Adaptative Perturbation Patterns: Realistic Adversarial Learning for Robust NIDS

João Vitorino [0000-0002-4968-3e53] *, Nuno Oliveira [0000-0002-5030-7751] and Isabel Praça [0000-0002-2519-9899] *

School of Engineering, Polytechnic of Porto (ISEP/IPP), Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development (GECAD), 4249-015 Porto, Portugal; icp@isep.ipp.pt
* Correspondence: jpmvo@isep.ipp.pt (J.V.); icp@isep.ipp.pt (I.P.)

Abstract: Adversarial attacks pose a major threat to machine learning and to the systems that rely on it. Nonetheless, adversarial examples cannot be freely generated for domains with tabular data, such as cybersecurity. This work establishes the fundamental constraint levels required to achieve realism and introduces the Adaptative Perturbation Pattern Method (A2PM) to fulfill these constraints in a gray-box setting. A2PM relies on pattern sequences that are independently adapted to the characteristics of each class to create valid and coherent data perturbations. The developed method was evaluated in a cybersecurity case study with two scenarios: Enterprise and Internet of Things (IoT) networks. Multilayer Perceptron (MLP) and Random Forest (RF) classifiers were created with regular and adversarial training, using the CIC-IDS2017 and IoT-23 datasets. In each scenario, targeted and untargeted attacks were performed against the classifiers, and the generated examples were compared with the original network traffic flows to assess their realism. The obtained results demonstrate that A2PM provides a time efficient generation of realistic adversarial examples, which can be advantageous for both adversarial training and attacks.

Keywords: realistic adversarial examples; adversarial attacks; adversarial training; machine learning; tabular data; intrusion detection

1. Introduction

Machine learning is transforming the way modern organizations operate. It can be used to automate and improve various business processes, ranging from the recognition of patterns and correlations to complex regression and classification tasks. However, adversarial attacks pose a major threat to machine learning models and to the systems that rely on them [1], [2]. To deceive a model into predicting incorrect results, adversarial examples can be generated by slightly modifying original data. Depending on the utilized method, the data perturbations can be created in one of three settings: black-, gray- and white-box. The first solely queries a model’s predictions, whereas the second may also require knowledge of its structure or the utilized feature set, and the latter needs full access to its internal parameters.

A model’s robustness against these examples can be improved by performing adversarial training, a process where the training data is augmented with examples generated by one or more attack methods [3], [4]. Nonetheless, if the examples are unrealistic within a given domain, adversarial training is counterproductive because it will only deteriorate a model’s generalization. This is a pertinent aspect of the cybersecurity domain, where an example generated for a Network-based Intrusion Detection System (NIDS) must resemble real network traffic. Otherwise, a model will never encounter such example in a real scenario because it cannot be transmitted through a computer network. Furthermore, if an adversarial example utilized by a cyber-attack can be transmitted but is incompatible with its malicious purpose, evading detection will be futile.
This work addressed the challenge of generating realistic adversarial examples for classification tasks on domains with tabular data. The main contributions are the establishment of the fundamental constraint levels required to achieve realism and the introduction of the Adaptative Perturbation Pattern Method (A2PM) to fulfill these constraints in a gray-box setting. The capabilities of the developed method were evaluated in a cybersecurity case study with two scenarios: Enterprise and Internet of Things (IoT) networks. It generated adversarial network traffic flows for multi-class classification by creating data perturbations in the original flows of the CIC-IDS2017 and IoT-23 datasets.

To analyze the effects of A2PM on the noticeably different prediction processes of an Artificial Neural Network (ANN) and a tree-based algorithm, it was applied to directly attack Multilayer Perceptron (MLP) and Random Forest (RF) models. A total of 4 MLP and 4 RF classifiers were created with regular and adversarial training, and both targeted and untargeted attacks were performed against them. To provide a thorough study, assessments of example realism and time consumption were performed by comparing the generated examples with the corresponding original flows and recording the time required for each A2PM iteration.

The present article is organized into multiple sections. Section 2 defines the fundamental constraint levels and provides a survey of previous work on adversarial examples. Section 3 describes the developed method and the key concepts it relies on. Section 4 presents the case study and an analysis of the obtained results. Finally, Section 5 addresses the main conclusions and future work.

2. Fundamental Constraints

In recent years, adversarial examples have drawn attention from a research perspective. However, since the focus has been the image classification domain, the generation of realistic examples for domains with tabular data remains a relatively unexplored topic.

The common adversarial approach is to exploit the internal gradients of an ANN in a white-box setting, creating unconstrained data perturbations [5]–[7]. Consequently, most state-of-the-art methods do not support other machine learning models nor other settings, which severely limits their applicability to other domains. This is a pertinent aspect of the cybersecurity domain, where white-box is a highly unlikely setting. Considering that a NIDS is developed in a secure context, an attacker will commonly face a black-box setting, or occasionally gray-box [8], [9].

