Journal of
Sensor and Actuator Networks

MDPI

Article

A Multi-Agent Intrusion Detection System Optimized by a Deep Reinforcement Learning Approach with a Dataset Enlarged Using a Generative Model to Reduce the Bias Effect

Matthieu Mouyart, Guilherme Medeiros Machado and Jae-Yun Jun *

LyRIDS, ECE Paris, 10 rue Sextius Michel, 75015 Paris, France; matthieumouyart34@gmail.com (M.M.); gmedeirosmachado@ece.fr (G.M.M.)
* Correspondence: jaeyunjk@gmail.com

Abstract: Intrusion detection systems can defectively perform when they are adjusted with datasets that are unbalanced in terms of attack data and non-attack data. Most datasets contain more non-attack data than attack data, and this circumstance can introduce biases in intrusion detection systems, making them vulnerable to cyberattacks. As an approach to remedy this issue, we considered the Conditional Tabular Generative Adversarial Network (CTGAN), with its hyperparameters optimized using the tree-structured Parzen estimator (TPE), to balance an insider threat tabular dataset called the CMU-CERT, which is formed by discrete-value and continuous-value columns. We showed through this method that the mean absolute errors between the probability mass functions (PMFs) of the actual data and the PMFs of the data generated using the CTGAN can be relatively small. Then, from the optimized CTGAN, we generated synthetic insider threat data and combined them with the actual ones to balance the original dataset. We used the resulting dataset for an intrusion detection system implemented with the Adversarial Environment Reinforcement Learning (AE-RL) algorithm in a multi-agent framework formed by an attacker and a defender. We showed that the performance of detecting intrusions using the framework of the CTGAN and the AE-RL is significantly improved with respect to the case where the dataset is not balanced, giving an F1-score of 0.7617.

Keywords: cybersecurity; intrusion detection; insider threat; multi-agent system; generative adversarial network; deep reinforcement learning

Citation: Mouyart, M.; Medeiros Machado, G.; Jun, J.-Y. A Multi-Agent Intrusion Detection System Optimized by a Deep Reinforcement Learning Approach with a Dataset Enlarged Using a Generative Model to Reduce the Bias Effect. *J. Sens. Actuator Netw.* **2023**, *12*, 68. https://doi.org/10.3390/jsan12050068

Academic Editor: Mohamed Amine Ferrag

Received: 15 August 2023
Revised: 8 September 2023
Accepted: 12 September 2023
Published: 18 September 2023

1. Introduction

In daily life, any system that needs to be protected is vastly more exposed to non-attack scenarios than attack scenarios. This aspect can also be observed through the publicly available datasets [1], in which the number of non-attack data is much larger than that of attack data. Hence, these datasets are in some sense "biased" towards non-attacks. Some of these datasets are the CMU-CERT [2,3], NSL-KDD [4], CICIDS2017 [5], among others. In this work, we claim that balancing a dataset between the non-attack and attack data (i.e., by adding novel data from the minority group, whose probability distribution resembles the distribution of the actual data (of the minority group)) can reduce existing bias effects and make an intrusion detection system learn better its parameters to protect a target system.

Recently, Xu et al. [6] worked in the domain of modeling the probability distribution of rows in tabular data and generating realistic synthetic data. The authors propose the Conditional Tabular Generative Adversarial Network (CTGAN) for modeling the probability distribution of both continuous and discrete variables and for generating their respective realistic synthetic data, with a benchmark of seven simulated and eight real datasets. They compare the results obtained from CTGAN to those obtained with several Bayesian methods and show their outperformance over the benchmark models.

Caminero et al. [7] addressed the problem of searching for fast and robust algorithms that can detect and classify dangerous traffic in data networks in the face of threats that

quickly change with time. They proposed an intrusion detection model using the multi-agent deep reinforcement learning (MADRL) framework formed by an attacker and a defender in a competitive (i.e., adversarial) setting. Whereas the attacker tries to trick the defender by encouraging itself to choose attacks (i.e., actions) that the defender is unable to detect correctly, the defender tries to learn to correctly detect the attacks sent by the attacker. The proposed method is coined as the *Adversarial Environment Reinforcement Learning* (AE-RL), which is based on the Double Deep Q-network (DDQN) [8] with epsilon-greedy strategies and with the Huber loss function [9] for both the attacker and the defender to optimize their respective parameters. The Huber loss is defined as a quadratic loss function, up to a threshold, and beyond this threshold, it behaves linearly. The competition between the attacker and the defender is expressed by their respective reward functions. For each time step, the attacker receives a reward equivalent to +1 when the defender fails to classify well the label of a given input sample (which can correspond to a type of attack or to non-attack). Otherwise, the attacker receives zero as its reward. Contrarily, the defender receives a reward equivalent to +1 when it can correctly classify the label of a given input sample. Otherwise, it receives zero as its reward. The Q-network of the attacker consists of three layers (input, one hidden, and output), whereas the Q-network of the defender consists of five layers (input, three hidden, and output). The employed dataset for this work is NSL-KDD [4]. The same feature vectors are fed to each Q-network as input (of 122 dimensions, consisting of 41 features with one-hot encodings for some of these features) with 39 attack types, categorized into 4 attack categories. Hence, the output of the attacker's Q-network has 40 dimensions (i.e., 39 attack types, in addition to the normal type), whereas the output of the defender's network has 5 dimensions (i.e., 4 attack categories, in addition to the normal category).

Brown et al. [10] showed their efforts to classify cyberattacks using machine learning algorithms. The authors propose a malware classification system that is based on a random forest model. Since malware can interact with many parts of a system, the authors collected their data from system calls and labeled them by using a malware detection algorithm. Their experiments were performed in both low-activity and heavy-activity cloud environments. The results show their method is effective in classifying malware in low-activity environments, but it has poor performance to heavy-activity ones. One of the reasons for this poor performance is due to the increase in features and data. Their approach could be adapted to heavy-activity environments if the authors used a deep-learning-based model.

In the present work, we are particularly interested in resolving the problem of how to improve the performance of an intrusion detection system in the face of insider threats when the provided dataset is unbalanced within the framework of machine–environment interaction (i.e., multi-agent deep reinforcement learning (MADRL) environment). Inspired by the works of Xu et al. [6] and Caminero et al. [7], the approach that we take to address this problem is first to generate novel realistic synthetic data using the CTGAN (which helps to balance the dataset). The hyperparameters of CTGAN are optimized using the tree-structured Parzen estimator (TPE) approach [11]. The TPE is a Bayesian optimization that is often used to optimize the hyperparameters of machine learning algorithms. It maintains two surrogate models that consist of probability distributions of hyperparameters of both bad and good performances. What the TPE does is first define a concept called the *promisingness* as a value that is proportional to the ratio of the probability distribution of the hyperparameters that give good performance to the probability distribution of the hyperparameters that give bad performance. Then, the TPE attempts to choose the hyperparameter values that maximize this *promisingness*. The promissingness is also known as the *expected improvement*.

We then define (and optimize) an intrusion detection system (for insider threats) by modeling it as a machine–environment interaction through the means of the MADRL approach, in particular using the AE-RL method. Among many intrusion types, in the present work, we focus on insider threats, provided by the CMU-CERT database [2]. To the best of our knowledge, we have not found in the literature any work on designing

a framework that combines both the generation of attack data using the CTGAN and intrusion detection (in a multi-agent environment) with insider threats that belong to various scenarios that correspond to different malicious human behaviors.

