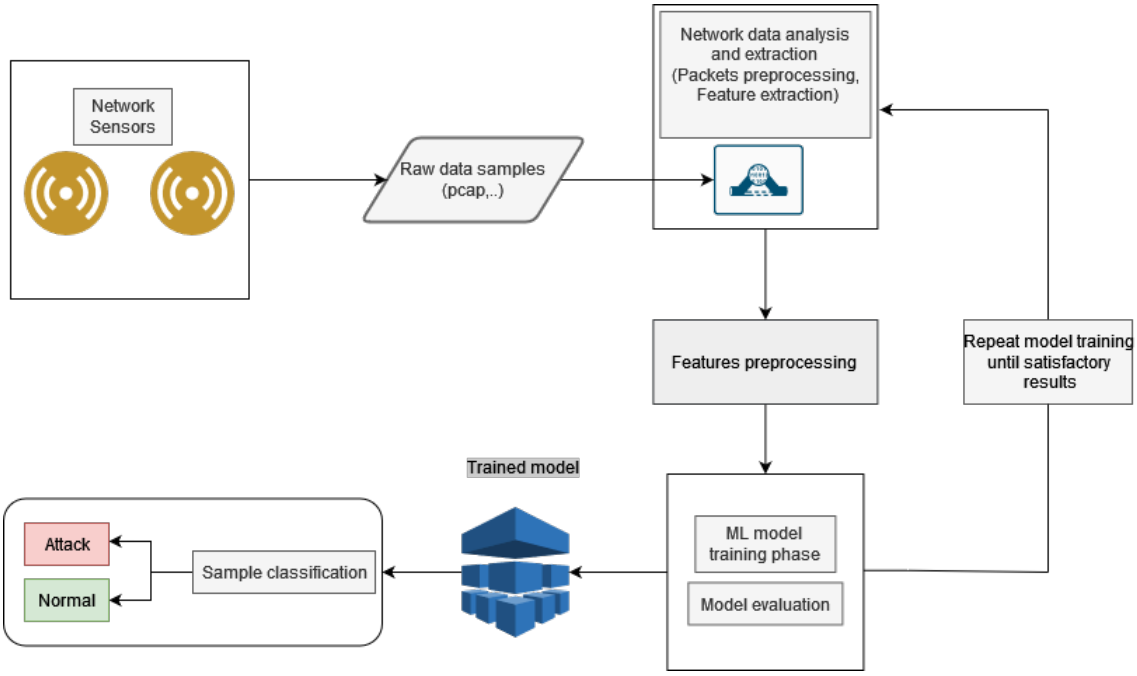

Fig. 2. Deployed architecture of classifier based IDS

3.1 Adversarial samples generator

Given a set a classifier *cla* that predicts classes from samples, let s be a sample for which the class sc has been correctly predicted by t. A successful adversarial sample s' derived from s is a sample obtained modifying s so that *ids* would not predict the class sc for s' despite both samples should be seen as "equivalent" (eg. in image processing, one criteria would be "a human do not see the change" or "a human interpret both images the same way"). This is the basics of adversarial samples theory. In order to ensure both samples are "equivalent", many criteria have been proposed.

The result of an evasion attack on sample is in practice an adversarial attack sample. Indeed, if the adversarial attack sample is successfully crafted, it will be not classified as an attack and thus evade the IDS. In the remainder of the paper, we reason about adversarial samples defined in T_p (the space of preprocessed samples) but only for attack samples.

A *malicious adversarial sample* (*MAdv*) is an adversarial sample derived only from attack samples. In the case of attack samples, the adversarial sample is considered "equivalent" if it yields the same consequences on the systems, ie. it has the same malicious impact. Malicious adversarial samples are basically a synonym of evasions attacks against IDSs built upon a classifier.

Let us now depict the usual generic approach followed to generate those samples. A function is used to represent this process. Its first input parameter is the sample to alter. For each input sample, many adversarial samples could be proposed. Then this function takes another input that helps exploring these alternatives. This second parameter is called a "noise" parameter as it has no particular meaning except to ensure that with the same function, one can obtain many adversarial samples for the same input sample.

Definition 7 (Adversarial sample generator). *An adversarial sample generator ag is a function with parameters (x,d) of type (T_p) (a sample) and $[0,1]^k$ (a value used to explore possible changes) and returns x' a sample (a member of T_p)*

One has to recall that this function is intended to be used only on samples that correspond to attacks (e.g. only samples that labels would be *attack*). Recall that T_p corresponds to vectors of scalars for which $+$ and $-$ have the usual meaning, then the perturbation introduced by ag on x with z is $ag(x,z)-x$.

This function is almost never used only once, it is used to produce most often huge amount of adversarial samples. These samples are either used as means to assess the likelihood of generating a successful malicious adversarial sample. It can also be used as means to improve the training pipeline of the IDSs. We will discuss it later. Such a function is often dedicated to certain kinds of attack and normal activities, e.g. a dataset and thus incidentally an IDS. Let us explains the constraints and best practices when generating such sets.

3.2 Adversarial sample generation and its usage

In this subsection section, we discuss what are the objectives behind producing those sets and their properties.

Adversarial attack samples could be required either for training IDSs or assessing their performances. In the second case, the IDSs are tested against a large number of adversarial samples to capture the likelihood of success of these meta-attacks. Yet, the problem is the following: the attack generator can be applied several times to the same attack, and nothing guarantees that it will necessarily produce distinct adversarial samples. Yet, most of the time set of samples are considered as approximation of the distribution of possible samples. Therefore, counting how many times such samples evade detection is sufficient to estimate the likelihood of undetected attacks.

In the case many adversarial samples are generated for the same sample, it might be important to recall (i.e. store) how much time the same adversarial sample is produced (possible). For this reason, we consider generating bags of adversarial samples instead of simply sets. Picking at random an element from the bag would mimic the distribution of generated adversarial samples. To differentiate bags from sets, bags would be denoted with brackets. $[[1,1,2,2,3]]$ is thus the bag that contains 5 integers, twice 1 and twice 2.

We introduce a notation to denote bags corresponding to i adversarial samples generated for each attack sample of a dataset. Whenever an adversarial sample is generated, the noise input is assumed to be drawn at random in the set of noise values. The distribution of noise values picked that way is assumed to follow a uniform distribution.

Let ag be an attack generator, D a dataset made only of attack samples.

Definition 8 (Bags of adversarial samples of D). *We denote $Uniform(i,ag,D)$ a bag defined as follow*

$$Uniform(i,ag,D) = [[ag(a,z_j)|a \in D,\ 1 \leq j \leq i,\ z_j \text{ uniform i.i.d. random variables over } [0,1]^k]]$$

Counting the occurrence of an element in such a bag for high values of i provide an approximation of the likelihood that this attack sample is actually produced if the noise element is picked actually at random. Notice that the set of possible images of ag for a fixed attack sample could be far smaller than the set of possible noise vectors (it is even expected).

This situation is illustrated in figure 3. For simplicity, we consider a simplified example of sample type with only three dimensions. The noise value is defined over $[0,1]^2$ and the attack generator ag_s is defined as follows:

$$ag_s((a_1,a_2,a_3),(z_1,z_2)) = \begin{cases} (0,a_2,a_3) & if \quad z_1 > 0.5 \\ (a_1,0,a_3) & if \quad z_2 < 0.5,\ z_1 \leq 0.5 \\ (a_1,a_2+z_1,a_3) & otherwise \end{cases}$$

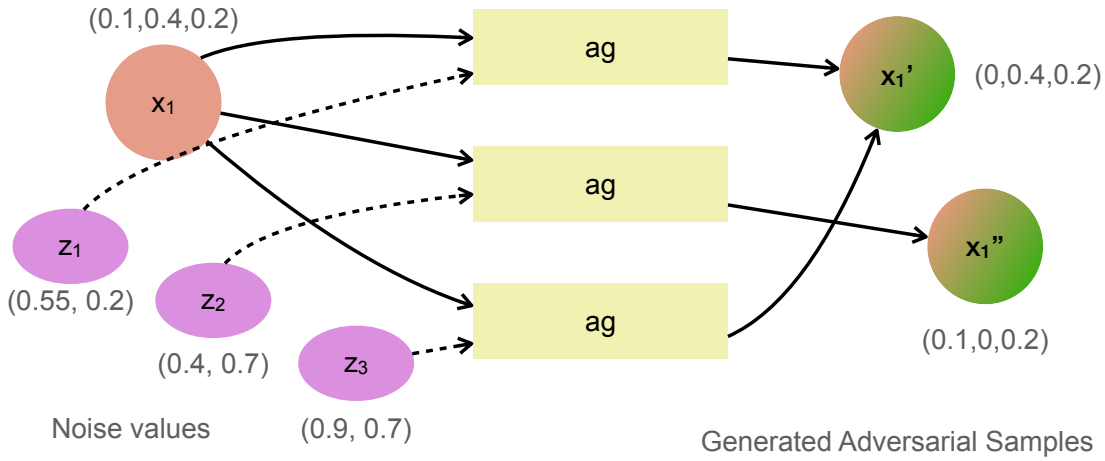

Fig. 3. Malicious adversarial samples generated for one attack and multiple noise values

This function applied on the same sample but for different noise values can produce exactly the same adversarial example. The values $(0.55, 0.2)$ and $(0.9, 0.7)$ illustrate this claim: they lead to the same result on x as shown in the figure 3. In such a situation, if we compute $Uniform(100, ag_s, (0.1, 0.4, 0.2))$ it is very likely that we obtain more than 40 times $(0.1, 0, 0.2)$ because the probability for a noise value uniformly distributed to have $z_2 < 0.5$ is 0.5.