The applicability of a method for adversarial training is significantly impacted by the models it supports, because it must be able to generate examples against the utilized model. Despite an adversarially robust generalization still being a challenge, significant progress has been made in ANN robustness research [10]–[13]. However, various other machine learning models can be used for a classification task. This is the case of network-based intrusion detection, where tree-based algorithms, such as RF, are remarkably well-established [14], [15]. They can achieve a reliable performance on regular network traffic, but their susceptibility to adversarial examples must not be disregarded. Hence, these algorithms can benefit from adversarial training and several defense strategies have been developed to intrinsically improve their robustness [16]–[19].

In addition to the setting and the supported models, a method must account for the realism of the generated examples. Martins et al. [20] performed a systematic review of recent developments in adversarial attacks and defenses for intrusion detection. The authors reviewed various strategies in distinct settings but noted that none performed an evaluation in a real scenario. Therefore, it is imperative to establish the fundamental constraints an example must comply with to be applicable to a real scenario on a domain with tabular data. We define two constraint levels:

1. **Domain constraints** – Specify the inherent structure of a domain;
2. **Class-specific constraints** – Specify the characteristics of a class.

To be valid on a given domain, an example can solely reach the first level. Nonetheless, full realism is only achieved when it is also coherent with the distinct characteristics
of its class, reaching the second. In a real scenario, each level will contain concrete constraints for the utilized data features. These can be divided into two types:

- **Intra-feature constraints** – Restrict the value of a single feature;
- **Inter-feature constraints** – Restrict the values of one or more features according to the values present in other features.

In a real computer network scenario, an example must fulfil the domain constraints of the utilized communication protocols and the class-specific constraints of each type of cyber-attack. Apruzzese et al. [8] proposed a taxonomy to evaluate the feasibility of an adversarial attack on a NIDS, based on access to the training data, knowledge of the model and feature set, reverse engineering and manipulation capabilities. It can provide valuable guidance to establish the concrete constraints of each level for a specific system.

Even though some methods attempt to address a few constraints, many exhibit a clear lack of realism. Table 1 summarizes the characteristics of the most relevant methods of the current literature. The keyword ‘PO’ corresponds to any model that can output class probabilities instead of hard class labels.

Table 1. Summary of relevant methods and addressed constraint levels.

| Method     | Setting   | Supported Models | Domain Constraints | Class-specific Constraints |
|------------|-----------|------------------|--------------------|---------------------------|
| FGSM [3]   | White-box | ANN              | ×                  | ×                         |
| C&W [21]   | White-box | ANN              | ×                  | ×                         |
| DeepFool [22] | White-box | ANN              | ×                  | ×                         |
| Houdini [23] | White-box | ANN              | ×                  | ×                         |
| StrAttack [24] | White-box | ANN              | ×                  | ×                         |
| ZOO [25]   | White-box | ANN              | ×                  | ×                         |
| JSMA [26]  | White-box | ANN              | ✓                  | ✓                         |
| OnePixel [27] | Black-box | PO               | ✓                  | ×                         |
| RL-S2V [28] | Black-box | PO               | ×                  | ×                         |
| BMI-FGSM [29] | Black-box | Any              | ×                  | ×                         |
| GAN [30]   | Black-box | Any              | ×                  | ×                         |
| WGAN [31]  | Black-box | Any              | ×                  | ×                         |
| Boundary [32] | Black-box | Any              | ×                  | ×                         |
| Query-Efficient [33] | Black-box | Any              | ×                  | ×                         |
| Polymorphic [34] | Gray-box | ANN              | ×                  | ✓                         |

Regarding the Jacobian-based Saliency Map Attack (JSMA) [26] and the OnePixel attack [27], their approaches could potentially preserve a domain structure. The first was developed to minimize the number of modified pixels in an image, requiring full access to an ANN, whereas the latter only modifies a single pixel, based on the class probabilities predicted by a model. These methods perturb the most appropriate features without affecting the remaining, which appears to be beneficial. However, validity cannot be ensured because the perturbations do not account for any intra or inter-feature constraints.

On the other hand, the Polymorphic attack [34] addresses the preservation of original class characteristics. Chauhan et al. developed it for the cybersecurity domain, to generate examples compatible with a cyber-attack’s purpose. The authors start by applying a feature selection algorithm to obtain the most relevant features for the distinction between benign network traffic and each cyber-attack. Then, the values of the remaining features, which are considered irrelevant for the classification, are perturbed by a Wasserstein Generative Adversarial Network (WGAN) [31]. On the condition that there are no class-specific constraints for the remaining features, this approach could improve the coherence of an example with its class. Nonetheless, due to the unconstrained perturbations created by WGAN, the domain structure is disregarded, which leads to invalid examples.
Overall, the current literature does not properly address the fundamental constraints of domains with tabular data, which inhibits both adversarial training and attacks. To the best of our knowledge, no previous work has introduced a method capable of generating realistic adversarial examples on tabular data.

