Hardening Machine Learning based Network Intrusion Detection Systems with Synthetic NetFlows

Andrea Venturi[1,*,†], Dimitri Galli[1,†], Dario Stabili[2] and Mirco Marchetti[1]

[1]University of Modena and Reggio Emilia, Department of Engineering "Enzo Ferrari", 41125 Modena, Italy
[2]University of Bologna, Department of Computer Science and Engineering, 40126 Bologna, Italy

Abstract

Modern Network Intrusion Detection Systems (NIDS) involve Machine Learning (ML) algorithms to automate the detection process. Although this integration has significantly enhanced their efficiency, ML models have been found vulnerable to adversarial attacks, which alter the input data to fool the detectors into producing a misclassification. Among the proposed countermeasures, adversarial training appears to be the most promising technique; however, it demands a large number of adversarial samples, which typically have to be manually produced. We overcome this limitation by introducing a novel methodology that employs a Graph AutoEncoder (GAE) to generate synthetic traffic records automatically. By design, the generated samples exhibit alterations in the attributes compared to the original netflows, making them suitable for use as adversarial samples during the adversarial training procedure. By injecting the generated samples into the training set, we obtain hardened detectors with better resilience to adversarial attacks. Our experimental campaign based on a public dataset of real enterprise network traffic also demonstrates that the proposed method even improves the detection rates of the hardened detectors in non-adversarial settings.

Keywords

ML-based NIDS, Graph Neural Network, Data Augmentation, Adversarial Training

1. Introduction

Modern Network Intrusion Detection Systems (NIDS) are increasingly relying on some Machine Learning (ML) mechanisms to automate their tasks [1]. Nevertheless, despite the proven efficiency of ML algorithms in detecting malicious activities [2], there is also a large research trend evidencing how ML-based NIDS are also affected by severe vulnerabilities. Among these, the most prominent threat is posed by *adversarial attacks*, which consist of small perturbations applied to the input samples to fool the detector into producing an incorrect classification [3].

As the cost of even a single misclassification is extremely high in security-related contexts, it is essential to devise valid countermeasures against adversarial attacks. Yet, no existing solution completely addresses all types of attacks and is suitable for all the possible models [4]. In this paper, we propose a novel methodology to improve the robustness of ML-based NIDS

ITASEC 2024: The Italian Conference on CyberSecurity, April 09–11, 2024, Salerno, Italy
*Corresponding author.
†These authors contributed equally.
andrea.venturi@unimore.it (A. Venturi); dimitri.galli@unimore.it (D. Galli); dario.stabili@unibo.it (D. Stabili); mirco.marchetti@unimore.it (M. Marchetti)
0000-0003-3822-968X (A. Venturi); 0009-0006-0280-2498 (D. Galli); 0000-0001-6850-334X (D. Stabili); 0000-0002-7408-6906 (M. Marchetti)

against adversarial attacks. Our approach leverages the concept of *adversarial training*, in which adversarial records are incorporated into the training set of vulnerable ML-based NIDS. By re-training them on the resulting augmented dataset, the obtained hardened ML-based NIDS can detect even the perturbed samples.

Despite its efficacy, even adversarial training is affected by several limitations as (i) it requires a large number of adversarial samples typically produced manually; (ii) the adversarial records injected into the training set must be representative of all the possible perturbations that an attacker could apply; (iii) the hardened classifiers are often subject to a decrease in the performance in non-adversarial contexts. We overcome these limitations by proposing an original approach for generating synthetic network flows automatically, leveraging a novel architecture that extends the *Adversarially Regularized Graph AutoEncoder* (ARGA) model [5]. We hypothesize that the samples generated through our procedure can be treated as candidate adversarial attacks suitable for the adversarial training procedure, thus avoiding the expensive and error-prone manual production of a sufficient number of representative adversarial samples.

We apply our methodology to an experimental case study, considering a state-of-the-art ML-based NIDS trained on a publicly available dataset. We verify our hypothesis, demonstrating that the proposed approach leads to hardened detectors that are more resilient when facing adversarial attacks than the baseline original versions. As an additional positive effect, we highlight that the hardened detectors also show improved performance in non-adversarial contexts, overcoming the most important limitation of adversarial training techniques.

The remainder of the paper is organized as follows. Section 2 introduces adversarial attacks and compares our paper against related work. Section 3 describes the proposed methodology. Section 4 defines the case study used for performance evaluation. Section 5 presents and discusses all the experimental results. Section 6 draws the conclusions with some final remarks.