Deep neural network models are known to be vulnerable to *MAdv*s [10]. Hopefully, efficient adversarial defense approaches have been proposed to prevent these attacks, and are surveyed in [24]. Adversarial training consist in adding sets of adversarial samples to the training dataset of an IDS.

Definition 9 (Adversarial Training pipeline for IDSs). *The adversarial training of an IDS is a training pipeline that uses a training dataset containing malicious adversarial samples generated from an attack generator ag.*

Basically, if the attack generator is not available, one has first to train an IDS as defined previously and then propose a relevant attack generator for this IDS. Finally, a new IDS is trained using the first training dataset to which adversarial samples generated with *ag* are added (for instance using the $Uniform$ procedure). Adversarial training or its extensions remain the most effective approaches to improve the robustness of classifiers against *MAdv*s [25]. The purpose of using *ag* is to inject samples into the training dataset to make the IDS robust compared to *ag* without having to execute anything on the real system. Otherwise, extending the dataset would be too expensive.

As said in section 3.1, the adversarial attack samples need to remain "equivalent" as much as possible i.e. still entail a malicious impact on the system. Finding a good attack generator remains challenging mainly because the extent to which a sample can be modified remains hard to evaluate. Next section discusses this situation and highlights the role of the set of allowed sample modifications in the performances of attack generator.

3.3 Adversarial neighborhood and a catalog of its various types

Adversarial sample generation is based on the assumption that an attack generator (AG) can only generate relevant adversarial attack samples. Section 7 highlights that it is possible to generate a large number of adversarial attack samples from a single attack sample. Hence, the question become what is the complete set of samples that can be obtained from an attack sample x through an attack generator. We introduce a notation to describe more easily this set. As this set can change for each attack, we need a function to designate for each attack this set.

Definition 10 (Adversarial neighborhood of attacks (ANA)). *An adversarial neighborhood of attacks ana is a function that returns for each attack sample x a set of sample denoted ana(x) such that each element of ana(x) is declared equivalent to x according to ana*

This notation facilitates the describe the possible outputs of an attack generator. Defining the adversarial neighborhood is the starting point of any attack generator design. However, choosing the neighborhood is always a very complex and challenging task in any constrained domain, specifically in the network security domain. As the ANA identifies to what extent attack sample can change and yet have a similar malicious impact on the system. It clearly means that the ANA depends on the attacks considered, in the kind of data that are collected on network activities.

This notion of neighborhood seems mostly related to cases where all the dimension of some subspace of the sample space $T_p = [0,1]^q$ can be freely modified.

Let review the different approaches

- The first approach allow to change everything in sample as in [3] and [29]. Such a neighborhood is far from realistic as it can hardly guarantee the malicious impact of these samples. On the other hand, it requires no particular knowledge on attacks at first sight. Yet, it might generate sample that are no longer malicious.
- The second group of approaches rely on their knowledge of the practical backgrounds that accompanied the construction of the dataset (e.g., tools used to generate the attacks, the network heterogeneity in terms of devices and systems). The dataset designers define the neighborhood the subset of sample features (e.g. dimension) that can be modified as in [16].
- The third group of approach still aim to determine which dimension in the same can be modified and how. Statistical criteria such as SHAP [18], LIME [26] on sample features are used to define the sample feature that can be modified freely. Other approaches have been applied such as singular value decomposition (SVD) [14] in order to extracting the important features that can preserve the impact of an attack class during sampling generation as done also in in [28], [27], [6], and [36].
- The last group of approaches relies on the domain experts. As cited in [11], and [31]. An expert analyzes the dataset to specify the features that can be modified for adversarial sample generation.

Many works focus on better exploring these ANAs as it can help to find successful malicious adversarial samples. Other studied how to improve the way noise is picked, or the attack generator built so that with less sample we can sufficiently improve the robustness of the defense mechanism [22]. The next section details the role of the ANA in creating contradictory samples with respect to a dataset containing both normal and attack samples (e.g. the training dataset of an IDS). Then, it explains how such samples can threaten the quality of an IDS trained with such a dataset, and how to mitigate those threats.

4 Contradictory adversarial sample, threat and mitigation

Attack generator are used on both sides, attack and defense. Thus, one has to be careful that their definition might be an issue for the defender either because it facilitates too much the task of the attacker to evade

detection, or because it might impair the quality of the trained IDS on non malicious adversarial samples. This section presents the contribution that motivated this paper, it explains why the notion of contradictory sample in section 2.2 needs first to be extended to capture the full consequences of choosing the ANA of an attack generator.

4.1 Revisiting the labelling issues for adversarial training datasets

In previous sections we highlighted the importance for the defender to ensure a training pipeline of good quality and to have samples that are label error free, as much as possible. In particular, we highlighted the case of contradictory samples. Recall that the adversarial training process basically consists in adding to a dataset called original dataset *OD* a set of adversarial samples called *AD* (adversarial dataset), *OD* is assumed to contain both normal and attack samples. Conversely *AD* contains only malicious adversarial attack samples.

We identified the following issue: even if two labelled dataset *OD* and *AD* have separately no contradictory samples, their union $OD \cup AD$ can have some contradictory samples. Actually, the pairs of contradictory samples would involve a normal sample of *OD* and some attack sample of *AD*. From the defender point of view, this situation would be the worst as an attack adversarial sample could not be distinguished from normal samples present in the original training dataset. The question become, is it sufficient not to use these samples in adversarial training dataset. Our claim is that it is not sufficient. In order to explain why it is not sufficient, we extends the notion of contradictory set to capture potential contradictions: contradictions between normal samples, and possible adversarial samples.

Definition 11 (Extended Contradictory set of *OD*). *Given an adversarial neighborhood ana, and a dataset OD, the Extended Contradictory set of OD with respect to the adversarial neighborhood ana is the set of all contradictory samples contained in* $OD \cup ana(A(OD))$ *and is noted* $EC(ana, OD)$*, more formally:*

$$EC(ana, OD) = \bigcup_{(x,normal) \in N(OD)} \{(x, normal), (x, attack) | (x, attack) \in ana(A(OD))\}$$

If this set is not empty it means that given the adversarial neighborhood *ana*, the content of *OD* make it possible to define an attack generator *ag* from *ana*, and an attack x such that $ag(x)$ is a sample that matches perfectly a normal sample from the training set *OD*. For Attacks samples from *OD* for which this is possible, it is thus very likely to be able to perform a successful evasion attack. Indeed, an IDS trained on *OD* is trained with the normal sample from the extended contradictory set and would most likely not contradict the class of the training sample, e.g. normal.

Definition 12 (Best Evasion Attack Candidates (BEAC)). *Best evasion attack candidate set is the set of attack samples in OD whom adversarial neighborhood do include a normal sample from OD.*

$$BEAC(ana, OD) = ana^{-1}(N(OD)) \cap A(OD)\}$$

This notion could be seen as a clue to the attacker whose sample could best evade the IDS after applying the attack generator. It should be noted that to have a non empty BEAC set, the attacker must have a good knowledge of normal samples. Let now illustrate all these concepts on a simplified dataset.

Figure 4 depicts a simplified dataset D_{simple} with four elements including two labelled attack samples, $(x_1, attack)$ and $(x_2, attack)$, and two normal ones $(n_1, attack)$ and $(n_2, attack)$(the detailed values are provided later). The sample space is $[0,1]^2$ and the noise space is $[0,1]$. We consider *ag* defined as follow:

$$ag((a_1, a_2), z) = \begin{cases} (0.1, a_2) \text{ if } z \leq 0,5 \\ (a_1, 0.8) \text{ Otherwise} \end{cases}$$

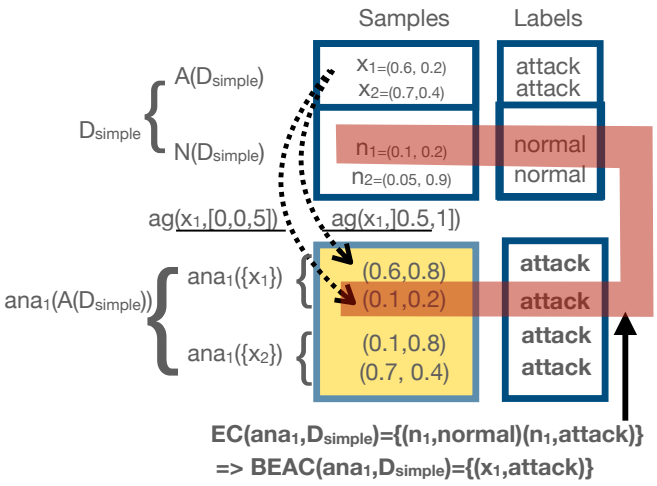

Fig. 4. Relation between *ANA*, *BEAC* and *EC*

We consider the following values for the samples: $x_1 = (0.6, 0.2)$, $x_2 = (0.7, 0.4)$, $n_1 = (0.1, 0.2)$, $n_2 = (0.05, 0.9)$. Let ana_1 be the neighborhood used for ag, then $ana_1(\{x_1\}) = \{(0.1, 0.2), (0, 6, 0.8)\}$, $ana_1(\{x_2\} = \{(0.1, 0.4), (0.7, 0.8)\}$. In this example, please notice that in fact n_1 belongs to $ana_1(\{x_1\})$. In this context, $EC(ana_1, D_{simple}) = \{(n_1, normal), (n_1, attack)\}$ an that the attack from which the adversarial sample n_1 can be obtained is x_1. Therefore, an IDS trained on D_{simple} has a high likelihood to predict the adversarial sample $ag(x_1, 0)$ as a normal sample (as it equals to n_1).