3. Developed Method

A2PM was developed with the objective of generating adversarial examples that fulfill both domain and class-specific constraints. It benefits from a modular architecture to assign an independent sequence of adaptative perturbation patterns to each class, which analyze specific feature subsets to create valid and coherent data perturbations.

Even though it can be applied in a black-box setting, the most realistic examples are obtained in gray-box, with only knowledge of the feature set. To fully adjust it to a domain, A2PM only requires a simple base configuration for the creation of a pattern sequence. Afterwards, realistic examples can be generated from original data to perform adversarial training or to directly attack a classifier in an iterative process (Figure 1).

![Figure 1. Adaptative Perturbation Pattern Method (Business Process Model and Notation).](image)

The generated examples can be untargeted, to cause any misclassification, or targeted, seeking to reach a specific class. New data perturbations could be generated indefinitely, but it would be computationally expensive. Hence, early stopping is employed to end the attack when the latest iterations could not cause any further misclassifications.

Besides static scenarios where the full data is available, the developed method is also suitable for scenarios where it is provided over time. After the pattern sequences are created for an initial batch of data, these can be incrementally adapted to the characteristics of subsequent batches. If novel classes are provided, the base configuration is used to autonomously create their respective patterns.

The performed feature analysis relies on two key concepts: value intervals and value combinations. The following subsections detail the perturbation patterns built upon these concepts, as well the advantages of applying them in sequential order.

3.1. Interval Pattern

To perturb uncorrelated numerical variables, the main aspect to be considered is the interval of values each one can assume. This is an intra-feature constraint that can be fulfilled by enforcing minimum and maximum values.

The Interval pattern encapsulates a mechanism that records the valid intervals to create perturbations tailored to the characteristics of each feature (Figure 2). It has a configurable ‘probability to be applied’, in the (0,1] interval, which is used to randomly
determine if an individual feature will be perturbed or not. Additionally, it is also possible to specify only integer perturbations for specific features.

![Diagram of Interval Pattern](image)

**Figure 2.** Interval Pattern (Business Process Model and Notation).

Instead of a static interval, moving intervals can be utilized after the first batch to enable an incremental adaptation to new data, according to a configured momentum. For a given feature and a momentum $k \in [0, 1]$, the updated minimum $m_i$ and maximum $M_i$ of a batch $i$ are mathematically defined as:

$$m_i = m_{i-1} \ast k + \min(x_i) \ast (1 - k)$$

$$M_i = M_{i-1} \ast k + \max(x_i) \ast (1 - k)$$

where $\min(x_i)$ and $\max(x_i)$ are the actual minimum and maximum values of batch $i$.

Each perturbation is computed according to a randomly generated number and is affected by the current interval, which can be either static or moving. The random number $\varepsilon \in (0, 1]$ acts as a ratio to scale the interval. To restrict its possible values, it is generated within the standard range of $[0.1, 0.3]$, although other ranges can be configured. For a given feature, a perturbation $P_i$ of a batch $i$ can be represented as:

$$P_i = (M_i - m_i) \ast \varepsilon$$

After a perturbation is created, it is randomly added or subtracted to the original value. Exceptionally, if the original value is less or equal to the current minimum, it is always increased, and vice-versa. The resulting value is capped at the current interval to ensure it remains within the valid minimum and maximum values of that feature.

### 3.2. Combination Pattern

Regarding uncorrelated categorical variables, enforcing their limited set of qualitative values is the main intra-feature constraint. Therefore, the interval approach cannot be replicated even if they are encoded to a numerical form, and a straightforward solution can be recording each value a feature can assume. Nonetheless, the most pertinent aspect of perturbing tabular data is the correlation between multiple variables. Since the value present in a variable may influence the values used for other variables, there can be several inter-feature constraints. To improve over the previous solution and fulfill both types of constraints, several features can be combined into a single common record.

The Combination pattern records the valid combinations to perform a simultaneous and coherent perturbation of multiple features (Figure 3). It can be configured with locked features, whose values are used to find combinations for other features without being
modified. Due to the simultaneous perturbations, its ‘probability to be applied’, in the 
(0,1] interval, can affect several features.

Figure 3. Combination Pattern (Business Process Model and Notation).

Besides the initially recorded combinations, new data can provide additional possibilities. These can be merged with the previous or used as gradual updates. For a given feature and a momentum $k \in [0,1]$, the number of updated combinations $C_i$ of a batch $i$ is mathematically expressed as:

$$C_i = C_{i-1} * k + \text{unique}(x_i)$$

where $\text{unique}(x_i)$ is the number of unique combinations of batch $i$.

Each perturbation created by this pattern consists of a combination randomly selected from the current possibilities, considering the locked features. It directly replaces the original values, ensuring the features remain coherent.