2. Background and Related Work

ML-based NIDS have demonstrated high efficacy in identifying malicious activities with minimal false positives. These tools usually analyze network traffic in the form of *netflows*, which are tabular data structures summarizing the characteristics of the communication between two hosts in a network with metadata and statistics (e.g., duration of the transmission, number of exchanged bytes). Network flows are extremely popular in network intrusion detection as they allow fast analysis of the traffic data without incurring storage or privacy issues [6].

Despite the benefits provided by the introduction of ML algorithms in the detection processes, they are also shown to be vulnerable to adversarial attacks [7, 8]. In this paper, we focus on adversarial attacks based on evasive samples, where an attacker applies smart perturbations to the features of a malicious netflow so that it is incorrectly classified as benign by a trained ML-based NIDS. To obtain a suitable adversarial attack, an attacker must solve the following minimization problem [9]:

$$\arg\min_{\epsilon} f(x') \neq f(x), \quad \text{with} \quad x' = x + \epsilon \tag{1}$$

where x is the original sample, $f(x)$ denotes the output of the ML model f computed on x and ϵ is the perturbation added to x to create the adversarial sample x', which leads to a misclassification.

Adversarial attacks are particularly critical in security contexts where a single mistake can have significant consequences for an organization [10]. Previous research on adversarial attacks against ML-based NIDS leveraging manipulated samples has shown that even minimal perturbations can drastically decrease the performance of ML detectors [11].

Some countermeasures exist, but the solutions are still at an early stage, especially in network intrusion detection [4]. Among existing strategies in the literature, adversarial training is indicated as one of the most effective [12]. The idea is to train ML algorithms over augmented datasets that include adversarial perturbed records [13]. However, this process requires the generation of many adversarial samples to be effective. In addition, adversarially manipulated samples to be injected into the dataset must be representative of a wide range of attacks.

We overcome these limitations by proposing a method based on a novel Graph AutoEncoder (GAE) architecture that is able to generate automatically a large number of potential adversarial samples that can be used in the adversarial training procedure. The GAE allows us to take into account the network topology in the generation process. Since graph representation can express the complex dependencies among different malicious transmissions, the generative process leveraging the Graph Neural Network (GNN) is proven to be more effective.

Similar approaches exist in the literature. The papers in [14] and [15] represent our closest related work. In particular, the authors of [14] adopt an architecture based on a Generative Adversarial Network (GAN) to automatically generate adversarial examples that, when included in the training set, lead to good robustness. However, they do not evaluate the performance of the hardened detector in non-adversarial contexts. This limitation is also present in other similar papers (e.g., [16, 17]). Instead, we evaluate the hardened classifiers even in normal scenarios, and we demonstrate that our approach leads to increased detection rates. We remark that other countermeasures are affected by a drop in performance when no adversarial attack occurs (e.g., [9, 18, 19]). On the other hand, in [15], a framework based on Deep Reinforcement Learning (DRL) is employed to perturb samples to obtain an adversarial evasion, and the results are used as a means for adversarial training. With respect to this solution, we do not generate evasive samples explicitly, but rather we reconstruct netflows that are similar to the original ones that might not bypass the classifiers. As we will explain in Section 3, our approach exploits this similarity to enhance the detection performance of the ML-based NIDS, even in the absence of adversarial attacks.

3. Methodology

In this section, we present our novel methodology to strengthen the robustness of ML-based NIDS against adversarial attacks. The general idea is to follow a strategy based on adversarial training, in which the original ML detectors are re-trained over an augmented training set containing many adversarial samples. This process leads to the development of hardened classifiers, which are capable of correctly identifying adversarial attacks with high confidence.

As discussed before, adversarial training is among the most promising defensive techniques, but it has several limitations. Our approach overcomes them with an automatic generation of realistic — yet synthetic — netflows that, when injected into the training set of a ML-based NIDS, can enhance the detector's resilience to adversarial attacks. In particular, our methodology

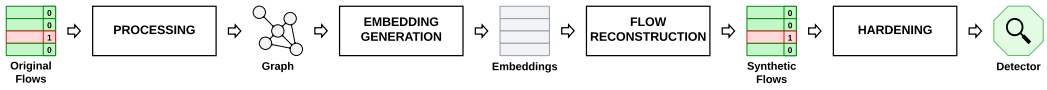

Figure 1: Methodology phases.

leverages an architecture based on ARGA [5], a recently proposed GNN for graph embedding. Our choice is driven by the ability of GAE models to also take into account the topology of the network when encoding the records. This allows the production of more accurate synthetic samples with respect to traditional autoencoders, which consider each data point individually [20].