From the attacker's point of view, applying the attack generator on samples that are transformed into elements of $EC(ana1, D_{simple})$ could represent its best chance of success. In this case, the BEAC set is the singleton: $\{(x_1, attack)\}$. In this section, we have shown why the adversarial neighborhood of attack need to be known in order to anticipate for an IDS trained on a dataset the adversarial sample that if labelled attacks would contradicts normal sample from it training set. Ignoring such a situation is shown in the next section to be the source of several weaknesses for IDSs with or without adversarial training applied.

4.2 Threat to IDS robustness due to misuse of Attack Generators

In this subsection, we investigate how the knowledge of the *ANA* of an Attack Generator should be considered with care. Otherwise, retraining an IDS would either not provide the expected robustness to evasion attacks, or impair the system availability due to higher false positive detection rates. Those two events are the major issues that need to be taken care of when refining an IDS.

In section 3.3, we pointed out that several *ANA* have been considered for adversarial training. Moreover, in [23] authors insist on the fact that *ANA* definition could be difficult to capture in so call feature space (i.e. after preprocessing the raw observation of the system activity). In particular, they emphasize the difficulty of guaranteeing for *ANA* that the neighborhood contains only actual attacks, i.e. activities that produce a malicious impact.

Definition 13 (Impactful Neighborhood of Attacks). *An adversarial Neighborhood of attacks is said to be an Impactful Neighborhood of Attacks if it contains only samples for which a network activity could be observed in a case it yields a malicious impact on the system.*

A first issue is related to the difficulty to ensure that all the samples generated for adversarial training belong to some impactful neighborhoods of attacks. The first trivial case of such an issue can be found in the literature in early years of adversarial attack sample generation. This situation lead to a risk of generating adversarial samples labelled as attacks in the training process while they became simply normal sample after the attack sample is altered.

Definition 14 (Poisoning threat (Thr1)). *If adversarial training is performed for an IDS using an AG for which it cannot be proved that $AG(w,z) \in INA(w)$, then this training procedure is said to poison the IDS training dataset.*

Intuitively this situation would raise the rate of false positive on the IDS side without any action of the attacker. The use of Attack Generator for adversarial training aims at not paying the cost of carrying out the network activity on an actual system to capture observations of this activity. Yet, a workaround of the first threat could be to "execute" the adversarial sample against an actual network. Yet, recall that the number of needed samples remain unclear. Hence, it can lead to unbearable overhead to perform this dataset extension.

Definition 15 (Testing cost threat (Thr2)). *An adversarial training process is said to be subject to the testing cost threat if the adversarial attack samples are tested against an actual network to determine whether they remain attacks or not.*

Both threats could be simply disabled using only impactful neighborhood when designing an attack generator for generating an adversarial training dataset.

Let now assume the defender only considers an impactful neighborhood. Hence, the neighborhood of each attack only contains attacks. Now assume that *OD*, the original dataset available to train the IDS, has a non empty Extended Contradictory set, $EC_{ANA}(OD) \neq \emptyset$. This means that in this dataset, the defender knows a normal sample n and an attack a so that the observation of normal behavior, n, cannot be distinguished from at least a contradictory sample of a. Samples such as the normal sample n that belong to $EC_{ANA}(OD)$ are a weakness in the dataset used to form the IDS.

Definition 16 (Confusing normal sample threat (Thr3)). *An adversarial training dataset AD built from OD and neighborhood ana exhibits the confusing normal sample threat iff $N(AD) \cap EC(ana, OD) \neq \varnothing$*

Even if AT does not contain contradictory sample, it does offer the opportunity for the attacker to exploit the presence of "weak" normal samples that allow more efficient evasion attacks.

Finally the last threat is much more classic, attackers have a better evasion rate on underrepresented classes. Still, one would expect *BEAC* to be poorly represented (otherwise the detector's performance would be poor and the IDS not deployed). Yet this particular class is even worse because it is a class for which escape is a priori easier.

Definition 17 (Best evasion attack threat). *The* best evasion attack threat *corresponds to the situation where an attacker focuses on applying evasion attacks only for elements of $BEAC_{ANA}(OD)$ for an IDS trained on OD extended through adversarial training relying on neighborhood ANA.*

We now propose different mitigation strategies for these threats either based on fixing the *ANA* or the adversarial training pipeline.

4.3 Mitigation strategy

We identified four threats to an efficient usage of adversarial training related to the adversarial neighborhood. Note that the first two threats cannot be mitigated; they represent situations in which adversarial training is either too costly or risky as it may significantly reduce the system's availability. Here are mitigation strategies to deal with the two last threats.

Definition 18 (Sample removal). *This mitigation strategy consists in removing normal samples from D that belong to* $EC(ana, D)$.

This method is aimed at removing from training set the normal sample that are part of the extended contradictory set. Yet, the IDS might not have the forcefully be trained to output the attack class for such samples. In order to reduce the likelihood of evasion, we can change their label in D instead of removing them.

Definition 19 (Pessimistic relabelling). *Relabel any normal sample from* $D \cap EC(ana, D)$ *as attacks.*

We now introduce mitigation approaches for the last threat. The selection of attack samples on which the attack generator is applied is called the *attack sampling*. The attack sampling is almost never discussed on the attack side and is assumed to be uniform on defense one. Yet, attackers that restrict themselves to $BEAC(ana, D)$ elements would be less likely to be detected as too few training samples are available to the IDS for these elements or because it use an attack generator that creates an element of $EC(ana, D)$. Our mitigation strategy would be to change the proportion of $BEAC(ana, D)$ elements when sampling $A(D)$ during adversarial training.

Definition 20 (Oversampling of *BEAC*). *This strategy consists in increasing the likelihood of generating MAdvs from elements of BEAC.*

As the attacker might only apply evasion attack on BEAC set elements, the IDS need to be train in priority on this set. It thus need far more examples in this set because normal and malicious adversarial attack samples generated from this set can be very similar (and even equal if the first mitigation strategies are not applied).

First, the impact of confusing normal samples and best evasion attack candidates related threats need to be assessed on IDSs without mitigation. Secondly, it is also necessary to check to what extent the proposed mitigation approaches actually limit these threats. Next section details the experimentation campaign conducted to assess all these aspects on a dataset for which the ANA concept is defined.

5 Assessment method and architecture

This section presents the metrics and experiments used to asses both the threats identified previously and the effects of the proposed mitigation. It recall how GAN can be used as attack generator for a given neighborhood against IDSs. Finally, it details datasets targeted during our experiments.

5.1 Assessment objectives and selected metrics

As said above, we are interested in assessing the impact of non empty extended contradictory set. A first step is to find metrics to capture the performance of evasion attacks and adversarial training. Two basics expected properties of the IDS need to be measured: to what extent does the IDS detect actual attacks, and to what extent does it designate normal activities as attacks. The "abc" of classifier assessments relies on sets

of labelled test samples called test datasets to compute the so called confusion matrix [35]. The confusion matrix indicates for each kind of labels (normal or attack) the output of the IDS on the samples of this kind. In our case, we have a binary classification with "positive" and "negative" classes (positive stands for detected attack, negative stands for detection normal activity). Hence, true positives (TP) are actual attacks that are detected, false negatives (FN) are undetected attacks, true negatives (TN) are normal activities considered as legit activities by the IDS and false positives (FP) are normal activities identified as attacks. Different metrics can be derived from this confusion matrix [35]. The table recalls the main metrics used here based on counts taken from a confusion matrix (TP,TN,FP,FN) and explain how it can use to assess the IDS.

Table 1. Basic performance metrics and their description

Metric Name	Formula	Intuition from risk management point of view
Recall, R	$R = \frac{TP}{TP+FN}$	Ratio of detected attacks, meaningful if all attacks are equivalent
Precision, P	$P = \frac{TP}{TP+FP}$	Ratio of samples signaled as attacks that are actual attacks, useful if detection handling is costly
Accuracy, A	$A = \frac{TP+TN}{TP+TN+FP+FN}$	Ratio of correctly classified samples by the IDS, can be misleading if normal samples are far more present in the test set than attack samples.
F1-score, $F1$	$F1 = \frac{R*P}{R+P}$	Combined assessment of precision and recall. A high value means almost no attack evaded the detection and also almost no false positives among samples classified as attacks.