3.3. Pattern Sequences

Domains with diverse constraints may require an aggregation of several Interval and Combination patterns, which can be performed by pattern sequences. Furthermore, the main advantage of applying multiple patterns in a sequential order is that it enables the fulfilment of countless inter-feature constraints of greater complexity. It is pertinent to note that all patterns in a sequence are adapted to the original data independently, to prevent any bias when recording its characteristics. Afterwards, the sequential order is enforced to create cumulative perturbations on that data.

To exemplify the benefits of using these sequences, a small but relatively complex domain will be established. It contains three nominal features, F0, F1 and F2, and two integer features, F3 and F4. For an adversarial example to be realistic within this domain, it must comply with the following constraints:

- F0 must always keep its original value;
- F1 and F4 can be modified but must have class-specific values;
- F2 and F3 can be modified but must have class-specific values, which are influenced by F0 and F1.

The base configuration corresponding to these constraints is:

1. **Combination pattern** – Modify [F1];
2. **Combination pattern** – Modify [F2, F3], Lock [F0, F1];
3. **Interval pattern** – Modify [F3, F4], Integer [F3, F4].

A2PM will assign each class to a pattern sequence, according to the feature indices of this base configuration. For this example, the ‘probability to be applied’ will be 1.0 for all patterns, to demonstrate all three cumulative perturbations (Figure 4).
The first perturbation created for each class is replacing \( F_1 \) with another valid qualitative value. Then, without modifying the original \( F_0 \) nor the new \( F_1 \), a valid combination is found for \( F_0, F_1, F_2 \) and \( F_3 \). Finally, the integer features \( F_3 \) and \( F_4 \) are perturbed according to their valid intervals. Regarding \( F_3 \), to ensure it remains coherent with \( F_0 \) and \( F_1 \), this perturbation is created on the value of the new combination.

4. Case Study

A case study was conducted to evaluate the capabilities of the developed method, as well as its suitability for multi-class classification on the cybersecurity domain. The effects of A2PM on the noticeably different prediction processes of an ANN and a tree-based algorithm were analyzed by performing targeted and untargeted attacks against MLP and RF classifiers. To perform assessments of example realism and time consumption, the generated examples were compared with the original data and the time required for each iteration was recorded.

Two scenarios were considered: Enterprise and IoT networks. For these scenarios, adversarial network traffic flows were generated using the original flows of the CIC-IDS2017 and the IoT-23 datasets, respectively. In addition to evaluating the robustness of models created with regular training, the effects of performing adversarial training with flows generated by A2PM were also analyzed.

The study was conducted on relatively common hardware: a machine with 16GB of RAM, an 8-core CPU, and a 6GB GPU. The implementation relied on the Python 3 programming language and several libraries: Numpy and Pandas for data preprocessing and manipulation, Tensorflow for the MLP models, and Scikit-learn for the RF models. The following subsections describe the most relevant aspects of the implementation and present an analysis of the results obtained for each scenario.

4.1. Datasets and Data Preprocessing

Both CIC-IDS2017 and IoT-23 are public datasets that contain multiple labeled captures of benign and malicious network flows. The recorded data is extremely valuable for intrusion detection because it includes various types of common cyber-attacks and manifests real network traffic patterns.

CIC-IDS2017 [35] consists of 7 captures of cyber-attacks performed on a standard enterprise computer network with 25 interacting users. It includes Denial-of-Service and Brute-Force attacks, which were recorded in July 2017 and are available at the Canadian Institute for Cybersecurity. In contrast, IoT-23 [36] is directed at the emerging IoT networks, with wireless communications between interconnected devices. It contains network traffic created by malware attacks targeting IoT devices between 2018 and 2019, divided into 23 captures and available at the Stratosphere Research Laboratory.
From each dataset, two captures were selected and merged, to be utilized for the corresponding scenario. Table 2 provides an overview of their characteristics, including the class proportions and the label of each class, either 'Benign' or a specific type of cyber-attack. The 'PartOfAHorizontalPortScan' label was shortened to 'POAHPS'.

Table 2. Main characteristics of utilized datasets.

| Scenario          | Dataset (Captures) | Total Samples | Class Samples | Class Label   |
|-------------------|--------------------|---------------|---------------|---------------|
| Enterprise Network| CIC-IDS2017 (Tuesday and Wednesday) | 1,138,612      | 873,066       | Benign        |
|                   |                    |               | 230,124       | Hulk          |
|                   |                    |               | 10,293        | GoldenEye     |
|                   |                    |               | 7,926         | FTP-Patator   |
|                   |                    |               | 5,897         | SSH-Patator   |
|                   |                    |               | 5,796         | Slowloris     |
|                   |                    |               | 5,499         | Slowhttptest  |
|                   |                    |               | 11            | Heartbleed    |
| IoT Network       | IoT-23 (1-1 and 34-1) | 1,031,893     | 539,587       | POAHPS        |
|                   |                    |               | 471,198       | Benign        |
|                   |                    |               | 14,394        | DDoS          |
|                   |                    |               | 6,714         | C&C           |

Before their data was usable, both datasets required a similar preprocessing stage. First, the features that did not provide any valuable information about a flow’s benign or malicious purpose, such as timestamps and IP addresses, were discarded. Then, the categorical features were converted to numeric values by performing one-hot encoding. Due to the high cardinality of these features, the very low frequency categories were aggregated into a single category designated as 'Other', to avoid encoding qualitative values that were present in almost no samples and therefore had a small relevance.