The workflow of our methodology consists of four phases, as illustrated in Figure 1. The first phase, referred to as *processing*, converts the original netflow data into a graph representation. Hence, the *embedding generation* phase leverages the ARGA architecture, which accepts the graph as input and embeds both features and topological information of its components in compact vector representations. Then, in the *flow reconstruction* phase, these embeddings are used to reconstruct the traffic samples, making sure that they represent valid netflows. Finally, the *hardening* phase uses the reconstructed malicious flows as adversarial samples to inject into the training sets of ML-based NIDS during the adversarial training process for obtaining the hardened classifiers.

In the following, a more detailed explanation of the considered threat model and of the four phases of the proposed methodology.

3.1. Threat Model

We consider the same threat model previously proposed in [4]. In this scenario, a medium-sized enterprise with several internal hosts communicates with the internet through a single border router. The network is monitored using a ML-based NIDS, which analyzes traffic through netflow data from the border router. We suppose that some internal network hosts have been compromised by remote attackers capable of controlling them. Their purpose is to evade the detection of their malicious communications perturbing a close subset of features of the corresponding netflows, similarly as in previous proposals [11, 14]. For example, they can increment the duration of the malicious netflows by retarding the communications. Similarly, they can increment the number of bytes or packets by injecting junk data. Our proposal aims to devise a hardened version of the ML-based NIDS that is not susceptible to these kinds of modifications. We remark that our approach does not insert any novel component to the monitored network, and instead operates directly on the ML-based NIDS.

3.2. Processing

The first phase involves all the operations required to convert the network traffic into a suitable graph representation that can be submitted to the ARGA model in the subsequent phase.

A graph is formally defined as $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ denotes the set of vertices and $\mathbf{E}$ represents the set of edges. Assuming that the network traffic is provided in the form of netflows $\mathbf{D}$, which are the most common data representation in network intrusion detection, it can be naturally converted into a graph. A common approach associates the endpoints present in the netflows to the nodes, while the edges correspond to every netflow of the dataset. In other words, an edge links together the two endpoints appearing in a netflow as the source and destination of the communication and stores its features. This type of graph is denoted as *flow graph* $\mathbf{G_D}$ [21].

With flow graphs, the flow generation task requires a GAE capable of reconstructing edge features. Nevertheless, it is important to remark that most GNN in the literature are designed to perform tasks at the node level rather than on edges [22]. This motivates us to consider a dual graph representation named *line graph* $L(\mathbf{G_D})$. Line graphs can be obtained from flow graphs through a linearization procedure [23]. In particular, the edges in $\mathbf{G_D}$ are translated to the nodes of $L(\mathbf{G_D})$. Then, two nodes in $L(\mathbf{G_D})$ are connected if the corresponding edges in $\mathbf{G_D}$ share a common endpoint. This representation allows us to convert the task of generating new samples from an edge generation problem in $\mathbf{G_D}$ to a node generation problem in $L(\mathbf{G_D})$, with the advantage of exploiting the most advanced GNN architectures.

3.3. Embedding Generation

After having obtained a suitable graph representation, we proceed to generate synthetic netflow data through our innovative architecture. We logically divide this core operation into two separated — yet extremely connected — phases: *embedding generation* and *flow reconstruction*. The former stage aims to produce vector representations of the nodes in the line graph in a low-dimensional space, which will be then utilized to reconstruct the original records in the latter stage.

For the first step, we use the Adversarially Regularized Graph Autoencoder (ARGA) model [5], which is an unsupervised model that has been recently adopted in network intrusion detection with evident success [24]. In particular, ARGA generates an embedded representation for each node of $L(\mathbf{G_D})$. As a thorough description of the architecture is out of the scope of this paper, we outline here the fundamental principles of its design and operation.

The architecture of ARGA consists of two main modules: a *GAE* and an *adversarial module*.