In the reminder, the notation for each metric is extended with the name of the dataset used as a test set, and the IDS it is submitted to when it is not obvious. Assume we compute the confusion matrix for the IDS $ids1$ using the test dataset TD then the recall would be denoted as $R(ids1, TD)$.

Additionally, metrics to assess how the evasion capability of an attack changed after applying an attack generator or IDS are required. Thus, we can compute the expectation ($\mathbb{E}$) of the recall, precision and other metrics on bags of adversarial samples generated through the $Uniform(i, ag, D)$ function.

We define Evasion Increase Rate for an IDS ids_1 and an attack generator ag and a set of attacks D as follows (for non null $R(ids_1, D)$):

$$EIR(ids_1, ag, D) = \mathbb{E}\left(\frac{R(ids_1, D) - R(ids_1, Uniform(1, ag, D))}{R(ids_1, D)}\right)$$

Hence, an EIR very close to 1 means the attack generator ag almost manages to make all attacks undetected by the IDS among those that were originally detected. A negative value means the attack generator makes it worse in terms of evasion, and a null value means it does not change anything in the average case.

The last metric helps to compare two IDSs against attacks generators. We introduce the evasion reduction rate that compares the EIR of two IDSs against two attack generators taking the same initial test dataset. Basically, it is used to see how two IDS work against the same type of evasion attacks.
The *evasion reduction rate (ERR)* compares EIR for the same attack generator but against different IDSs.

$$ERR(ids_2, ids_1, ag_2, ag_1, D) = 1 - \frac{EIR(ids_2, ag_2, D)}{EIR(ids_1, ag_1, D)}$$

Hence, assume ids_2 is a more robust than ids_1 against ag_1, $ERR(ids_2, ids_1, ag_1, ag_1, D)$ is expected to be strictly positive and as close as possible to 1. It is still important to check that the recall of both IDS is

similar to avoid false conclusion : a more robust IDS with a significantly lower recall on D is not necessarily better. If the value is negative, it means the second IDS is strictly less robust. Now we need to explain how attack generators are obtained in our experiments and how we do use them to generate test sets and adversarial training datasets.

5.2 Adversarial dataset design using Generative Adversarial Network (GAN)

In this section, we recall how one can obtain an attack generator using Generative Adversarial Networks (GAN). Then, we explain how we use it to generate the collection of adversarial samples used either for testing purpose of an IDS (mimicking an attack), or for adversarial training.

We assume a black-box training strategy for the GAN as it allows us to exploit it on both attacker and defender sides. Such generators are built to specifically target one IDS. The GAN is made of two components: the Generator and the Discriminator. The GAN-based attack generator targets one pre-trained IDS model and repeatedly updates the parameters of its components: the Generator and the Discriminator. We only consider neighborhoods that are subspaces (i.e. identified by a set of features that can be changed). The Generator only modifies the features of the attacks that are part of the selected ANA for the attack generator and generate *MAdv*s. The generator input is made of an attack sample concatenated with a random vector The Generator is trained to evade the detection from the Discriminator component. The Discriminator component concurrently is trained to match as much as possible the behavior of the IDS. This architecture aims at avoiding to have a Generator that cannot be progressively be optimized due to an IDS that is too efficient in terms of detection in the first step of the training.

Each component has its own loss function: Discriminator is penalized when it outputs a different class for a sample than the targeted IDS, and the generator is penalized when it generates adversarial samples that do not evade the Discriminator detection (ie. they are classified as attacks by the Discriminator). Thus, among the key parameters of the Generator and Discriminator, there is the number of update iterations called epochs. We consider GAN trained here either on 20, 100 or 1000 epochs which are typical values when training GAN to attack IDSs. A generator of type $GAN{-}N$ would denote a generator trained in N epochs. As the Generator keeps changing during the training, we only retain the best Generator when tested against the target IDS (and not the Discriminator this time). This point is important to avoid oddities as the optimization process is not guaranteed to be monotonous in terms of performances of the Generator against the IDS.

Let us now detail how the datasets are named and built in our experiments. First, the amount of BEAC samples in a dataset of adversarial samples for a given attack generator depend only on the dataset and the neighborhood considered in the generator definition. As we want to see how this ratio affects training and testing, we will generate datasets of adversarial samples in two steps (assuming the generator is trained). Let q in general denote the desired ratio of desired adversarial samples directly derived from BEAC samples for a dataset D and attack generator ag. For simplicity, we assume q is a fraction (i/p) with i, p two integers.

Then, we define $AdvDataset(N,q,D,ag)$ a set of N samples drawn uniformly from $Uniform((p-i)*\lceil N/p\rceil, ag, A(D)) \cup Uniform(i*\lceil N/p\rceil, ag, A(D))$. This union of bags contains at least N elements, the proportions of adversarial samples derived from BEAC samples is in the average q, and all those samples are adversarial attack samples derived from D.

The adversarial training datsets consist of the union of the dataset OD, that needs to be extended, with a set of adversarial samples. This set is obtained applying $AdvDataset$ on OD with a selected generator, a size parameter, and a ratio of BEAC elements (if omitted the ratio is assumed unchanged compared to the proportion of such element in OD). We focus in our experimentation only on adversarial training dataset sizes that are multiples of the size of the dataset they extend. Let $AdvTrainDateset$ be the procedure applied to generate these datasets. Its parameters are OD, s, ag, and q such that:

- *OD* is the dataset to be extended.
- *s* defines the size of the set of adversarial sample to be added as a multiple of the size of *OD* (ie. s times the cardinal of the original dataset)
- *ag* is the attack generator used to produce the adversarial samples
- *q* is the rate of BEAC elements of *OD* used as seed to generate the adversarial samples (considering the ANA of *ag*).

The adversarial training datasets obtained by $AdvTrainDateset$ are built using the generated sample from $AdvDataset$.

We rely for dataset extension of *OD* using GANs specifically trained to attack IDSs trained on *OD* as it is. In addition to the above mentioned parameters, we need a parameter to tell the number of epochs during which the generator is trained. Moreover, there could be variability in GAN training performances because the training process rely on sampling uniformly the noise domain to generate adversarial samples. Hence, training the GAN could yield very distinct results. For each vector of parameters, we always train 50 attack generators of the same kind and use each of them to generate the collections of adversarial samples used as test sets. It provides us a mean to capture the average behavior the identified threats and mitigations. It gives us also a mean to determine how stable are our conclusions. Hence, when it is not specifically said the value depicted are the average value over experiments carried out with each of these 50 GANs. For adversarial datasets, we train attack generators as long as needed against an IDS trained on the original dataset so that they achieve an *EIR* higher than a threshold. If this threshold is not met, the process is fully restarted as it might be the hint that the Discriminator became too good too early. Since IDS training datasets are generally highly vulnerable to adversarial attack samples, we chose a very high threshold (0.99).

In an integrated process, we generate many artifacts given an IDS trained only on non adversarial samples and its training set.

Algorithm 1 *IDS* Adversarial training

Input: OD (an original dataset), ids (an IDS trained on OD), *s* (extension size factor), q (rate of added adversarial sample obtained from *BEAC* samples, ep the number of epochs to be considered (if 0, the threshold is applied);
Output: *ag* the attack generator targetting *ids*, *at* corresponding to $AdvTrainSet(s,ag,q)$, *idsat* the IDS trained on *ad*;

1: **procedure** ADV-TRAIN(OD, ids, s, q)
2: Train a GAN over ep epochs or as long as the threshold is not met if $ep = 0$ against *ids*. Identify the Generator with the best performances against *ids* and store it into *ag*.
3: Apply $AdvTrainData(OD,s,ag,q)$ to obtain one adversarial training dataset, let *ad* be this dataset.
4: train an IDS on *ad* and store it into *idsat* ; {A}t the end of this process *idst* contains the new ids, *ag* the attack generator use for this, and *ad* contains the adversarial training dataset.
5: **end procedure**

We explained the kind of dataset we need to generate and how we do obtain the related attack generator.

Let us now discuss the "original datasets" on which we focused, and what are theirs possible ANAs, and related BEAC rates. This will give us basic clues about the reality of the situation of a non-empty extended contradictory set.

5.3 Datasets in use

This section describes the datasets used to evaluate the impact of adversarial neighborhoods on IDSs and attacks and discuss the neighborhood that can be found in the literature for these datasets.