Finally, the holdout method was applied to randomly split the data into training and evaluation sets with 70% and 30% of the samples, respectively. To ensure that both sets preserved the original class proportions, the split was performed with stratification. The resulting CIC-IDS2017 sets were comprised of 8 imbalanced classes and 83 features, 58 numerical and 25 categorical, whereas the IoT-23 sets contained 4 imbalanced classes and approximately half the structure size, with 42 features, 8 numerical and 34 categorical.

4.2. Base Configurations

After the data preprocessing stage, the distinct characteristics of the datasets were analyzed to identify the concrete constraints required for each scenario and establish the base configurations for A2PM.

Regarding CIC-IDS2017, some numerical features had discrete values that could only have integer perturbations. Due to the correlation between the one-hot encoded categorical features, they required combined perturbations to be compatible with a valid flow. Additionally, to guarantee the coherence of a generated flow with its type of cyber-attack, the encoded features representing the utilized communication protocol could not be modified. Hence, the following configuration was used for the Enterprise scenario, after it was converted to the respective feature indices:

1. **Interval pattern** – Modify {numerical features}, Integer {discrete features};
2. **Combination pattern** – Modify {categorical features}, Lock {protocol}.

Despite the different features of IoT-23, it presented similar constraints. The main difference was that, in addition to the communication protocol, a generated flow had to be coherent with the application protocol as well, which was designated as the service. The base configuration utilized for the IoT scenario was:
1. **Interval pattern** – Modify [numerical features], Integer [discrete features];
2. **Combination pattern** – Modify [categorical features], Lock [protocol, service].

It is pertinent to note that, for the ‘Benign’ class, A2PM would only generate benign network traffic that could be misclassified as a cyber-attack. Therefore, the configurations were only applied to the malicious classes, to generate examples compatible with their malicious purposes. Furthermore, since the examples should resemble the original flows as much as possible, the ‘probability to be applied’ was 0.6 and 0.4 for the interval and combination patterns, respectively. These values were established to slightly prioritize the small-scale modifications of individual numerical features over the more significant modifications of combined categorical features.

### 4.3. Models and Fine-tuning

For each scenario, both regular and adversarial training were performed. The first utilized the original training sets, whereas the latter augmented the training data by including one adversarial example per malicious flow. To prevent any bias, these examples were generated using solely the training data. These distinct training approaches were used to create a total of 4 MLP and 4 RF models, as described below.

An MLP [37] is a feedforward ANN consisting of an input layer, an output layer and one or more hidden layers in between. Each layer can contain multiple nodes with forward connections to the nodes of the next layer. When utilized as a classifier, a class is chosen according to the activations of the output layer.

Due to the high computational cost of training an MLP, it was fine-tuned by a Bayesian optimization technique [38]. A validation set was created with 20% of a training set, which corresponds to 14% of the original samples. Since an MLP accounts for the loss of the training data, the tuning process sought to minimize the loss of the validation data. To prevent overfitting, early stopping was employed to end the training when this loss stabilized. Additionally, due to the class imbalance present in both datasets, the assigned class weights were inversely proportional to their frequency.

The fine-tuning led to a four-layered architecture for both training approaches. The hidden layers relied on the Rectified Linear Unit (ReLU) activation function and the dropout technique, which inherently prevents overfitting by randomly disregarding a certain percentage of the nodes during training. To perform multi-class classification, the output layer used the **Softmax** function. The MLP architecture for the Enterprise scenario was:

1. **Input layer** – 83 nodes, 512 batch size;
2. **Hidden layer** – 64 nodes, ReLU activation, 10% dropout;
3. **Hidden layer** – 32 nodes, ReLU activation, 10% dropout;
4. **Output layer** – 8 nodes, Softmax activation.

A similar architecture was utilized for the IoT scenario, although it presented a decreased batch size and an increased dropout:

1. **Input layer** – 42 nodes, 128 batch size;
2. **Hidden layer** – 32 nodes, ReLU activation, 20% dropout;
3. **Hidden layer** – 16 nodes, ReLU activation, 20% dropout;
4. **Output layer** – 4 nodes, Softmax activation.

The remaining parameters were common to both scenarios because of their equivalent classification tasks. Table 3 summarizes the MLP configuration.

| Parameter        | Value               |
|------------------|---------------------|
| Objective Loss   | Categorical Cross-Entropy |
| Optimizer        | Adam Algorithm       |
| Learning Rate    | 0.001                |
| Maximum Epochs   | 50                   |
| Class Weights    | Balanced             |
On the other hand, an RF [39] is an ensemble of decision trees, where each individual tree performs a prediction according to a different feature subset, and the most voted class is chosen. It is based on the wisdom of the crowd, the idea that a multitude of classifiers will collectively make better decisions than just one.