From the approach that we propose in the present work, we have found that the probability mass function (PMF) of the synthetic insider threats that we generate using the CTGAN approach (with its hyperparameters optimized with the TPE algorithm) are close to the PMF of the actual CMU-CERT data. Also, we have found that when the dataset is balanced by incorporating the attacks generated by the CTGAN, the performance of the intrusion detection system can be significantly increased.

The work hereby presented is structured as follows. In Section 2, we show the works that are related to the one that we present hereby. In Section 3, we present the method that we propose in this work. In Section 4, we show the results that we obtained and discuss them. Finally, Section 5 shows the concluding remarks and possible future works.

2. Related Works

In this section, we will review existing research that uses GAN to address the issue of unbalanced data in detecting cyberattacks using reinforcement learning (RL) models. The literature in the field of cyberattack detection suggests that recurrent neural network (RNN) and RL approaches are not yet fully explored [12]. Our paper will specifically focus on insider threat detection (ITD) as a type of cyberattack detection.

2.1. Balancing Datasets Comprising Attacks and Non-Attacks

ITD involves cyberattacks from trusted entities within an organization [13]. Since such attacks happen within an organization, one can think the organization can easily prevent them by monitoring as many data sources as possible (i.e., networks, employees' computers, active directories, etc.). However, the literature shows that ITD is a challenging task because it can leave no footprint of the attack. The data are often unbalanced, and since the real-world data are protected, there is a lack of datasets for the ITD community [13–15].

In the literature, the GAN has been identified as a strategy to address unbalanced classes by generating synthetic samples. While the synthetic minority oversampling technique (SMOTE) and modified versions of the convolutional neural network (CNN) are also used for unbalanced data, the GAN has been proven to be a state-of-the-art approach [16]. Other strategies may be ineffective in extreme scenarios or costly for large datasets [16].

Yuan et al. [14] propose a framework for insider threat detection that performs three main activities: identify the anomalous behavior sequence of users, augment the dataset, and classify suspect behaviors. To identify the anomalous behaviors, an LSTM-autoencoder encodes the user behavior sequences, and then the anomalous behaviors are selected from the embedding space. A GAN generator learns a generative distribution that is close to the distribution of the actual anomalous user behavior, and the GAN converges when the discriminator is not capable of differentiating the synthetic data from the real data. The authors show that their framework outperforms three other baselines in terms of the area under the curve (AUC).

Gayathri et al. [13] propose the use of a conditional GAN (CGAN) to augment an insider threat detection dataset. The methodology is also based on three phases: detect the user behavior, augment the dataset, and classify the attacks. The authors affirm that the CGAN is better suited to the task of generating cyberattack data because it increases the diversity of the training set by including data from all classes. Their generator and discriminator are formed by three fully connected layers. The discriminator is regularized using a dropout of 0.2, and Adam and Leaky Relu are also used. Their results show that, by combining the CGAN with any other cyberattack classification, they outperformed existing methods in terms of precision, recall, F1-score, Cohen's Kappa, and Mathew's correlation coefficient (MCC).

Gayathri et al. [15] propose an end-to-end framework to classify insider threats that uses the CWGAN-GP model, which is a combination of the CGAN and Wasserstein GAN (WGAN) models. The WGAN was proposed to address the stability and convergence issues of the vanilla GAN by optimizing the Wasserstein-1 distance instead of the Jensen–Shannon divergence (JSD). WGAN-GP is an ameliorated version that uses gradient penalty instead of weight-clipping for better stabilization. The generator of the CWGAN-GP model has four layers with 20, 32, 64, and 100 nodes, respectively. Both the generator and the discriminator have dense layers. The activation function of the generator is leaky ReLU for the first three layers and linear activation for the output layer. The authors considered linear, non-linear, and ensemble classifiers to evaluate their performance. The CWGAN-GP model's results were compared to those obtained from the SMOTE and the CGAN, and it outperformed all the other models in terms of precision, recall, and F1-score.

Sharma et al. [17] studied the intrusion detection problem by generating synthetic attacks using the conditional generative adversarial network and by detecting these attacks using the XGBoost classifier [18] over two publicly available datasets: NSL-KDD [4] and CICIDS2017 [5].

On the other hand, we have also found in the literature a study that explores ChatGPT [19] to generate social engineering attacks, phishing attacks, and automated hacking, among other types of attack scenarios [20]. Then, the authors examine defense techniques that use generative artificial intelligence (GenAI) methods to improve security measures.

Xu et al. realized that tabular data usually consists of a mixture of continuous and discrete variables. While continuous variables may have multiple modes, discrete variables are sometimes imbalanced (i.e., their respective values are often not uniformly distributed along their ranges, but these values are concentrated for some subset of values), making the modeling of their probability distributions difficult, and therefore, the generation of the respective realistic synthetic data is difficult as well [6]. The CTGAN addresses the aforementioned issue by using *mode-specific normalization*, the *conditional generator*, and *training by sampling*. For our study, we work with the CMU-CERT insider threat dataset, which is a tabular dataset with columns that have both continuous and discrete values. Therefore, we have decided to employ the CTGAN to generate novel insider threats.

2.2. Designing Multi-Agent Intrusion Detection Systems

Aryal et al. [21] recently presented a survey on adversarial attacks for malware analysis. In this work, we are particularly interested in designing multi-agent intrusion detection systems.

Elderman et al. [22] set the problem of securing networks in an adversarial reinforcement learning framework formed by two agents (i.e., an attacker and a defender), in which each agent has the goal to win as many Markov games with incomplete information and stochastic elements. They showed that Monte Carlo learning with the softmax-exploration strategy was the most effective approach (among the methods that they considered) for their specific network scenario.

Sethi et al. [23] observed that existing intrusion detection systems (IDS) suffer to adapt to changing attacks or to unseen attacks, and for this reason, the networks that need to be secured remain vulnerable. The authors propose a new intrusion detection system based on the deep Q-network (DQN) [24] with attention mechanisms distributed over multiple agents, which are coordinated to provide an efficient and scalable IDS to detect and classify advanced network attacks. They used the NSL-KDD [4] and CICIDS2017 [5] datasets to test their model.

Jin et al. [25] addressed the issues of the low detection rate and the high false-positive rate observed while detecting intrusions in wireless sensor networks. They propose an intrusion detection scheme based on the use of both a multi-agent system and a node trust value. They use the notion of typical node trust attributes (based on the combination of the Beta distribution and a tolerance factor) and the Mahalanobis distance function to define the normal behaviors. They show that this scheme offers a higher detection rate

and a lower false-positive rate than other approaches, even in the presence of several types of intrusions.

Sadhasivan and Balasubramanian [26] addressed the problem of the fact that the arrivals of new attacks are not updated in the cyberattack dataset as quickly as needed, leading to poor detection accuracy. They propose an adaptive rule-based multi-agent intrusion detection system (ARMA-IDS) that improves detection performance by adaptively updating the attack information to the available database. In the proposed model, they combine rules and classification algorithms (such as AdaBoost [27] and JRip [28]) in a multi-agent framework to detect attacks that are available through benchmark datasets such as KDD Cup 1999 [29] and SCADA [30]. The authors show the effectiveness of the proposed model, which offers a high detection rate for both datasets.