NSL-KDD dataset To our knowledge, NSL-KDD is the only dataset that provides a clear definition of the ANA based on the dataset creator criteria (section 3.3). Since it does not cover recent attacks, NSL-KDD is considered an out-of-date dataset. Many papers, however, used it and overlooked the threat identified in section 4.2. Therefore, we assessed NSL-KDD because it provides a clear definition of *ana*1. The NSL-KDD attacks are classified as Denial of Service (DoS), User to Root (U2R), Root to Local (R2L), and Probe. Each record in the NSL-KDD training dataset has 41 features and class identifiers. For NSL-KDD, the concept of *ana*1 is defined by the concept of functional features: the dimension that maintains the attack malicious impact [16]. Hence, the ANA is obtained by changing the non-functional features of an attack class. NSL-KDD features are split into four groups: Intrinsic, Time-based, Content, and Host-based. Not surprisingly, the functional features are different for each attack category:

- DoS attacks: Intrinsic and Time-based
- U2R attacks: Intrinsic and Content
- R2L attacks: Intrinsic and Content
- Probe attacks: Intrinsic, Time-based and Host-based

We concentrate in this work on the normal, DoS, and probe classes because R2L and U2R have few samples. It should be noted that after searching for the BEAC set on the training dataset of NSL-KDD restricted to DoS and Probe attacks. We found that $BEAC(ana1, D)$ contains only DoS samples that represent 3.74% of DoS samples. In this dataset, the samples contain categorical attributes, such as protocol type or flags. As a result, we used the standard pre-processing approach on the dataset samples to obtain fully numerical attributes [17].

CIC-IDS2017 dataset As previously indicated, identifying the perfect neighborhood is very challenging. It might not even be unique conversely to NSL-KDD. In order to study this aspect, we selected CIC-IDS2017 [28], a dataset that is a good trade-off between a recent dataset and a mature one (one that is well understood).
CIC-IDS2017 contains fourteen types of attack traffic, including DoS and infiltration, along with normal traffic. Each sample of the dataset consists of more than 80 features. We found different definitions of neighborhood that can be considered as "relevant neighborhoods" for this dataset. It should be noted, however, that all of the neighborhoods defined in this dataset correspond to selecting a subspace of the feature space that can be freely modified. It means that to define these ANAs, we simply need to idetify which features can be freely modified. We identified two different ANAs. The first one is based on statistical criteria [27], while the second is based on expert knowledge [11].

- In [27], the authors of CIC-IDS2017 define *ana*2 by recognizing the most 28 relevant features associated with DoS attacks using the SVD approach. Their approach assume these features should remain as is. Hence, the adversarial neighborhood is provided by modifying all features except the 28 relevant ones.
- [11] follows security expert criteria and define *ana*3. They identified the relevant attack features as the one that cannot be modified by an attacker without losing the malicious impact of these network activities. This analysis results in the identification of 42 relevant features for DoS attacks. *ana*3 is obtained freely modifying the non relevant features again.

As done for the NSL-KDD dataset, we compute the proportion of *BEAC* elements for each neighborhood. For the first definition of *ANA* it reachs 9.3% of DoS samples, while the *BEAC* set in the second definition represents 5.2% of the DoS samples. The pre-processing approach to convert the value of the attributes in this dataset into fully numerical attributes is based on [7].

6 Assessing the threats on IDS training pipeline

This section measures the threat level of identified issues in IDS training pipeline before and after adversarial training. We are especially interested in examining how well IDS performs on the non-empty BEAC set and whether or not this set is problematic for IDS detection performances before even considering adversarial training.

In our experiments, we train IDS on multiple datasets to test adversarial training with and without mitigations. We use the following notation to identify those IDSs: $ids-N-X-an-sr-opt$ where an, sr and opt are optional variables.

- N denotes the name of the dataset. Mainly, *nsl* for NSL-KDD and *cic* for CIC-IDS2017.
- X denotes the size of dataset considered for training. (detailed later).

The parameter an specifies the type of adversarial neighborhood in use for a dataset. Mainly, $ana1$ for the NSL-KDD neighborhood, while $ana2$ and $ana3$ represent the statistical and expert neighborhoods for CIC-IDS2017, as described in section 5.3. The parameter sr specifies the sampling rate used by $BEAC(an,X)$ if applied (not provided if unchanged). The opt parameter indicates whether the confusing normal samples are removed (noted as *wn* for without normal) or relabeled as attacks (noted as *na* for normal as an attack).

6.1 Assessment of detection performance of IDSs with adversarial training

This subsection examines how well regular adversarial training with different adversarial neighborhoods performs without using the proposed mitigation strategies.

NSL-KDD As previously stated, probe and DoS attacks illustrate two distinct situations. When we compute the NSL-KDD BEAC set, we find that it contains no probe attack samples but only DoS samples. Thus, we train two IDSs, one for Probe and one for DoS attacks, and observe how they behave. The dataset to train the IDS to detect DoS attacks contains normal and DoS samples from the NSL-KDD training dataset noted as od_{DoS}. The second data set, denoted as od_{Pr}, contains normal and Probe samples only from the NSL-KDD training dataset.
We train two batches of 50 GANs, following the architecture of $GAN-100$ against $ids-nsl-od_{Pr}$ and $ids-nsl-od_{DoS}$. We observe that evasion attacks reach an average EIR of above 0.99 for $ids-nsl-od_{DoS}$ and is 0 for $ids-nsl-od_{Pr}$. These results suggest that our claim that the content of the BEAC set is correlated to the success of the evasion based on malicious adversarial samples is likely. Note that the trained IDSs have shown usual performances on non adversarial samples, e.g. as in [34], of ML-based IDSs. Hence, we have a 0.85 recall (ability to detect attacks) for $ids-od_{dos}$ on non adversarial samples. The next step is to understand how efficient is the adversarial training for different size factor parameters. Here, we test an adversarial training for size factors of 2, 5, and 10, leading to extended datasets for $at2$, $at3$, and $at4$. Each dataset is used to train an IDS without certain mitigations of the identified threats.
These IDSs have been tested against strong attack generators of type $GAN-1000$ to better understand their performances, and results are presented in Figure 5. We notice that the risk of the adversarial evasion attacks drops significantly from 100% EIR on $ids-nsl-od$ to reach 9% EIR on a very costly IDS in terms of training, which is $ids-nsl-at_4-ana1$. Adversarial training performs well but at the expense of scaling factors that are 5 or 10 times the size of the original dataset. Training an IDS on 10 times larger datasets, despite a linear effect on the training cost, might not be accepted as the training time also depends on the number of epochs(i.e for large epochs parameter 10 times larger is perhaps too large). Note that applying adversarial training has not impaired the performance of *IDSs* when no evasion attack is applied, as shown in the first three lines of Table 5.

CIC-IDS2017 The $ids-cic-od$ is trained to detect CIC-IDS2017 DoS samples and has a recall of 0.94 on non-adversarial samples, consistent with the literature [7]. However, a $GAN-20$ attack completely evaded IDS detection. We investigated the efficiency of enhanced CIC-IDS2017-based IDSs with adversarial training for a different size factor, namely on $at2$, $at3$, and $at4$, as we did for IDSs trained on NSL-KDD. We also trained two groups of IDSs on the extended datasets by using two different adversarial neighborhoods $ana2$ and $ana3$ as noted in section 5.3. After testing these two groups of IDSs against an attack generator of type $GAN-100$, the group of IDSs using $ana2$ systematically failed to detect the adversarial samples regardless the size of the adversarial training used. Note the IDSs in the $ana3$ group performed better against this type of attack, as depicted in figure 5. This result is consistent with our intuition that the larger is the ANA for adversarial sample generation the harder is the adversarial training and the larger is the BEAC set.
Even if the detection rate on non-adversarial samples is not degraded before and after adversarial training, as shown in the first three rows of table 2, it is clear that adversarial training is less efficient in CIC-IDS2017 than in NSL-KDD.

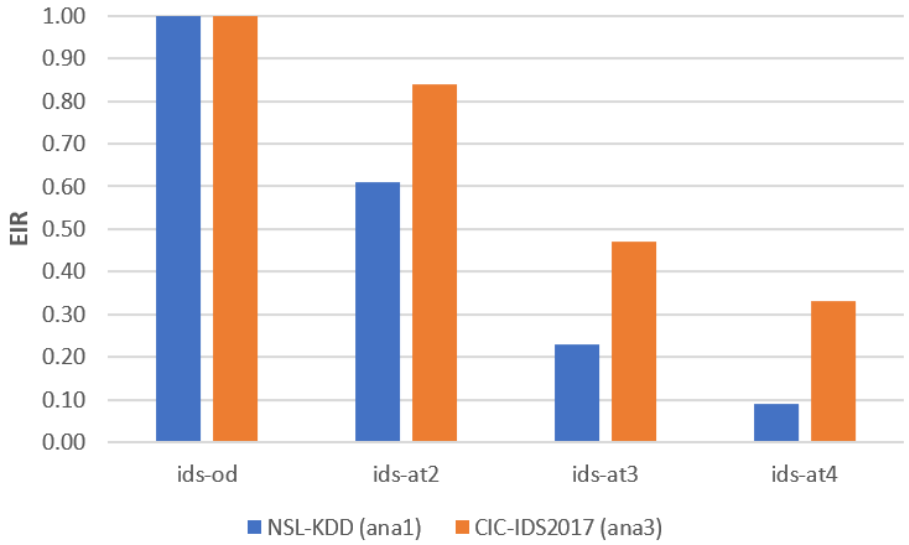

Fig. 5. The average EIR of $ids-at_2$, $ids-at_3$ and $ids-at_4$ for NSL-KDD and CIC-IDS2017

Table 2. Performance metrics of idss for different adversarial training dataset based on CIC-IDS2017

IDSs	Precision	Accuracy	F1 Score	Recall
$ids-cic-od$	95.3	94.7	94.3	94.0
$ids-cic-at2-ana3$	95.9	95.4	95.0	94.7
$ids-cic-at4-ana3$	96.7	96.2	96.0	95.8
$ids-cic-od-na$	96.9	94.9	96.3	97.7
$ids-cic-at2-ana3-25$	96.2	95.4	95.5	95.5

6.2 Assessing BEAC set threat on IDSs without mitigation

We need to determine whether the existence of a non-empty *BEAC* set is problematic for the IDS detection performance.