Since training an RF has a significantly lower computational cost, a 5-fold cross-validated grid search was performed with well-established hyperparameter combinations. In this process, 5 stratified subsets were created, each with 20% of a training set. Then, five distinct iterations were performed, each training a model with four subsets and evaluating it with the remaining one. Hence, the validation approach utilized for an MLP was replicated 5 times per combination. The macro-averaged F1-Score, which will be described in the next subsection, was selected as the metric to be maximized. Table 4 summarizes the optimized RF configuration, common to both scenarios and training approaches.

| Parameter                  | Value         |
|----------------------------|---------------|
| Splitting Criteria         | Gini Impurity |
| Number of Trees            | 100           |
| Maximum Depth of a Tree    | 32            |
| Minimum Samples in a Leaf  | 2             |
| Maximum Features           | $\sqrt{\text{Number of Features}}$ |
| Class Weights              | Balanced      |

### 4.4. Attacks and Evaluation Metrics

A2PM was applied to perform adversarial attacks against the fine-tuned models for a maximum of 50 iterations, using the data from the holdout evaluation sets. The attacks were untargeted, causing any misclassification of malicious flows to different classes, as well as targeted, seeking to misclassify malicious flows as the ‘Benign’ class.

To perform a trustworthy evaluation of the effects of the generated examples on a model’s performance, it was essential to select suitable metrics. The utilized metrics and their interpretation are briefly described below [40], [41].

The F1-Score calculates the harmonic mean of precision and recall of multiple classes, considering both false positives and false negatives. In the cybersecurity domain, a score of 100% indicates that all cyber-attacks are being correctly detected and there are no false alarms. To account for class imbalance, this metric can be macro-averaged, which gives minority classes the same relevance as the overrepresented. The resulting macro-averaged F1-Score is mathematically defined as:

$$ Macro-averaged \text{ F1-Score} = \frac{1}{C} \sum_{i=1}^{C} \frac{2 \cdot P_i \cdot R_i}{P_i + R_i} $$

where $P_i$ and $R_i$ are the precision and recall of class $i$, and $C$ is the number of classes.

Since untargeted attacks attempt to cause any misclassification from one class to another, they affect the precision and recall of different classes. Therefore, this is a reliable metric for the performance evaluation. Additionally, due to the multiple imbalanced classes present in both datasets, it is also the most suitable validation metric for the employed fine-tuning approach.

Nonetheless, it does not properly reflect the effects of targeted attacks, which seek to reach a specific class. For these attacks, a trustworthy evaluation must measure the proportion of samples from other classes that were misclassified as the target class, which is ‘Benign’ for both scenarios. This Misclassification Rate corresponds to the false negative rate, considering the target class as negative and the remaining classes as positive. The best possible value is 0% because it indicates that there are no misclassifications. This metric can be expressed as:
\[
\text{Misclassification Rate} = \frac{FN}{TP + FN}
\]

where \( FN \) is the number of false negatives, samples misclassified as the target class, and \( TP \) is the number of true positives, samples correctly classified as other classes.

4.5. Enterprise Scenario Results

In the Enterprise network scenario, A2PM generated adversarial network flows using the original flows of the CIC-IDS2017 dataset. The results obtained for the targeted and untargeted attacks were analyzed, and the realism of the generated examples and the time consumption of the performed iterations were assessed.

The untargeted attacks caused significant performance declines in the models created with a regular training approach. A single iteration lowered the macro-averaged F1-Score of both MLP and RF by more than 20%. Even though the RF was more affected in the subsequent iterations, the total decline of both models was approximately 79%. Hence, their inability to distinguish between the different classes evidences their inherent susceptibility to adversarial attacks on Enterprise networks.

In contrast, the models created with adversarial training preserved considerably higher scores, with a gradual decrease of less than 2% per iteration. Despite some examples still deceiving them into predicting incorrect classes, both models were able to learn the intricacies of each type of cyber-attack, which mitigated the effects of adversarial examples. The scores achieved by the RF were consistently higher than the MLP, indicating a better robustness (Figure 5).

![Figure 5. Untargeted attack results of Enterprise network scenario.](image)

Regarding the targeted attacks, the results obtained for the regular training exhibited different growths of the Misclassification Rate. The MLP reached 90% in the last iteration, whereas the RF only required 4 iterations to surpass it. Their very high rates of malicious flows predicted to be benign corroborates the susceptibility of both models.