Achbarou et al. [31] observed the fact that making IDS efficient is not easy in a distributed environment such as the cloud environment. This is an open problem, and there are no satisfactory solutions for the automated evaluation and cloud-security analysis. They also observe the positive features of the multi-agent paradigm, such as adaptability, collaboration, and distribution. From this observation, they propose a distributed IDS (DIDS) in a multi-agent framework to identify and prevent new and complex malicious attacks.

Suwannalai and Polprasert [32] worked on designing an anomaly-based network intrusion detection system using the Adversarial Environment Deep Q-Network (AE-DQN) over the NSL-KDD dataset, similar to what Caminero et al. [7] did.

All the approaches mentioned in this section do not balance the respective datasets but directly work with the original unbalanced datasets.

2.3. Designing Multi-Agent Intrusion Detection Systems with Balanced Datasets Comprising Attacks and Non-Attacks

Ma and Shi [33] observed the difficulty in detecting intrusions in dynamic environments with various types of fast-changing attacks, requiring novel, fast, and robust solutions to detect them. To address this issue, the authors propose a method called Adversarial Environment Reinforcement Learning with SMOTE (AESMOTE) for detecting intrusions. It consists of combining a reinforcement learning (RL) algorithm with a technique that balances a dataset comprising both attacks and non-attacks. On the one hand, the RL algorithm that they used is the one proposed by Caminero et al. (i.e., the Adversarial Environment Reinforcement Learning (AE-RL)) [7]. On the other hand, the data-balancing algorithm that they employed is the one proposed by Chawla et al. (i.e., the synthetic minority oversampling technique (SMOTE)) [34]. The proposed method is trained using the NSL-KDD datasets [4]. While the SMOTE can be used to balance a dataset (between attacks and non-attacks), the GAN has been proven to be a state-of-the-art approach [16].

The synthetic minority oversampling technique (SMOTE) is an algorithm proposed by Chawla et al., 2002 [34]. It allows the sampling of novel data from the minority group. This algorithm defines line segments that join each of the data points from the minority group with their nearest neighbors. Then, it randomly samples novel data points along these line segments until the total number of data points from the minority group becomes about the number of data points from the majority group. Other oversampling techniques can be randomly resampling techniques such as bootstrapping. As Gayathri et al. [15] showed in their work, the performance of insider threat generation using the conditional generative adversarial network (CGAN) is superior to the synthetic minority oversampling technique (SMOTE) and to other resampling techniques (such as bootstrapping). Hence, we chose the CTGAN algorithm to sample novel attack data to balance the overall dataset. Another aspect that differs from the work of Ma and Shi [33] is that we work with the CMU-CERT dataset (because we are interested in insider threat detection) instead of the NSL-KDD dataset.

3. Methodology

3.1. Dataset

The dataset used in this work is the CMU-CERT insider threat dataset version 4.2 (see Figure 1). It is a synthetic dataset proposed by the Computer Emergency and Response Team (CERT) division of Carnegie Mellon University (CMU) [2]. The insider threat dataset comes in various versions, having improvements in each new version. We decided to use the 4.2 version in this work because it is the version that contains the most cyberattacks proportionally to the normal data. It contains 30,602,325 entries in total, among which only 7623 entries are cyberattacks; this means that the percentage of cyberattacks is 0.025%. This dataset is composed of 5 different data sources, namely:

- *Logon* corresponds to session auditing logs inside an information system, containing the fields *id*, *date*, *user*, *pc*, and *activity* (*logon/logoff*). There are 427,628 entries, among which 198 are cyberattack data (0.046% of the total entries).
- *Device* corresponds to external device auditing logs inside an information system, containing the fields *id*, *date*, *user*, *pc*, and *activity* (*connect/disconnect*). There are 205,476 entries, among which 2786 are cyberattack data (1.37% of the total entries).
- *Http* corresponds to the HTTP requests made by the users inside an information system, containing the fields *id*, *date*, *user*, *pc*, *url*, and *content*. There are 28,438,284 entries, among which 3860 are cyberattack data (0.013% of the total entries).
- *Email* corresponds to the emails sent within an information system (which can be sent inside and outside the information system), containing the fields *id*, *date*, *user*, *pc*, *to*, *cc*, *bcc*, *from*, *size*, *attachment_count*, and *content*. There are 1,315,459 entries among which 469 are cyberattack data (0.035% of the total entries).
- *File* corresponds to a file auditing log inside an information system, containing the fields *id*, *date*, *user*, *pc*, *to*, *filename*, and *content*. There are 222,801 entries among which 10 are cyberattack data (0.004% of the total entries).

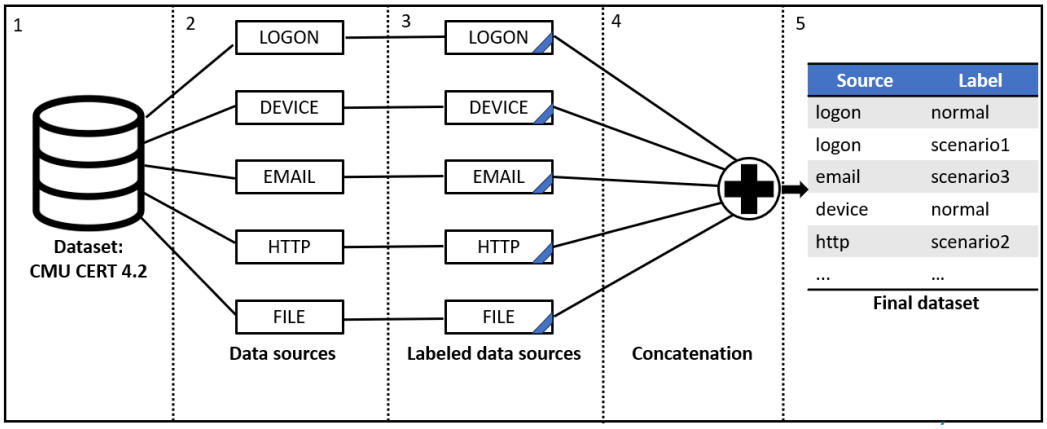


Figure 1. The adaptation procedure of the original dataset.

The dataset also contains lightweight directory access protocol (LDAP) logs of the 2 years covered by the dataset. These LDAP logs can be used to list which users are present in the considered company at any given time. The cyberattack data are generated by 70 users inside the company who act according to 3 scenarios:

- **Scenario 1**: A user who did not previously use removable drives or work after hours begins logging in after hours, using a removable drive, and uploading data to *wikileaks.org*. The user leaves the organization shortly thereafter.
- **Scenario 2**: A user begins surfing job websites and soliciting employment from a competitor. Before leaving the company, the user uses a thumb drive (at significantly higher rates than what the user used to do previously) to steal data.
- **Scenario 3**: A system administrator becomes disgruntled. He downloads a keylogger and uses a thumb drive to transfer it to his supervisor's machine. The next day, he

uses the collected keylogs to log in as his supervisor and send out an alarming mass email, causing panic in the organization. He leaves the organization immediately.