IDS reinforced by adversarial training requires at least twice the size of the original dataset to have good detection performance against evasion attacks. This improvement comes at a very high cost in terms of computational training time, as training on larger datasets has a linear effect on the training cost, as illustrated in figure 5. However, comparing the detection performance of different sample categories within a dataset, while focusing on the BEAC set adversarial samples, and other adversarial samples, reveals that even the most expensive and robust IDSs are highly vulnerable to the BEAC set samples. Figure 3 depicts this observation on the IDSs trained with the NSL-KDD dataset. We observed a significant difference in detection performance between adversarial samples inside and outside the BEAC set. For instance, $ids-nsl-at_4-ana1$ is trained on an adversarial dataset that is 10 times larger than the original dataset. This IDS has a very good detection performance on the non-BEAC adversarial samples, with a R of 82%. However, detection performance on the BEAC set adversarial samples dropped dramatically to 15%. Furthermore, the same observation is noticed on a different dataset, the CIC-IDS2017 which follows *ana*3 neighborhood as shown in figure 4. Although the detection capabilities in this situation are lower than in the NSL-KDD case. On the other hand, $ids-cic-at4-ana3$ still has good detection performance on adversarial samples that do not belong to the BEAC set, with a R of 52%. Yet, its detection performance on the BEAC set is too low, with a detection rate of around 8%.

These results demonstrate how risky adversarial samples generated from the BEAC set are for IDSs. It can be observed whatever systematically considering different neighborhood definitions in the adversarial sampling strategy. Therefore, in the next section, we perform experiments to evaluate the mitigation proposed in section 4.3 to improve the IDS detection against the adversarial sample generated from the BEAC set. The the detection performance on non-adversarial attacks or other evasion attacks that differ from the BEAC set are also assessed to check the overall performance are not impacted.

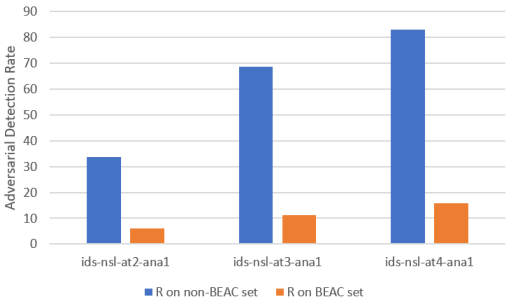

Table 3. Measurement of adversarial detection rate for BEAC and non-BEAC samples for various IDSs trained and tested on NSL-KDD.

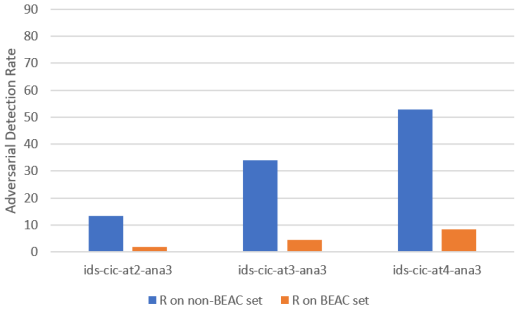

Table 4. Measurement of adversarial detection rate for BEAC and non-BEAC samples various idss trained and tested on CIC-IDS2017 with *ana*3.

7 Assessing the impact of the mitigation strategies

We want to evaluate the effectiveness of the proposed mitigation strategies (4.3). In particular, we have to assess the impact of various adversarial training sampling strategies on IDS detection capabilities on adversarial samples.

7.1 Confusing samples mitigation

This subsection evaluates the impact on performances when applying both regular and robust IDS of sample removal and sample pessimistic relabeling strategies defined in section 4.3.

Effect on regular IDS performances As pointed out, confusing samples can make it harder for the IDS to resist attack generators. We train two IDSs to see how the normal sample threat affects the resilience of regular IDSs. The first is called ($ids-nsl-od-wn$), and was trained on *od* without confusing normal samples using the sample removal mitigation. The second is $ids-nsl-od-na$, and was trained on od. We relabeled the confusing normal samples as attacks, as proposed in the sample pessimistic relabelling mitigation.

Table 5. Performance metrics of various ids models trained on NSL-KDD dataset

IDSs	Precision	Accuracy	F1 score	Recall
$ids-nsl-od$	85.8	86.4	84.4	85.1
$ids-nsl-at2-ana1$	82.4	86.2	84.6	86.9
$ids-nsl-at4-ana1$	88.4	87.6	85.2	82.2
$ids-nsl-od-na$	88.1	87.9	85.7	83.4
$ids-nsl-at2-ana1-25$	87.6	87.3	84.9	82.5

These IDSs are not designed to be resilient to evasion attacks. Thus, we use the weaker attack generator. In the case of NSL-KDD, we primarily use the generator of type $GAN-100$, and in the case of CIC-IDS2017, we mainly use the generator of type $GAN-20$. Each attack generator was specifically trained against $ids-od$ and tested on $ids-od-wn$ and $ids-od-na$.

NSL-KDD: We observe in the case of NSL-KDD that the attacks still manage to evade $ids-nsl-od-wn$ completely. However, the average *ERR* for $ids-nsl-od-na$ is 0.5 in this case.

CIC-IDS2017: This set of experiments shows that even after applying the first two mitigation techniques, the attack generator is still quite effective in the case of the second definition of neighborhood *ana*2, with an ERR of nearly 0.
In the case of the third definition of neighborhood *ana*3, we observed that the detection performance of the adversarial samples of the $ids-cic-od-wn$ and the $ids-cic-od-na$ have improved, with the ERR increasing to 0.15 and 0.12, respectively.
Managing the confusion normal samples can help mitigate threats even on weak IDSs. These findings are encouraging because relabeling is a low-cost adversarial training method. As a result, normal samples related to $EC(od)$ appear to be very useful for adversarial training.
Let now consider adversarial training with these mitigation strategies.

Effect on robust IDS performance We examine the mitigation approaches for confusing samples combined with IDS reinforced by regular adversarial training. We assess the performance using $ids-at_2$, $ids-at_2-wn$, with sample removal mitigation and $ids-at_2-na$ using the sample pessimistic relabeling mitigation. This time, we use 50 $GAN-1000$ for NSL-KDD and $GAN-100$ for CIC-IDS2017 attack generators against each of these idss.

NSL-KDD: The first row in table 6 summarizes the results of the case of IDSs trained with NSL-KDD. The *ERR* of the ids without mitigation but with adversarial training on at_2 remains low at 0.39. However, the first mitigation strategy, sample removal, performs significantly better this time. It increased the ERR by almost 50% to 0.57. Surprisingly, the relabelling strategy did not perform as expected, with a slight increase in *ERR*.

Table 6. ERR of *GAN* attacks on idss trained and tested on NSL-KDD and CIC-IDS2017 with *ana*3

	ids-at2	ids-at2-wn	ids-at2-na
NSL-KDD (ana1)	0.39	0.57	0.43
CIC-IDS2017 (ana3)	0.16	0.36	0.33

CIC-IDS2017: When the first two mitigations were applied to $ids-cic-at2-ana3$, the ERR increased from 0.16 to approximately 0.36 on both IDSs. However, $ids-cic-at2-ana2$ appears to be very vulnerable to evasion attacks, with *ERR* nearly equal to zero on all tested idss.
The first two mitigations manage to close the gap between IDSs trained using adversarial training of scaling factor 2, i.e., 0.39, and scaling factor 5, 0.83 in the case of NSL-KDD by at least 50%. They also strengthen the IDSs' resistance to evasion attacks. However, we found that the larger the *ANA* space, the less effective the adversarial training, as in the case of $ids-cic-at2-ana2$. This limitation is due to the complexity of the sampling in the high dimensionality of the neighborhood. This issue is beyond the scope of this paper. Moreover, while those two mitigations perform well against evasion attacks, removing a portion of normal samples from the training dataset can impair IDS detection capabilities against regular attacks, especially if this proportion of normal confusion samples is large.
In the next subsections, we examine the effect of the oversampling strategies on the *BEAC* set elements when generating the adversarial samples. We compare it to adversarial training approaches in which this rate is unchanged.