The GAE generates the embeddings $\mathbf{Z}$, and is composed of an encoder and a decoder. The encoder accepts as input the adjacency matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$ of $L(\mathbf{G_D})$, which is a square matrix that expresses the connections for each of the n nodes, and the content matrix $\mathbf{X} \in \mathbb{R}^{n \times f}$ of $L(\mathbf{G_D})$, in which each row is the f-dimensional feature vector of each node. The encoder produces the embeddings $\mathbf{Z} \in \mathbb{R}^{n \times d}$ through a Graph Convolutional Network (GCN) [25], encoding both topological information about the neighborhood of a node (expressed by matrix A) and its netflow features (expressed by matrix X) into a single vector of dimensions $d << f$. The decoder takes $\mathbf{Z}$ as input and aims to reconstruct the adjacency matrix $\mathbf{A}$, leveraging another GCN. The training process essentially aims to minimize the reconstruction error between the original adjacency matrix $\mathbf{A}$ and the reconstructed one $\mathbf{A'}$. In other words, throughout training, the encoder learns to generate embeddings with increasing quality, while the decoder exploits the information encoded in the embeddings to reconstruct $\mathbf{A}$ with high fidelity.

The adversarial module employs a *Multi-Layer Perceptron* (MLP) to recognize whether the

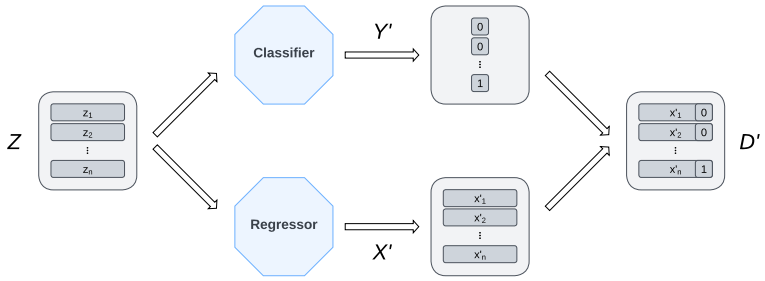

Figure 2: Architecture for the reconstruction of the original samples.

latent variable belongs to a real prior distribution (e.g., a Gaussian distribution) or is generated by the generator (i.e., the encoder). By training the GAE and the adversarial module together, it is possible to force the latent variables to follow a particular probability distribution, enhancing their robustness even in the case of noisy and sparse input data [5].

3.4. Flow Reconstruction

The ARGA model is capable of reconstructing the $\mathbf{A}$ matrix. Nevertheless, we are interested in producing a set of synthetic netflows to inject into the training set of ML-based NIDS.

To this purpose, we propose an original architecture based on two independent components: a classifier and a regressor, as illustrated in Figure 2. Both components take the embeddings generated in the previous phase as input. However, they serve different goals: the regressor aims to reconstruct the features $\mathbf{X'}$ of the initial netflows, while the classifier is used to reproduce the respective labels $\mathbf{Y'}$. We then merge the two parts to obtain the reconstructed dataset $\mathbf{D'} =< \mathbf{X'}, \mathbf{Y'} >$.

We remark that even the original paper for ARGA proposes a variant model, named ARGA_AX, that was able to reconstruct the content matrix [5]. However, two main reasons guided our choice of a novel architecture based on a separate classifier and regressor. First, the original methodology does not involve any label reconstruction. On the other hand, for our purposes, it is crucial to consider also the labels in the reconstruction process, as we aim to harden supervised ML-based NIDS. Second, we note that there exist numerous papers indicating that GCN-based decoders are not effective when reconstructing the features of the original samples (e.g., [26, 27]). We confirm this limitation of GCN architectures in our preliminary experimental campaign before the production of this paper.

The training workflow for this stage is similar to the one used for the decoder of the GAE with both the components trying to minimize the reconstruction error between the original records $\mathbf{D}$ and the reproduced samples $\mathbf{D'}$. In particular, the regressor minimizes the error between $\mathbf{X}$ and $\mathbf{X'}$, while the classifier minimizes the difference between $\mathbf{Y}$ and $\mathbf{Y'}$. This translates to a generation of a new set of synthetic netflows that can be related to the original ones using the following equation:

$$\mathbf{D'} = \mathbf{D} + \epsilon \tag{2}$$

where $\mathbf{D}$ and $\mathbf{D'}$ refer to the initial set of netflows provided as input to the procedure and the output of the entire process, respectively, and the ϵ represents the error introduced by our architecture. Our intuition is that the perturbation produced by an attacker when devising adversarial evasion samples (Equation 1) can be interpreted as the ϵ in Equation 2. Hence, the netflows in $\mathbf{D'}$ represent adversarial attack candidates that could evade a given ML-based NIDS. In this sense, we hypothesize that we can employ them to perform adversarial training.