3.2. Data Pre-Processing

We need to pre-process this dataset to correctly use it. This pre-processing is performed in two steps. In the first step, we clean the data, extract the cyberattack data, as well as the normal data, and extract the features that we consider to be the most useful. In the second step, the resulting dataset is adapted for the framework of multi-agent reinforcement learning. To be able to discern normal data (i.e., non-attack data) from a cyberattack dataset, a folder named "answers" (which contains all the cyberattack identifications) is used. This folder contains all cyberattacks in the dataset, which are separated by the scenario associated. Each cyberattack can then be associated with its scenario and its data source. After the differentiation between normal and cyberattack data is performed, both categories of data can be extracted. Finally, for each data source (independently from their categories), a field selection step is performed such that only the most useful fields are left. The fields are chosen such that they are common in all data sources (see Algorithm 1 for details).

Algorithm 1 Data pre-processing

```
for each entry in the folder answer do
    Select the answer ID
    for each entry in the dataset do
        if the event ID equals the current answer ID then
            Get the associated data source
            Get the associated insider threat scenario
            Add the entry to the cyberattack dataset
        else
            Add the entry to the normal dataset
for each cyberattack do
    Keep User, PC, and Date fields
for each normal data do:
    Keep User, PC, Date fields
```

In the end, all the entries in the dataset are separated with respect to their categories (i.e., normal or cyberattack), and the respective fields are properly considered. The next step is the dataset adaptation for the multi-agent RL framework. To correctly adapt the dataset, it is required to concatenate all the entries into one file and label them according to their data source and their categories (i.e., normal or cyberattack). To do so, a field named 'label' is added to all the entries, which will correspond to the data source and its category. As an example, if the data source *logon* is associated with **Scenario 1**, the label that corresponds to each associated entry is "logon1" (see Algorithm 2 for details).

Algorithm 2 Data adaptation for multi-agent RL

```
for each cyberattack do
    Create column 'Label'
    Get the associated data source
    Get insider threat scenario associated
    Set label to be: data source + insider threat scenario number
for each normal data do
    Create column 'Label'
    Get the associated data source
    Get insider threat scenario associated
    Set label to be: data source
Concatenate normal data and cyberattacks
```

Hence, in this work, we considered four different scenarios (as described in Section 3.1) with their respective labels as follows:

- **Normal scenario** (i.e., non-attack scenario) with the following classes (i.e., labels): *logon*, *device*, *http*, *email*, *file*;
- **Scenario 1** with the following classes (i.e., labels): *logon1*, *device1*, *http1*;
- **Scenario 2** with the following classes (i.e., labels): *device2*, *http2*;
- **Scenario 3** with the following classes (i.e., labels): *logon3*, *device3*, *email3*, *http3*, *file3*.

3.3. Conditional Tabular Generative Adversarial Network (CTGAN)

In this work, we generate novel synthetic insider attacks using the Conditional Tabular Generative Adversarial Network (CTGAN) [6]. We train both an attack generator and an attack discriminator using the insider attacks that are present in the CMU-CERT dataset (version 4.2) [2] and pre-processed as indicated in Section 3.2. The CMU-CERT is a tabular dataset consisting of a mixture of continuous and discrete variables. In such a dataset, the continuous variables can have multiple modes, whereas the discrete variables are often imbalanced. This fact makes modeling their respective probability distributions difficult.

As explained by Xu et al. [6], the CTGAN addresses the aforementioned issue by using *mode-specific normalization*, the *conditional generator*, and *training by sampling*. First, the continuous variables are treated differently from the discrete variables. For each continuous column (from a data table), the variational Gaussian mixture model (VGM) [35] is used to estimate the number of modes and to fit a Gaussian mixture. In this way, the values of each continuous-value column can be represented with a one-hot vector (whose size depends on the number of modes and whose value depends on the mode to which a value belongs) and a scalar (whose value depends on the continuous value, and the mean and the standard deviation of the mode to which this continuous value belongs). Second, for each discrete-value column (from a data table), the probability mass function (PMF) of a row (formed by the discrete columns) is estimated by conditioning it with a value for each discrete-value column, which is sampled in turn from a PMF that is constructed as the logarithm of its frequency. Because of this conditioning procedure, the resulting method is called the conditional tabular GAN. Finally, the one-hot vectors and the scalar values that are needed to represent continuous-value columns and the one-hot vectors required to represent discrete-value columns can be generated using a GAN that consists of two fully connected hidden layers for both the generator and the discriminator (see Figure 2). In the generator network, the batch normalization and ReLU activation functions are used in the hidden layers, whereas in the output layer, the *tanh* activation function is used to learn to generate the scalar values for continuous-value columns, and the *Gumbel softmax* function [36] is used for one-hot vectors of both continuous-value columns and discrete-value columns. In the discriminator network, the *leaky ReLU* activation and *dropout* are used on each hidden layer. Finally, the CTGAN is trained using the *Wasserstein gradient penalty* (WGAN-GP) loss function [37] with the *Adam* optimizer [38].

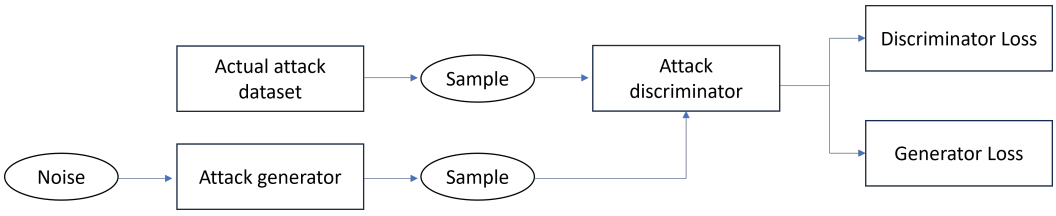


Figure 2. GAN paradigm.

The quality of the cyberattacks generated with the CTGAN model is evaluated by computing the mean absolute error between the probability mass function (PMF) of the real data and the PMF of the generated data. In order to generate realistic cyberattacks, it is required to split the cyberattacks with respect to their data source. Hence, a CTGAN model is trained for each data source, as shown in Figure 3. Further, the hyperparameters

of each CTGAN model (i.e., a CTGAN model per data source) are optimized using the tree-structured Parzen estimator (TPE) algorithm [11]. These hyperparameters are shown in Table 1.

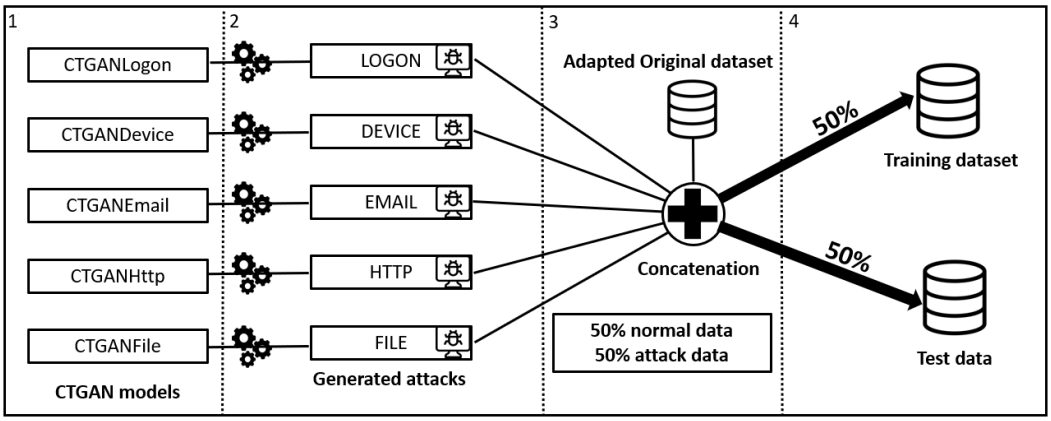


Figure 3. The generation of attacks using the CTGAN.