7.2 Adversarial training on *BEAC* set

In this section, we examine adversarial training datasets formed by controlling the sampling rate of samples from the *BEAC* set.
This set of experiments compares the effect of the last proposed mitigation to sample removal and sample relabeling. We consider *BEAC* sampling rates of $\{0.25, 0.50, 0.75\}$. The *sr* parameter value is used to represent percentages. Therefore, $ids-at2-25$ represents an adversarial training on a dataset with a size factor of two and a sampling rate of 0.25. We use $GAN-1000$ on the NSL-KDD neighborhood and $GAN-100$ on the CIC-IDS2017 that follows *ana*3 neighborhood against idss trained with an adversarial training set of size factor of 2, e.g., $ids-at_2-j$ family.

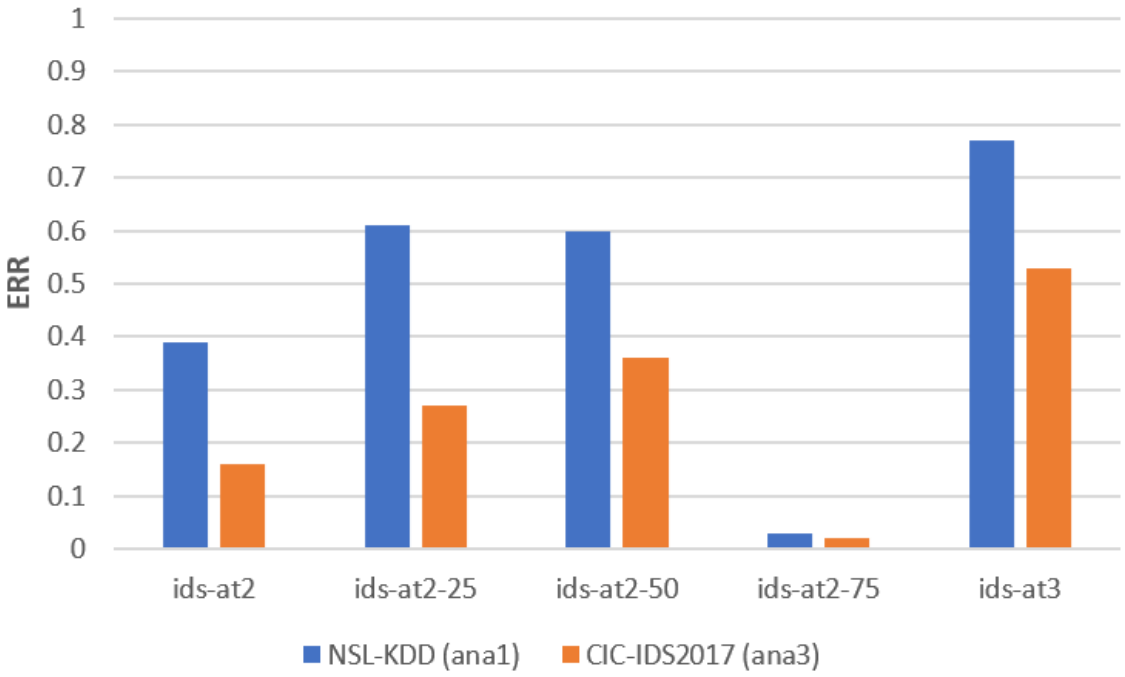

Fig. 6. Average ERR on 50 *GAN* against $ids-at_2-j$ for various j trained and tested with NSL-KDD and CIC-IDS2017 with $ana3$.

The results of the experiments on idss trained on NSL-KDD and CIC-IDS2017 with $ana3$ are shown in figure 6. The results reveal that adversarial evasion attacks are much less effective in most of these idss than $ids-at_2$ but significantly more effective in $ids-at_2-75$. However, this situation is expected because a high sampling rate prevents the IDS from correctly detecting evasion attacks applied to attacks in $A(D)-BEAC(D)$. A sample rate of 25% to 50% seems similar for the NSL-KDD neighborhood, while the latter seems slightly better for CIC-IDS2017.
Furthermore, using oversampling of *BEAC* mitigation narrows the margin with $ids-at_3$ in both neighborhoods by nearly dividing the difference in ERR by two.

7.3 Assessing the effect of the mitigations on BEAC set

Section 6.2 demonstrated that IDSs are sensitive to evasion attacks from the BEAC set, even after adversarial training has increased overall detection performance on the whole population within the tested dataset.
Table 7 shows that the detection performance of IDS is improved drastically on evasion attacks that specifically belong to BEAC set after applying the proposed mitigations. For instance, in the case of IDSs that follow the NSL-KDD neighborhood, the results show that using the first two mitigations increase detection on BEAC by approximately 3 times compared to regular adversarial training, as the adversarial detection rate increases from 6% on $ids-nsl-at2-ana1$ to nearly 20% on $ids-nsl-at2-ana1-wn$ and $ids-nsl-at2-ana1-na$. However, after applying the oversampling mitigation strategy, the adversarial detection rate rose by 5, with $ids-nsl-at2-ana1-25$ and $ids-nsl-at2-ana1-50$ reaching nearly 40%.

Nonetheless, implementing the mitigations increases detection on the BEAC set regardless of the chosen neighborhood or adversarial sampling technique. This finding is backed by our experimental results: the detection rate on BEAC samples is improved not only for IDSs defined for NSL-KDD neighborhood but also for IDSs trained with respect to the second neighborhood defined for CIC-2017 (*ana*3). However, the detection ratio differs from one ANA to the other. In addition, the oversampling mitigation outperforms the two other mitigations proposed as it ensures good average detection on the full set of attacks but also specifically on adversarial sample generated from BEAC set elements.

Table 7. IDSs detection performance on evasion attacks belonging to BEAC set

	ids-at2	ids-at2-wn	ids-at2-na	ids-at2-25	ids-at2-50
NSL-KDD BEAC set on *ana*1	6	20.2	20	39.2	39.5
CIC-IDS2017 BEAC set on *ana*3	1.9	14.1	16.9	26.3	32.7

8 Related Work

[8] proposes criteria for better dataset sampling strategies and the use of diversification approaches to improve the training of generic classifiers. However, they do not take into account the particularity of IDS as a security classifier or even the specificity of adversarial samples. With the recent advances in Machine Learning research, adversarial attacks have piqued the interest of researchers in a wide variety of domains. Several studies have focused on the sampling strategy of *MAdv*s in order to achieve a reasonable balance between IDS training performance and detection capabilities against adversarial evasion attacks.
In [3], [29] and [19] generated *MAdv*s by applying the mutation on the entire set of features on the CIC-IDS2017 attack samples. While in [12], they generated *MAdv*s from NSL-KDD attack classes using two methods, either by mutating all the dimensions of attack samples or by mutating the 16 principal components obtained with Principal Component Analysis (PCA) as a dimensionality reduction technique. Other works follow the same methodology in generating adversarial samples, such as in [37] and in [1]. In all these works, there is no limit to the *MAdv* sampling subspace because the adversarial generator is applied to the entire set of features in the feature space. However, not defining the adversarial neighborhood of the *MAdv*s can increase the likelihood of generating a *MAdv* that successfully avoids IDS detection. But, the impact of those *MAdv*s on the actual systems is not assured because there are no constraints on the adversarial generator; thus the preservation of attack behaviors is not assured after the mutation process.
To the best of our knowledge, all approaches for the NSL-KDD dataset that take *ANA* into account define attributes that divide attack samples as functional or non-functional features based on the categorization given by [16]. Hence, the subspace of producing *MAdv*s is bounded by adjusting only the non-functional features of any attack class while preserving attack behavior by leaving the functional features unchanged, as in [17], [38] and [32]. However, choosing the adversarial neighborhood for *MAdv*s generation is a very challenging task in any constrained domain, specifically in the network security domain. Furthermore, selecting *ANA* is critical for defining any adversarial sampling strategy because the subspace of sampling *MAdv*s depends on the selected adversarial neighborhood. This difficulty of defining *ANA* is reflected in the literature as many papers define different approximations of *ANA* even for the same attack class in the same dataset, as in CIC-IDS2017.

To our knowledge, the choice of *ANA* is typically performed through statistical criteria or throughout a domain expert analysis. According to [28], the authors use RandomForestRegressor to determine the optimal short feature set for each attack, which can then be used to detect those attacks. [20] and [36] use the [28] method to define *ANA* in order to craft *MAdv*s with GAN attacks generator. [27] defines ANA as a dimensional reduction technique based on singular value decomposition (SVD), which can provide insights into the relationship of the selected features after dimension reduction. The SVD can then be used to determine the most significant features for the adversarial neighborhood from the given feature set. [6] the authors rely on the Shapley Additive Explanations (SHAP) to identify *ANA* that preserves the attack's functionality during the adversarial sampling. Other works rely on other statistical criteria, such as in [2] and in [33]. Nevertheless, using statistical approaches to explore the adversarial neighborhood can suffer from problems, such as the disagreement problem [15], in which different statistical criteria end up selecting different functional features for the same type of attack inside the same dataset.