However, when adversarial training was performed, significantly lower rates were achieved. By training with one generated example per malicious flow, the models successfully learned to detect cyber-attack variations. The adversarially trained RF stood out for preserving the 0.09% rate it obtained on the original data throughout the entire attack, which highlights its excellent generalization (Figure 6).
The generated examples were analyzed and compared with the corresponding original flows, considering the intricacies and malicious purposes of the cyber-attacks. To perform an unbiased assessment of example realism, a randomly generated number was used to select one example, which is detailed below.

The selected flow had the ‘Slowloris’ class label, corresponding to a Denial-of-Service attack that attempts to overwhelm a web server by opening multiple connections and maintaining them as long as possible [42]. The created data perturbations increased the total flow duration and the packet Inter-Arrival Time (IAT), while reducing the number of packets transmitted per second and their size. These modifications were mostly focused on enhancing time-related aspects of the cyber-attack, to prevent its detection. Hence, in addition to being valid network traffic that can be transmitted through a computer network, the adversarial example also remained coherent with its class. Table 5 provides an overview of the modified features.

Table 5. Modified features of an adversarial ‘Slowloris’ example.

| Feature                      | Original Value | Perturbed Value |
|------------------------------|----------------|-----------------|
| Flow duration                | 3,003,828      | 27,080,257      |
| Maximum forward IAT          | 2,003,994      | 4,640,482       |
| Standard deviation of flow IAT| 710,048        | 2,585,735       |
| Forward packets per second   | 18,451,990     | 1,501,914       |
| Minimum forward segment size | 40             | 36              |
| Forward bytes in initial window | 29,200        | 24,883          |

To analyze the time consumption of the developed method, the number of seconds required to generate the examples of each iteration were recorded and averaged. The generation was performed at rate of 10,000 examples per 1.7 seconds on the utilized hardware, which evidences the fast execution and scalability of the developed method. Therefore, A2PM provides a time efficient approach to both adversarial training and attacks performed on a NIDS in an Enterprise network.
4.6. IoT Scenario Results

In the IoT network scenario, the adversarial network flows were generated using the original flows of the IoT-23 dataset. The analysis performed for the previous scenario was replicated to provide a similar evaluation.

The untargeted attacks iteratively caused small decreases in the macro-averaged F1-Score of the models created with regular training. Despite the RF starting to stabilize from the fifth iteration forward, the MLP continued its decline for an additional 17%. This high difference suggests that RF, and possibly tree-based algorithms in general, have a better inherent robustness to adversarial examples of IoT network traffic.

Unlike in the previous scenario, adversarial training did not provide considerable improvements. Nonetheless, the models had fewer incorrect class predictions during most of the iterations, which indicates that the augmented training data still contributed to the creation of more robust models (Figure 7).

Regarding the targeted attacks, the regular training exhibited much slower growths than in the previous scenario. In the last iteration, the MLP only surpassed a Misclassification Rate of 55%, and the RF remained at approximately 14%. These rates evidence the decreased susceptibility of both models, especially the RF, to adversarial examples directly targeting the ‘Benign’ class.

Furthermore, with adversarial training, the models were able to reach even lower rates. Even though many examples still evaded detection by the MLP, the number of malicious flows predicted to be benign by the RF were significantly reduced. Hence, the latter successful detected most of the cyber-attack variations (Figure 8).

The randomly selected flow for the assessment of example realism had the ‘DDoS’ class label, which corresponds to a Distributed Denial-of-Service attack performed by the malwares recorded in the IoT-23 dataset. The created data perturbations replaced the encoded categorical features of the connection state and history with another valid combination, already utilized by other original flows of that class. Hence, the generated network flow example remained compatible with its intended malicious purpose, achieving realism. Table 6 provides an overview of the modified features.
5. Conclusions

This work established the domain and class-specific constraint levels, which an adversarial example must comply with to achieve realism on tabular data, and introduced A2PM to fulfil these constraints in a gray-box setting, with only knowledge of the feature set. The capabilities of the developed method were evaluated in a cybersecurity case study with two scenarios: Enterprise and IoT networks. MLP and RF classifiers were created with regular and adversarial training, using the network flows of the CIC-IDS2017 and IoT-23 datasets, and targeted and untargeted attacks were performed against them. For each scenario, the effects of the attacks were analyzed, and assessments of example realism and time consumption were performed.

The modular architecture of A2PM enabled it to create pattern sequences that were independently adapted to each type of cyber-attack, accounting for the concrete constraints of the utilized datasets. Both targeted and untargeted attacks successfully decreased the performance of all MLP and RF models, with significantly higher declines exhibited in the Enterprise scenario. Nonetheless, the inherent susceptibility of these models to adversarial examples was mitigated by augmenting their training data with one generated example per malicious flow.

Overall, the obtained results demonstrate that A2PM provides a time efficient generation of valid and coherent examples for network-based intrusion detection. Therefore, the developed method can be advantageous for adversarial attacks, to iteratively cause misclassifications, and adversarial training, to increase the robustness of a model.
In the future, the utilized patterns can be improved to enable the configuration of more complex intra and inter-feature constraints. Since it is currently necessary to use both Interval and Combination patterns to perturb correlated numerical features, a new pattern can be developed to address their required constraints. It is also pertinent to analyze other datasets and other domains. Besides contributing to robustness research, future case studies can further reduce the knowledge required to create realistic examples.