Table 1. The CTGAN's hyperparameters that are optimized using the TPE algorithm [11].

Hyperparameters	Meaning
Batch size	The size of the samples employed to optimize the GAN's parameters
Discriminator decay	A weight decay employed by the Adam optimizer for the discriminator neural network
Discriminator learning rate	The initial learning rate used by the Adam optimizer for the discriminator neural network
Discriminator steps	The steps for the discriminator neural network
Generator decay	A weight decay employed by the Adam optimizer for the generator neural network
Generator learning rate	The initial learning rate used by the Adam optimizer for the generator neural network
Log frequency	An indicator parameter for if the PMF is defined with the logarithmic function.

After the optimization steps, the CTGAN models can be used to generate high-quality cyberattacks (see Appendices A and B for details). Once the cyberattacks are generated, they can be concatenated with the real data to balance the overall dataset. Then, the resulting dataset can be split into training and test datasets. As indicated in Figure 3, the splitting factor is 50%, meaning that the original dataset is split into 50% for training the Adversarial Environment Reinforcement Learning (AE-RL) algorithm (which is described in Section 3.4) and 50% for testing this algorithm.

3.4. Adversarial Environment Reinforcement Learning (AE-RL)

As an approach to detect intrusions (in particular, insider threats), we used the approach proposed by Caminero et al. [7], which is the *Adversarial Environment Reinforcement Learning (AE-RL)* algorithm (see Figure 4). In this approach, the double deep Q-network (DDQN) proposed by van Hasselt et al. [8] is implemented within the multi-agent framework consisting of an attacking agent (that emulates an insider that threatens an organization's computer network) and a defending agent (that tries to identify the insider's threats) using the CMU-CERT pre-processed (see Section 3.2) and augmented (see Section 3.3).

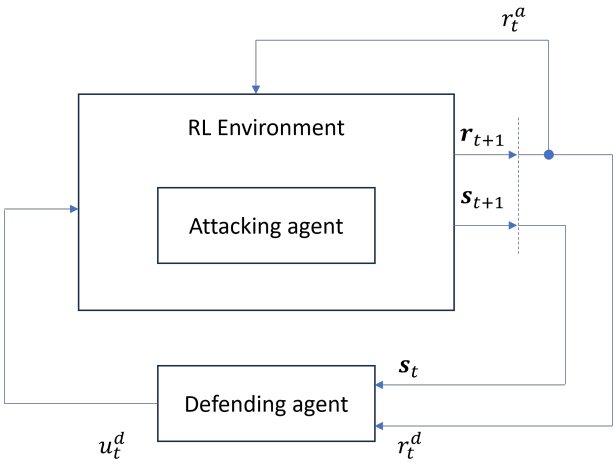


Figure 4. RL paradigm.

Each agent (both the attacker and the defender) tries to optimize its action within the framework of the Markov decision process (MDP). The action optimization is attempted by maximizing the action-value function, which is approximated by a deep neural network (DNN). The DNN for the attacker is illustrated in Figure 5a and the DNN for the defender in Figure 5b. The input features to each of these two DNNs are the same, while the output variables for the attacker's DNN are attack categories, and those for the defender's DNN are the considered scenarios. For both DNNs, the input variables can be viewed as the state variables, the output variables as actions, and the values returned for each of these actions as the values of the action-state function. The parameters of these DNNs are optimized by performing the backpropagation, which consists of updating the model parameters following the negative direction of the gradients of the respective loss function with respect to the parameters. The loss function is applying the Huber function [9] to

$$L_t = \left(Y_t^{\text{DoubleQ}} - Q^{\psi}\left(\mathbf{s}_t, u_t; \theta^{\psi}\right)\right)^2, \tag{1}$$

where $\psi \in \{\text{attacker, defender}\}$ and

$$Y_t^{\text{DoubleQ}} = R_t^{\psi} + \gamma Q^{\psi}\left(\mathbf{s}_{t+1}, \arg\max_{u^{\psi}} Q^{\psi}\left(\mathbf{s}_{s+1}, u^{\psi}; \theta^{\psi}\right); \widetilde{\theta}^{\psi}\right), \tag{2}$$

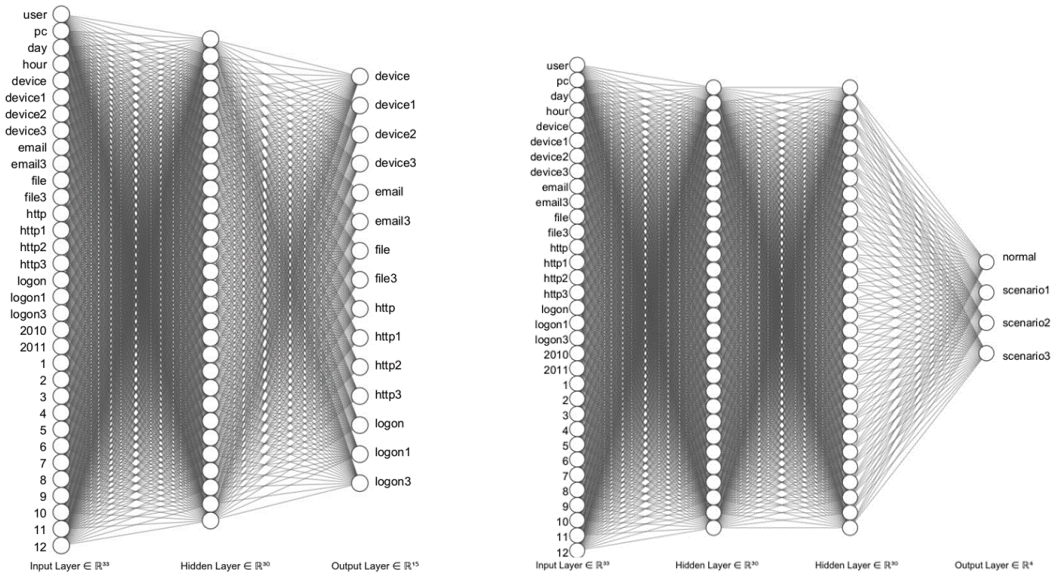
with γ being the discount factor, θ^{ψ} being the parameters of the main ψ-DNN, and $\widetilde{\theta}^{\psi}$ being the parameters of the target ψ-DNN.