[11] offers systematic expert analysis to develop effective and efficient adversarial samples on the CIC-IDS2017. In their work, they proposed the idea of categorizing the features of this dataset into four groups, i.e., grouping flows based on whether they can be modified by an adversary and still yield correct flows. [31] relies on [11] work to present their own definition of *ANA*. While in [39], the authors introduce a MACGAN framework that is divided into two parts. In order to examine *ANA*, the initial part of MACGAN is used to analyze the attack based on the author's expertise. Then, in the second part, they use a GAN attack generator to bypass the IDS.

In this paper, we examine the quality of the extended dataset after *MAdv*s mutations. In addition, this paper aims to examine the threats linked to the generation process used in adversarial training. Furthermore, we propose a new strategy to improve the IDS performance of adversarial training by changing adversarial sampling strategies to account for confusing samples and *BEAC* samples in two datasets, the NSL-KDD, and the CIC-IDS2017.

9 Conclusions

This paper examines how different adversarial sample generation approaches affect the quality of adversarial evasion attack samples. In particular, this paper focuses on how the newly generated samples can affect IDS' robustness performance and the quality of IDS's training pipeline. We revisited previous work on well-understood models and datasets. This assessment aids in defining the notion of adversarial neighborhood of an evasion attack sample (*ANA*), which helps to identify and formalize threats to the robustness of IDSs against adversarial evasion attacks. These threats are enabled by flaws in the structure and content of the dataset rather than its representativeness. We have demonstrated that even the most robust intrusion detection systems are vulnerable and perform poorly against a specific set of evasion adversarial attacks, the best evasion attack candidates samples (BEAC). In addition, we have developed a method to improve adversarial training performance by making it to focus on the BEAC set in the dataset. We found that this technique increases the detection of BEAC samples and the detection of IDS on all adversarial evasion attacks, regardless of the *ANA* used to generate the adversarial samples.

However, while the proposed mitigations improve the efficacy of adversarial training, this strategy has a limit. This drawback comes from the complexity of covering the full dimensions in the feature space when producing efficient adversarial samples (e.g., the limitation is severe when the dimensions of *ANA* are very close to the full dimensions in the feature space).

In future work, we will investigate the adversarial neighborhood of attacks in the problem space domain to determine how to overcome this limitation on proposed mitigations in the feature space domain [23].

Acknowledgements This research is part of the chair CyberCNI.fr with support of the FEDER development fund of the Brittany region.

References

1. Alahmed, S., Alasad, Q., Hammood, M.M., Yuan, J.S., Alawad, M.: Mitigation of black-box attacks on intrusion detection systems-based ml. Computers **11**(7) (2022)
2. Alhajjar, E., Maxwell, P., Bastian, N.: Adversarial machine learning in network intrusion detection systems. Expert Syst. Appl (2021)
3. Ayub, M.A., Johnson, W.A., Talbert, D.A., Siraj, A.: Model evasion attack on intrusion detection systems using adversarial machine learning. In: 2020 54th Annual Conference on Information Sciences and Systems (CISS) (2020)
4. Backes, M., Manoharan, P., Grosse, K., Papernot, N.: Adversarial perturbations against deep neural networks for malware classification. CoRR (2016)
5. Chaitou., H., Robert., T., Leneutre., J., Pautet., L.: Threats to adversarial training for idss and mitigation. In: Proceedings of the 19th International Conference on Security and Cryptography - SECRYPT,. pp. 226–236. INSTICC, SciTePress (2022)
6. Chauhan, R., Shah Heydari, S.: Polymorphic adversarial ddos attack on ids using gan. In: 2020 International Symposium on Networks, Computers and Communications (ISNCC) (2020)
7. Faker, O., Dogdu, E.: Intrusion detection using big data and deep learning techniques. In: Proceedings of the 2019 ACM Southeast Conference. ACM SE '19, Association for Computing Machinery (2019)
8. Gong, Z., Zhong, P., Hu, W.: Diversity in machine learning. IEEE Access (2019)
9. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NIPS (2014)
10. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples (2015)
11. Hashemi, M.J., Cusack, G., Keller, E.: Towards evaluation of nidss in adversarial setting. In: Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks. Big-DAMA '19, Association for Computing Machinery (2019)
12. Khamis, R.A., Shafiq, M.O., Matrawy, A.: Investigating resistance of deep learning-based ids against adversaries using min-max optimization. In: ICC (2020)
13. Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J.: Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecur. (2019)
14. Klema, V., Laub, A.: The singular value decomposition: Its computation and some applications. IEEE Transactions on Automatic Control **25**(2), 164–176 (1980)
15. Krishna, S., Han, T., Gu, A., Pombra, J., Jabbari, S., Wu, S., Lakkaraju, H.: The disagreement problem in explainable machine learning: A practitioner's perspective. arXiv preprint arXiv:2202.01602 (2022)
16. Lee, W., Stolfo, S.J.: A framework for constructing features and models for intrusion detection systems. ACM TISSEC (2000)
17. Lin, Z., Shi, Y., Xue, Z.: IDSGAN: Generative Adversarial Networks for Attack Generation against Intrusion Detection. arXiv e-prints (2018)
18. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. Curran Associates Inc. (2017)
19. Martins, N., Cruz, J.M., Cruz, T., Abreu, P.H.: Analyzing the footprint of classifiers in adversarial denial of service contexts. In: Moura Oliveira, P., Novais, P., Reis, L.P. (eds.) Progress in Artificial Intelligence. pp. 256–267. Springer International Publishing (2019)
20. Msika, S., Quintero, A., Khomh, F.: Sigma : Strengthening ids with gan and metaheuristics attacks (2019)
21. Papernot, N., Mcdaniel, P., Goodfellow, I.J., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against machine learning. ACM ASIACCS (2017)
22. Picot, M., Messina, F., Boudiaf, M., Labeau, F., Ayed, I.B., Piantanida, P.: Adversarial robustness via fisher-rao regularization. ArXiv (2021)

23. Pierazzi, F., Pendlebury, F., Cortellazzi, J., Cavallaro, L.: Intriguing properties of adversarial ml attacks in the problem space. 2020 IEEE Symposium on Security and Privacy (SP) (2020)
24. Qiu, S., Liu, Q., Zhou, S., Wu, C.: Review of artificial intelligence adversarial attack and defense technologies. Applied Sciences (2019)
25. Ren, K., Zheng, T., Qin, Z., Liu, X.: Adversarial Attacks and Defenses in Deep Learning. Engineering (2020)
26. Ribeiro, M.T., Singh, S., Guestrin, C.: "why should i trust you?": Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '16, Association for Computing Machinery (2016)
27. Sharafaldin, I., Habibi Lashkari, A., Ghorbani, A.A.: A detailed analysis of the cicids2017 data set. In: Mori, P., Furnell, S., Camp, O. (eds.) Information Systems Security and Privacy. pp. 172–188. Springer International Publishing (2019)
28. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: ICISSP (2018)
29. Shu, D., Leslie, N.O., Kamhoua, C.A., Tucker, C.S.: Generative adversarial attacks against intrusion detection systems using active learning. In: Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning. WiseML '20, Association for Computing Machinery, New York, NY, USA (2020)
30. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing properties of neural networks. In: ICLR (2014)
31. Teuffenbach, M., Piatkowska, E., Smith, P.: Subverting network intrusion detection: Crafting adversarial examples accounting for domain-specific constraints. In: Holzinger, A., Kieseberg, P., Tjoa, A.M., Weippl, E. (eds.) Machine Learning and Knowledge Extraction. pp. 301–320. Springer International Publishing (2020)
32. Usama, M., Asim, M., Latif, S., Qadir, J., Ala-Al-Fuqaha: Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems. IWCMC (2019)
33. Usama, M., Qayyum, A., Qadir, J., Al-Fuqaha, A.: Black-box adversarial machine learning attack on network traffic classification. In: 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC) (2019)
34. Vinayakumar, R., Alazab, M., Soman, K.P., Poornachandran, P., Al-Nemrat, A., Venkatraman, S.: Deep learning approach for intelligent intrusion detection system. IEEE Access (2019)
35. Wang, Z.: Deep learning-based intrusion detection with adversaries. IEEE Access (2018)
36. Xuan Qui, C.P., Hong Quang, D., Duy, P.T., Thi Thu Hien, D., Pham, V.H.: Strengthening ids against evasion attacks with gan-based adversarial samples in sdn-enabled network. In: 2021 RIVF International Conference on Computing and Communication Technologies (RIVF) (2021)
37. Yang, K., Liu, J., Zhang, C., Fang, Y.: Adversarial examples against the deep learning based network intrusion detection systems. In: MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM) (2018)
38. Zhao, S., Li, J., Wang, J., Zhang, Z., Zhu, L., Zhang, Y.: attackgan: Adversarial attack against black-box ids using generative adversarial networks. Procedia Computer Science (2021)
39. Zhong, Y., Zhu, Y., Wang, Z., Yin, X., Shi, X., Li, K.: An adversarial learning model for intrusion detection in real complex network environments. In: Yu, D., Dressler, F., Yu, J. (eds.) Wireless Algorithms, Systems, and Applications. pp. 794–806. Springer International Publishing (2020)