We remark, however, that a careless generation procedure can lead to an incorrect set of synthetic netflows with unfeasible feature values, whose introduction in a training set could corrupt the learning procedure of the ML-based NIDS. For this reason, we verify that the generated netflows present features that are feasible in practice. For example, we impose that the regressor must not produce negative values for numerical features (e.g., duration, number of exchanged bytes). Similarly, we exclude from $\mathbf{D'}$ the netflows presenting values outside the corresponding feature domain or with wrong values in the derived features (e.g., bytes per second). In this way, we make sure that the reconstructed records are representative of realistic samples and can thus be safely injected into the training set.

3.5. Hardening

The last phase involves all the operations necessary to perform adversarial training on a target ML-based NIDS. Supposing that the original ML-based NIDS has been trained on the dataset $\mathbf{D}$, we re-train it over an augmented dataset $\mathbf{D}^*$ that is formed by merging together the original records $\mathbf{D}$ and the generated samples $\mathbf{D'}$. Our hypothesis is the same as in adversarial training: because the dataset $\mathbf{D'}$ includes perturbed netflows acting as adversarial samples, the final hardened ML-based NIDS is capable of detecting even adversarial attacks. However, as the samples in the set $\mathbf{D'}$ are designed to be similar to the original ones in the set $\mathbf{D}$, our adversarial training procedure can enhance the detection performance in the absence of adversarial attacks.

4. Case Study

In this section, we describe the experimental case study in which we apply our approach to enhance the resilience of an ML-based NIDS at the state-of-the-art. We present the considered dataset and the details of the implementation of the methodology[1].

4.1. Dataset

We base our case study on a widely used dataset for network intrusion detection: ToN-IoT [28].

ToN-IoT is a collection containing telemetry data of IoT/IIoT sensors, operating systems logs, and network traffic traces. For our purposes, we consider only the reduced network traces, which include various cyberattacks and normal traffic from an enterprise IoT network.

We apply common preprocessing operations as in [24]. We consider *Backdoor*, *DDoS*, *DoS*, *Injection*, *Password*, *Ransomware*, *Scanning*, and *XSS* attacks in our evaluation, excluding those with a too small number of netflows that would prevent a valid performance assessment. Hence,

[1]Source code available at https://github.com/dimgalli/hardening-ml-nids.git.

we build eight separate collections of data where each contains 200 000 normal records and 10 000 attack samples, to maintain a realistic 20:1 benign:malicious ratio. We refer to each dataset as $\mathbf{D}_\alpha$, where α indicates the type of attack considered. These will be used as input to our flow generation methodology, and for training and evaluating the ML-based NIDS (Section 4.2).

4.2. Implementation

We now detail the implementation of the introduced methodology, starting from the detector target of our analysis, and then passing to each proposed phase.

4.2.1. Detector

In order to obtain the ML-based NIDS target of our analysis, we follow the best practices considering binary classifiers tailored to detect specific attack variants, rather than a multi-class solution [29].

We use *Random Forest* (RF) classifiers, as related literature indicates that this algorithm is among the best-performing models for network intrusion detection tasks [30]. We utilize the same parameters as in previous work for adversarial attacks against ML-based NIDS [31].

We train two versions of each RF instance. A first baseline version is trained using the original samples provided in the considered dataset $\mathbf{D}_\alpha$. Instead, a second hardened version is trained using the augmented dataset $\mathbf{D}^*_\alpha$ as described in Section 3.5, and as we will detail in Section 4.2.3. The training-test split used to form the respective sets for both versions of each RF instance is 80-20, regardless of the considered dataset (original or augmented). To simulate realistic scenarios, we maintain a benign:malicious ratio of 20:1 in both training and test sets [32].

4.2.2. Flow Generation

During the processing stage, we build a line graph $L(\mathbf{G}_{D_\alpha})$ from the dataset $\mathbf{D}_\alpha$ described in Section 4.1. The purpose is to train an instance of our model that is tailored to produce netflows from each specific attack in ToN-IoT.

The embedding generation procedure accepts as input $L(\mathbf{G}_{D_\alpha})$ obtained in the processing stage and uses the ARGA model to produce the encoded variables $\mathbf{Z}$. In our case study, we choose the ARVGA_GD variant [5]. The architecture includes a variational encoder and a decoder with two consecutive GCN layers. The discriminator is built on a standard MLP with one hidden layer and one output neuron. We keep the same parameters as in the original paper.