As suggested Caminero et al. [7], the rewards of both the attacker and the defender at time instant t are defined as

$$r_t^{\text{attacker}} = \mathbb{I}\{y_t \neq \hat{y}_t\},$$
$$r_t^{\text{defender}} = \mathbb{I}\{y_t = \hat{y}_t\},$$

where $\mathbb{I}\{\cdot\}$ is the indicator function that indicates whether the considered condition is true or not, y_t is the attack generated by the attacker, and $\hat{y}_t$ is the attack that the defender estimates. Hence, if the defender correctly estimates the attack, the defender gains +1 as a reward, while the attacker obtains zero as its reward. Otherwise, the defender obtains zero reward, whereas the attacker gains +1 as its reward. As indicated by this definition of reward, on the one hand, the attacker gains a positive reward when it arrives to trick the defender with a particular attack that it chose. If this is the case, the attacker will try to exploit the knowledge that it gained about the defender's vulnerability and, in this way, maximize its expected return. The attacker also explores better policies (i.e., other types of attacks) that can harm the defender even more. On the other hand, the defender gains a positive reward when it arrives to correctly classify (i.e., detect) the attacker's action. If this

is the case, the defender will try to exploit the solution that it found to improve its ability to detect the attackers' intentions. The defender also explores other detection solutions to see if it can further improve its ability to detect the attacks. This adversarial environment makes both the attacker and the defender more intelligent in their functions, and as a result, the defender can improve its ability to detect insider threats.



(**a**) Attacker's neural network. (**b**) Defender's neural network.

Figure 5. Attacker's and defender's neural networks.

The Q-network of the attacker consists of three layers (input, one hidden, and output), whereas the Q-network of the defender consists of five layers (input, three hidden, and output). In terms of the RL hyperparameter values, we have used those that Caminero et al. used [7].

Finally, Figure 6 shows the overall framework that combines the CTGAN (Section 3.3) and the AE-RL (explained in this section).

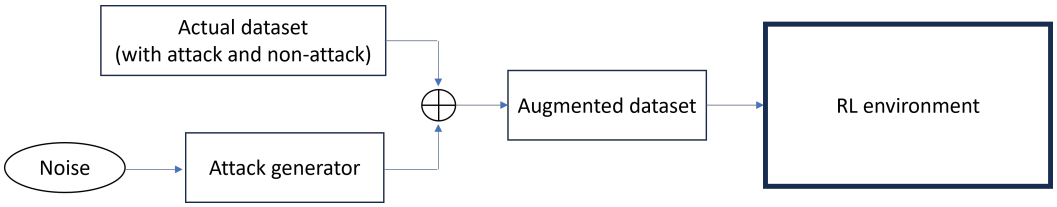


Figure 6. GAN-RL paradigm.

4. Results and Discussion

We implemented the framework formed by the CTGAN and the AE-RL shown in Figure 6 on a computer with the hardware and software specifications shown in Table 2.

As shown in Figure 3, the synthetic attacks are generated for each data source separately. Hence, we optimized the hyperparameters of the CTGAN for each of the five data sources (i.e., logon, device, http, email, and file) using the tree-structured Parzen estimator (TPE). Table 3 shows the seven hyperparameters of the CTGAN, the default hyperparameter values (which are the default values suggested by Xu et al. [6]), and the hyperparameter values optimized for each of the five data sources of the CMU-CERT dataset. The *batch size* refers to the size of the data that are considered to update the parameters of both the generator and discriminator's neural networks. Both the discriminator decay and the generator decay are the weight decays for the Adam optimizer that both the discriminator and generator's neural networks in order to update their respective parameters. These

weight decays are used for regularization purposes. Finally, both the discriminator learning rate and the generator learning rate are the initial learning rates that the Adam optimizer uses for the discriminator's and the generator's neural networks, respectively.

Table 2. The hardware and software specifications used to implement the framework of CTGAN and AE-RL shown in Figure 6.

Characteristics	Values
PC	HP-Z2-Tower-G4-Workstation
CPU	Intel® Core™ i7-8700 CPU @ 3.20 GHz × 12
RAM	15.5 GiB
GPU	NVIDIA GP106GL [Quadro P2200]
Disk	1.5 TB
OS	Ubuntu 20.04.6 LTS
Python IDE	Spyder 3.3.6
Python	3.8.10
Hyperopt	0.2.7
Keras	2.7.0
Tensorflow	2.7.0
Torch	1.8.0
CTGAN [6]	0.7.0

Table 3. Optimization of the CTGAN's hyperparameters.

Hyperparameters	Default	Logon CTGAN	Device CTGAN	Http CTGAN	Email CTGAN	File CTGAN
Batch size	500	390	450	600	90	60
Discriminator decay	$1e^{-6}$	$7.19e^{-3}$	$2.22e^{-3}$	$8.08e^{-3}$	$2.23e^{-3}$	$8.95e^{-3}$
Discriminator learning rate	$2e^{-4}$	$5.89e^{-3}$	$4.45e^{-3}$	$9.72e^{-3}$	$8.35e^{-3}$	$2.05e^{-5}$
Discriminator steps	1	8	2	2	7	7
Generator decay	$1e^{-6}$	$8.90e^{-3}$	$7.21e^{-3}$	$1.57e^{-3}$	$4.99e^{-3}$	$8.66e^{-3}$
Generator learning rate	$2e^{-4}$	$7.03e^{-3}$	$1.42e^{-5}$	$2.06e^{-5}$	$4.62e^{-3}$	$1.36e^{-3}$
Log frequency	True	True	True	True	True	False

Figure 7 shows the mean absolute error (MAE) between the probability mass function (PMF) of the actual attacks and the PMF of the attacks generated with the CTGAN, for five different data sources (i.e., logon, device, http, email, and file) and for two different settings: one with the default hyperparameters of the CTGAN and the other with optimized hyperparameters. These results show that the MAE values with the optimized hyperparameters are lower than those with the default hyperparameter values.

Figure 8 shows the convergence of the two agents (i.e., the attacker and the defender), which are trained both with and without data augmented using the CTGAN method. For each of the two plots in Figure 8, the horizontal axis represents the number of epochs (which ranges up to 5000), whereas the vertical axis represents the total rewards attained by each of the two agents in an epoch. One epoch consists of 100 steps. At each step, the attacker obtains +1 as its reward value if it successfully attacks the defender (i.e., if the defender does not detect the attack). Otherwise, it obtains zero reward. On the other hand, the defender obtains +1 as its reward value if it successfully defends the attack that comes from the attacker (i.e., the attacker fails to attack the defender). Otherwise, it obtains zero reward. Figure 8a shows the convergence curves without the data augmented with the CTGAN method, whereas Figure 8b shows the convergence curves with the data augmented with the CTGAN method.