**Author Contributions:** Conceptualization, J.V., N.O. and I.P.; methodology, J.V. and N.O.; software, J.V.; validation, N.O. and I.P.; investigation, J.V. and I.P.; writing, J.V. and I.P.; supervision, I.P.; project administration, I.P.; funding acquisition, I.P. All authors have read and agreed to the published version of the manuscript.

**Funding:** The present work has received funding from the European Union’s Horizon 2020 research and innovation programme, under project SeCoIA (grant agreement no. 871967). This work has also received funding from the following projects: UIDP/00760/2020 and CyberFactory#1 (ref. NORTE-01-0247-FEDER-40124).

**Data Availability Statement:** Publicly available datasets were analyzed in this work. The data can be found at: [CIC-IDS2017](https://lisa.dsic.upv.es/datasets/cic-ids2017/), [IoT-23](https://www.norteknow.com/iot23/). A novel method was developed in this work. An implementation in the Python 3 programming language can be found at: [A2PM](https://github.com/A2PM).

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

**References**

1. C. Szegedy et al., “Intriguing properties of neural networks,” 2nd Int. Conf. Learn. Represent. ICLR 2014 - Conf. Track Proc., pp. 1–10, 2014.
2. European Union Agency for Cybersecurity, A. Malatras, and G. Dede, “AI Cybersecurity Challenges: Threat Landscape for Artificial Intelligence,” 2020. doi: 10.2824/238222.
3. I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc., pp. 1–11, 2015.
4. European Union Agency for Cybersecurity, A. Malatras, I. Agrafiotis, and M. Adamczyk, “Securing Machine Learning Algorithms,” 2022. doi: 10.2824/874249.
5. X. Yuan, P. He, Q. Zhu, and X. Li, “Adversarial Examples: Attacks and Defenses for Deep Learning,” IEEE Trans. neural networks Learn. Syst., vol. 30, no. 9, pp. 2805–2824, 2019, doi: 10.1109/TNNLS.2018.2886017.
6. N. Pitropakis, E. Panaousis, T. Giannetos, E. Anastasiadis, and G. Loukas, “A taxonomy and survey of attacks against machine learning,” Comput. Sci. Rev., vol. 34, p. 100199, 2019, doi: 10.1016/j.cosrev.2019.100199.
7. S. Qi, Q. Liu, S. Zhou, and C. Wu, “Review of Artificial Intelligence Adversarial Attack and Defense Technologies,” Appl. Sci., vol. 9, no. 5, 2019, doi: 10.3390/app9050909.
8. G. Apruzzese, M. Andreolini, L. Ferretti, M. Marchetti, and M. Colajanni, “Modeling Realistic Adversarial Attacks against Network Intrusion Detection Systems,” Digit. Threat. Res. Pract., vol. 1, no. 1, 2021, doi: 10.1145/3469659.
9. I. Corona, G. Giacinto, and F. Roli, “Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues,” Inf. Sci. (Ny), vol. 239, pp. 201–225, 2013, doi: 10.1016/j.ins.2013.03.022.
10. A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” 6th Int. Conf. Learn. Represent. ICLR 2018 - Conf. Track Proc., pp. 1–28, 2018.
11. L. Schmidt, K. Talwar, S. Santurkar, D. Tsipras, and A. Madry, “Adversarially robust generalization requires more data,” Adv. Neural Inf. Process. Syst., vol. 2018-Decem, no. NeurIPS, pp. 5014–5026, 2018.
12. Y. Ganin et al., “Domain-adversarial training of neural networks,” Adv. Comput. Vis. Pattern Recognit., no. 9783319583464, pp. 189–209, 2017, doi: 10.1007/978-3-319-58347-1_10.
13. F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, “Ensemble adversarial training: Attacks and defenses,” 6th Int. Conf. Learn. Represent. ICLR 2018 - Conf. Track Proc., pp. 1–22, 2018.
14. M. C. Belavagi and B. Muniyal, “Performance Evaluation of Supervised Machine Learning Algorithms for Intrusion Detection,” Procedia Comput. Sci., vol. 89, pp. 117–123, 2016, doi: 10.1016/j.procs.2016.06.016.
15. R. Primartha and B. A. Tama, “Anomaly detection using random forest: A performance revisited,” Proc. 2017 Int. Conf. Data Softw. Eng., pp. 1–6, 2018, doi: 10.1109/ICODSE.2017.8285847.
16. A. Kantchelian, J. D. Tygar, and A. D. Joseph, “Evasion and hardening of tree ensemble classifiers,” 33rd Int. Conf. Mach. Learn., vol. 5, pp. 3562–3573, 2016.