As classifier and regressor for the flow reconstruction stage, we again rely on the RF algorithm. Both the RF classifier and the RF regressor have a number of trees equal to 10, while the criterion to measure the quality of splits is the *Gini impurity* for the classification model and the *mean squared error* for the regression model. In the end, we obtain as output the synthetic flows $\mathbf{D}'_\alpha$.

4.2.3. Hardening

In the hardening stage, we use the generated netflows $\mathbf{D}'_\alpha$ to create the augmented dataset $\mathbf{D}^*_\alpha$.

In particular, we inject only the synthetic malicious flows of each $\mathbf{D}'_\alpha$ into the respective $\mathbf{D}_\alpha$ to obtain $\mathbf{D}^*_\alpha$. This is because attackers typically manipulate only malicious netflows to

evade detection. Hence, we are not interested in perturbations of benign data points. As this procedure doubles the number of malicious records in $\mathbf{D}^*_\alpha$ with respect to the ones present in $\mathbf{D}_\alpha$, we maintain a 20:1 benign:malicious ratio by inserting the necessary benign samples from the pool of available ones.

After having obtained the augmented dataset, we proceed to re-train the RF classifiers on the corresponding $\mathbf{D}^*_\alpha$ to create the hardened versions.

5. Evaluation and Results

In this section, we detail the experiments of our case study. We first introduce the evaluation scenarios, and then we present the results.

For each instance of the RF detectors (i.e., baseline or hardened), we consider two different evaluation scenarios: *standard evaluation* (Section 5.1) and *adversarial evaluation* (Section 5.2).

In the *standard evaluation* scenario, we aim to validate whether the considered classifiers exhibit good detection capabilities in the absence of adversarial attacks. Hence, we test each model using the corresponding original test set obtained from $\mathbf{D}_\alpha$. The goal of this experiment is twofold. First, it lets us assess the goodness of the baseline detectors. Second, it lets us verify that the defensive methodology that we propose does not produce hardened models with lower detection rates in non-adversarial settings with respect to the baseline versions. This is of crucial importance because otherwise, the defensive efforts to enhance resilience in the presence of adversarial attacks would be wasted by a lower level of robustness in normal conditions.

In the *adversarial evaluation* scenario, we assess the detection performance of the baseline and hardened detectors against adversarial attacks. In particular, we choose an attack strategy that has been demonstrated to evade RF-based NIDS in [11]. We remark that this strategy alters only the malicious netflows for each attack by manipulating combinations of different features with increasing perturbation steps. We choose to report the average performance score of each model among all the altered attributes and the perturbation steps.

We evaluate the classifiers by considering performance metrics that are largely employed in network intrusion detection. In particular, we use the *F1-score* for the standard evaluation, while we utilize the *Detection Rate* (*DR*) for the adversarial evaluation. The scores of these metrics range between 0 and 1, with higher values indicating better performance.

5.1. Standard Evaluation

We evaluate initially the baseline and the hardened classifiers in non-adversarial contexts. The results are presented in Table 1a, where each value represents the *F1-score* obtained by each instance of the baseline and hardened models against the corresponding test set. For each row, we use the bold to indicate the best value. The last row summarizes the scores of each instance with the average value among all models.

The results have a twofold value. First, we observe that the baseline classifiers obtain performance scores that are similar to those of the state-of-the-art [33, 34], with an average of 0.981. This is an indicator that the considered RF detectors represent a valid benchmark for our experiments. More notably, we note that, in this scenario, the hardened instances obtain scores that are even higher than baseline models, with an average of 0.987. We remark that the unique

Table 1
Performance scores of the baseline and hardened detectors in the two evaluation settings.

(a) Scores in standard evaluation.

Attack	*F1-score*	
	Baseline	*Hardened*
Backdoor	**1.000**	0.999
DDoS	0.981	**0.988**
DoS	0.995	**0.997**
Injection	0.991	**0.996**
Password	0.983	**0.987**
Ransomware	0.923	**0.939**
Scanning	0.998	**0.998**
XSS	0.978	**0.987**
avg	0.981	**0.987**

(b) Scores in adversarial evaluation.

Attack	*DR*	
	Baseline	*Hardened*
Backdoor	0.613	**0.747**
DDoS	0.781	**0.828**
DoS	0.465	**0.917**
Injection	0.887	**0.972**
Password	0.567	**0.624**
Ransomware	0.269	**0.587**
Scanning	0.727	**0.995**
XSS	0.925	**0.945**
avg	0.654	**0.827**

case in which the *F1-score* value for the hardened detector is lower than the baseline model is the *Backdoor* attack, but the drop is negligible (just 0.001), and the performance is maintained at top-level (0.999).