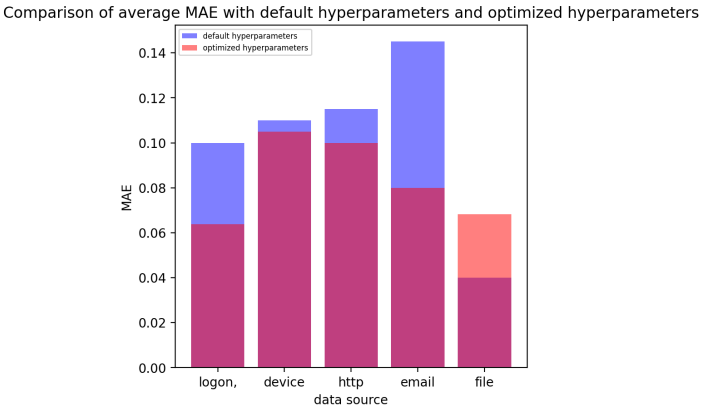
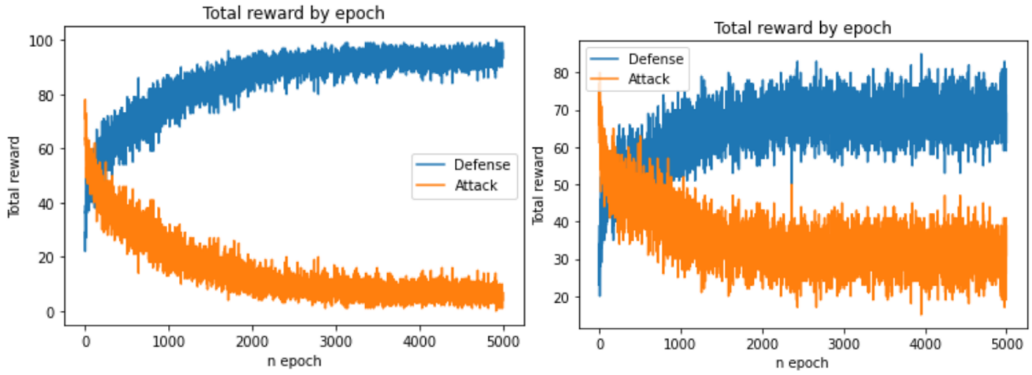


Figure 7. The comparison of mean absolute error (MAE) of the probability mass function (PMF) of the five activities that the CMU-CERT considers.



(**a**) No data augmentation. (**b**) With data augmentation.

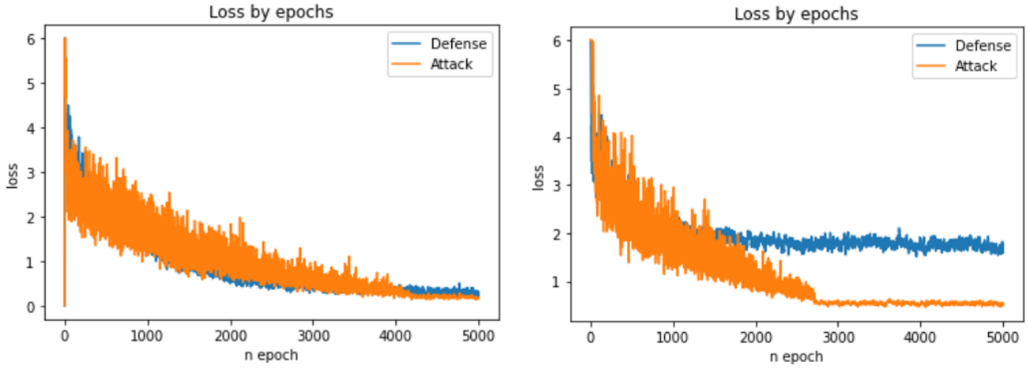
Figure 8. The convergence of the total rewards for both the attacker's and defender's neural networks.

For the case without data augmentation (Figure 8a), the attacker's total-reward curve starts with high values and decays to low values, whereas the defender's total-reward curve starts at relatively low values and increases to high values as the number of epochs augment. We think that these phenomena are observed because, as the number of attacks is small compared to the normal data in the original CMU-CERT dataset, the defender can quickly learn to defend itself from the attacks that originated by the attacker. Conversely, as the number of epochs advances, the attacker fails more often to attack the defender, which becomes able to defend its system more and more.

On the other hand, for the case with data augmentation (Figure 8b), we observe similar behaviors for both the attacker and the defender as in the case without data augmentation. However, the attacker does not reach values as low as those attained without data augmentation at the end of the experiment (i.e., at the maximal number of epochs). Similarly, the defender does not reach values as high as those attained without data augmentation at the end of the experiment. This may be explained by the fact that we augmented the original dataset with synthetically generated attacks. In this case, the defender does not succeed in defending itself as much as in the case without data (i.e., attack) augmentation. Hence, the defender struggles to learn to improve its defense system. As a result, the attacker's total rewards are higher than for the case without data augmentation. Further, both curves with data augmentation present more significant fluctuations than those observed in the case without data augmentation.

Figure 9 shows the Huber loss curves for both the attacker's and defender's neural networks when the dataset is augmented or not with the CTGAN. For each subfigure of Figure 9, the horizontal axis represents the number of epochs (ranging up to 5000), and the vertical axis represents the Huber loss. Recall that each epoch consists of 100 steps, and therefore, each of the loss values for each epoch corresponds to those of the last

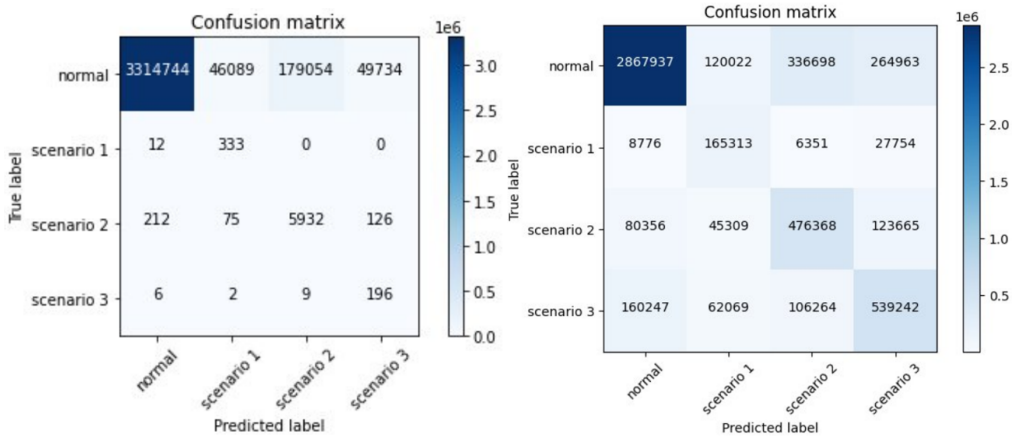
step. In particular, Figure 9a corresponds to the case without data augmentation. We observe that the loss curves of both the attacker and the defender decrease as the number of epochs increases, reaching a plateau with similar loss values. Similarly, Figure 9b shows the convergence of the Huber loss of both the attacker and the defender when the dataset is augmented with the attacks generated by the CTGAN. In this case, once again, both loss curves flatten as the number of epochs increases, but each of these two curves converge to some different values. In fact, the defender seemingly has difficulty improving its performance as its converged value is somewhat higher than that of the attacker. This phenomenon is aligned with the results observed in Figure 8b. As there are more attacks generated by the CTGAN, the defender seems to struggle more to correctly detect the attacks. This fact makes the attacker improve its performance, as shown in the corresponding figure.



(**a**) No data augmentation. (**b**) With data augmentation.

Figure 9. The convergence of the loss function for both the attacker's and defender's neural networks.