We can conclude that this experiment demonstrates that our approach devises hardened ML-based NIDS that are effective even in absence of adversarial attacks. This property is crucial for obtaining models that can be deployed in real-world scenarios.

5.2. Adversarial Evaluation

Then, we pass to assess the performance of the baseline and the hardened classifiers in the adversarial scenario. For this case, in Table 1b, we report the average performance scores obtained by each detector against the adversarial attacks produced following the strategy proposed in [11]. We use the Detection Rate (*DR* or Recall) as the performance metric.

We immediately notice that the baseline classifiers show insufficient performance in detecting adversarial samples, with an average *DR* of just 0.654. This denotes the lack of robustness of baseline detectors, and it is in line with the expectations from the state-of-the-art. On the contrary, we observe that hardened models are far more robust with an average *DR* of 0.827 (an increment of over 26%). Among the classifiers, we highlight those for the *DoS*, *Injection*, *Scanning* and *XSS* attacks that are even above a *DR* equals to 0.9. This positive result further testifies to the effectiveness of the proposed methodology as a valid countermeasure against adversarial attacks in network intrusion detection.

6. Conclusions

Adversarial attacks represent a serious menace to ML-based NIDS, as they allow skilled attackers to evade detection. Some countermeasures exist, but the research is still at an early stage. In this paper, we propose a novel approach to enhance the robustness of ML-based NIDS against adversarial attacks. In particular, we present a novel architecture based on a GAE that is able to

generate automatically synthetic netflows that we exploit as a means to adversarial training, a popular defensive technique in the literature. We apply our methodology to a case study involving a state-of-the-art ML-based NIDS trained on a public dataset containing real traffic traces. The results highlight the efficacy of our method in enhancing the robustness of the detectors when facing adversarial attacks. Moreover, we denote that our approach is capable of not reducing the performance of the hardened classifiers even in the absence of adversarial attacks, overcoming the limitations of traditional adversarial training scenarios.

References

[1] A. L. Buczak, E. Guven, A survey of data mining and machine learning methods for cyber security intrusion detection, IEEE Communications surveys & tutorials 18 (2015) 1153–1176.

[2] R. Sommer, V. Paxson, Outside the closed world: On using machine learning for network intrusion detection, in: 2010 IEEE symposium on security and privacy, IEEE, 2010, pp. 305–316.

[3] B. Biggio, F. Roli, Wild patterns: Ten years after the rise of adversarial machine learning, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018, pp. 2154–2156.

[4] G. Apruzzese, M. Andreolini, L. Ferretti, M. Marchetti, M. Colajanni, Modeling realistic adversarial attacks against network intrusion detection systems, Digital Threats: Research and Practice (DTRAP) 3 (2022) 1–19.

[5] S. Pan, R. Hu, S.-f. Fung, G. Long, J. Jiang, C. Zhang, Learning graph embedding with adversarial training methods, IEEE transactions on cybernetics 50 (2019) 2475–2487.

[6] A. Yehezkel, E. Elyashiv, O. Soffer, Network anomaly detection using transfer learning based on auto-encoders loss normalization, in: Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security, 2021, pp. 61–71.

[7] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, F. Roli, Evasion attacks against machine learning at test time, in: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13, Springer, 2013, pp. 387–402.

[8] A. Venturi, C. Zanasi, On the feasibility of adversarial machine learning in malware and network intrusion detection, in: 2021 IEEE 20th International Symposium on Network Computing and Applications (NCA), IEEE, 2021, pp. 1–8.

[9] K. Grosse, N. Papernot, P. Manoharan, M. Backes, P. McDaniel, Adversarial examples for malware detection, in: Computer Security–ESORICS 2017: 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II 22, Springer, 2017, pp. 62–79.

[10] K. Grosse, L. Bieringer, T. R. Besold, B. Biggio, K. Krombholz, Machine learning security in industry: A quantitative survey, IEEE Transactions on Information Forensics and Security 18 (2023) 1749–1762.

[11] G. Apruzzese, M. Colajanni, Evading botnet detectors based on flows and random forest

with adversarial samples, in: 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), IEEE, 2018, pp. 1–8.
[12] P. Maini, E. Wong, Z. Kolter, Adversarial robustness against the union of multiple perturbation models, in: International Conference on Machine Learning, PMLR, 2020, pp. 6640–6650.
[13] A. Kantchelian, J. D. Tygar, A. Joseph, Evasion and hardening of tree ensemble classifiers, in: International conference on machine learning, PMLR, 2016, pp. 2387–2396.
[14] M. Usama, M. Asim, S. Latif, J. Qadir, et al., Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems, in: 2019 15th international wireless communications & mobile computing conference (IWCMC), IEEE, 2019, pp. 78–83.
[15] G. Apruzzese, M. Andreolini, M. Marchetti, A. Venturi, M. Colajanni, Deep reinforcement adversarial learning against botnet evasion attacks, IEEE Transactions on Network and Service Management 17 (2020) 1975–1987.
[16] H. S. Anderson, J. Woodbridge, B. Filar, Deepdga: Adversarially-tuned domain generation and detection, in: Proceedings of the 2016 ACM workshop on artificial intelligence and security, 2016, pp. 13–21.
[17] Y. Ji, B. Bowman, H. H. Huang, Securing malware cognitive systems against adversarial attacks, in: 2019 IEEE international conference on cognitive computing (ICCC), IEEE, 2019, pp. 1–9.
[18] A. Al-Dujaili, A. Huang, E. Hemberg, U.-M. O'Reilly, Adversarial deep learning for robust detection of binary encoded malware, in: 2018 IEEE Security and Privacy Workshops (SPW), IEEE, 2018, pp. 76–82.
[19] G. Apruzzese, M. Colajanni, L. Ferretti, M. Marchetti, Addressing adversarial attacks against security systems based on machine learning, in: 2019 11th international conference on cyber conflict (CyCon), volume 900, IEEE, 2019, pp. 1–18.
[20] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE transactions on neural networks 20 (2008) 61–80.
[21] W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, M. Portmann, E-graphsage: A graph neural network based intrusion detection system for iot, in: NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, IEEE, 2022, pp. 1–9.
[22] S. Zhang, H. Tong, J. Xu, R. Maciejewski, Graph convolutional networks: a comprehensive review, Computational Social Networks 6 (2019) 1–23.
[23] F. Harary, R. Z. Norman, Some properties of line digraphs, Rendiconti del circolo matematico di palermo 9 (1960) 161–168.
[24] A. Venturi, M. Ferrari, M. Marchetti, M. Colajanni, Arganids: a novel network intrusion detection system based on adversarially regularized graph autoencoder, in: Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, 2023, pp. 1540–1548.
[25] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).
[26] J. Park, M. Lee, H. J. Chang, K. Lee, J. Y. Choi, Symmetric graph convolutional autoencoder for unsupervised graph representation learning, in: Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 6519–6528.
[27] M. Ma, S. Na, H. Wang, Aegcn: An autoencoder-constrained graph convolutional network,

Neurocomputing 432 (2021) 21–31.
[28] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, A. Anwar, Ton_iot telemetry dataset: A new generation dataset of iot and iiot for data-driven intrusion detection systems, IEEE Access 8 (2020) 165130–165150.
[29] M. Stevanovic, J. M. Pedersen, An analysis of network traffic classification for botnet detection, in: 2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA), IEEE, 2015, pp. 1–8.
[30] B. Abraham, A. Mandya, R. Bapat, F. Alali, D. E. Brown, M. Veeraraghavan, A comparison of machine learning approaches to detect botnet traffic, in: 2018 International Joint Conference on Neural Networks (IJCNN), IEEE, 2018, pp. 1–8.
[31] G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, M. Marchetti, On the effectiveness of machine and deep learning for cyber security, in: 2018 10th international conference on cyber Conflict (CyCon), IEEE, 2018, pp. 371–390.
[32] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, K. Rieck, Dos and don'ts of machine learning in computer security, in: 31st USENIX Security Symposium (USENIX Security 22), 2022, pp. 3971–3988.
[33] A. R. Gad, A. A. Nashat, T. M. Barkat, Intrusion detection system using machine learning for vehicular ad hoc networks based on ton-iot dataset, IEEE Access 9 (2021) 142206–142217.
[34] A. Sharma, H. Babbar, A. Sharma, Ton-iot: Detection of attacks on internet of things in vehicular networks, in: 2022 6th International Conference on Electronics, Communication and Aerospace Technology, IEEE, 2022, pp. 539–545.