Figure 10 shows the confusion matrices for both when only the original CMU-CERT dataset is considered (see Figure 10a) and when this dataset is augmented with the attacks generated with the CTGAN method (see Figure 10b). First, the total number of data shown in the confusion matrices corresponding to the original CMU-CERT dataset is lower than that of the confusion matrix corresponding to the augmented dataset, due to the fact that we have more attacks available in the dataset augmented with the CTGAN. The ratio is about 67% (i.e., 3,596,524/5,391,334).



(**a**) No data augmentation. (**b**) With data augmentation.

Figure 10. Confusion matrices with and without data augmentation.

From the results shown in the respective confusion matrices (shown in Figure 10), we can compute the values of *true positive* (TP), *true negative* (TN), *false positive* (FP), and *false negative* (FN). We can achieve this from two different perspectives: the *multi-class classification*, and the *binary classification*. By the perspective of *multi-class classification*,

we mean that both the positive/negative predictions and true/false actual samples are regarded with respect to a considered scenario (among four scenarios: **Normal**, **Scenario 1**, **Scenario 2**, and **Scenario 3** (which are defined in Section 3.1)). On the other hand, by the perspective of *binary classification*, we mean that both the positive/negative predictions and true/false actual samples are regarded from the point of view of attack and non-attack. From each of these two perspectives, we first define the respective values of TP, TN, FP, and FN. Then, from these numbers, we define the respective *precision*, *recall*, and *F1-score* as follows to measure the performance of detecting intrusions (in our case, insider threats) with and without data augmentation:

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}, \qquad \text{Recall} = \frac{\text{TP}}{\text{TP+FN}}, \qquad \text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{3}$$

In Table 4, we show the performance results of the detection of insider threats in terms of F1-score, precision, and recall. We compute these performance values for both with and without data augmentation. Further, we analyze them from the two aforementioned perspectives: multi-class classification (for which four classes are considered (i.e., **Normal**, **Scenario 1**, **Scenario 2**, and **Scenario 3**)) and binary classification (for which two classes are considered (i.e., *attack* and *normal*)). We also show the performance differences between the unbalanced and balanced cases in percentage. These difference rates are computed as

$$\frac{(\text{quantity}_{\text{balanced}} - \text{quantity}_{\text{unbalanced}})}{\text{quantity}_{\text{unbalanced}}} \times 100\%. \tag{4}$$

Table 4. Classification performance regarded both as multi-class classification and as binary classification.

Scenario	F1-Score (Unbalanced)	F1-Score (Balanced)	Difference Rate (%)	Precision (Unbalanced)	Precision (Balanced)	Difference Rate (%)	Recall (Unbalanced)	Recall (Balanced)	Difference Rate (%)
Normal	0.9601	0.8552	−11	0.9999	0.9200	−8	0.9234	0.7990	−13
Scenario 1	0.0620	0.5915	853	0.0321	0.5643	1658	0.9349	0.6214	−34
Scenario 2	0.0142	0.5769	3 965	0.007	0.5146	7097	0.9652	0.6564	−14
Scenario 3	0.0078	0.5502	6 992	0.0039	0.4210	10,707	0.9202	0.7940	−23
Attack/Normal	**0.0463**	**0.7617**	**15**	**0.0237**	**0.6826**	**28**	**0.9667**	**0.8619**	**−11**

Let us first analyze the results shown in Table 4 from the perspective of multi-class classification. For the case of the unbalanced dataset, the F1-score values are remarkably bad in general for all scenarios except for the normal scenario. Considering (3), this is mainly due to bad precision values. Notice that the recall values are in general significantly good. There can be several reasons why the precision values can be bad. The first reason can be due to the fact that, while we optimized the hyperparameters of the CTGAN, we employed the AE-RL hyperparameter values that Caminero et al. used for their work. From the confusion matrix without data augmentation (see Figure 10a), we can observe that for *Scenario 1*, *Scenario 2*, and *Scenario 3*, very large numbers of normal (i.e., non-attack) data are predicted as attacks. These phenomena make the FP values large and, as a consequence, the precision values too. Another observation that we can make is the fact that the true normal (i.e., true non-attack) and true attack data are noticeably unbalanced. Their respective quantities are 3,589,621 and 6903. In other words, the true attacks are only about 0.2% of the total test dataset. This fact seems to influence the results obtained with the unbalanced dataset. On the other hand, for the case of the balanced dataset, the obtained F1-scores are significantly larger than those of the unbalanced case. This difference rate is more noticeable for the *normal* case. However, in general, these F1-score values are not high, and we think that this may be because we used the AE-RL hyperparameter values that are used by Caminero et al. for their work but not adapted to ours. Further, from (3), we can explain the gain of F1-scores for the balanced case. Although the corresponding recall values are decreased with respect to the unbalanced case, these losses are smaller than the significant gain obtained regarding the precision. These results have made a significant increase in

F1-score values. Moreover, from Figure 10b, we observe that the distribution of the true normal (i.e., true non-attack) data and true attack data are more balanced. Their respective quantities are 3,589,620 and 1,801,714. Hence, the size of the true attack data represents about 33% of the total test dataset (which means more balanced than the unbalanced case).

Now, let us analyze the results shown in Table 4 from the perspective of binary classification. These results are shown in the last row of the table. In this case, we observe that the F1-score for the unbalanced case is very low (0.0463), whereas for the balanced case, it is significantly higher (0.7617). Both values may be increased by adapting the AE-RL hyperparameter values. Further, their respective precision and recall values are also presented, together with their difference rates computed, as shown in (4).

5. Conclusions and Future Work

In this work, we considered the Adversarial Environment Reinforcement Learning (AE-RL) algorithm [7] to detect intrusions (in particular, insider threats) within a multi-agent framework formed by an attacker and a defender. This intrusion detection system is learned and evaluated with a publicly available dataset called the CMU-CERT version 4.2 [2]. Because this dataset is unbalanced between the non-attacks and attacks, we showed that the performance of the intrusion detection system (implemented by the AE-RL) is limited. To overcome this limitation, we considered the Conditional Tabular Generative Adversarial Network (CTGAN) [6], with its hyperparameters optimized using the tree-structured Parzen estimator (TPE) [11], to balance the tabular dataset (i.e., CMU-CERT). We showed that the mean absolute errors between the probability mass functions (PMFs) of the actual data and the PMFs of the data generated using the CTGAN are relatively small. Then, from the optimized CTGAN, we generated synthetic insider threats and combined them with the actual ones to balance the original dataset. Then, we used the resulting dataset to train and test the AE-RL (i.e., the intrusion detection system) and showed that the performance of detecting intrusions (through the F1-score) has significantly improved. However, we observe that there is still room to improve the performance of detecting intrusions with and without data augmentation, and we think that this may be because we optimized the CTGAN hyperparameters but have employed the AE-RL hyperparameter values that Caminero et al. used for their work. We will adapt them in the near future. Other possible future works are the following. In this work, we only considered a multi-agent system formed by one attacker and one defender. We may be able to scale up the intrusion detection system that we present in this work, by increasing the number of both attackers and defenders. Other future work can be used other than the CMU-CERT dataset for the framework that we presented in this work, such as the NSL-KDD [4] and CICIDS2017 [5]. Further, we possibly consider time-series data (instead of pointwise data) to detect intrusions, adapting the framework that we presented in this work. Moreover, in the future, we are also interested in designing an intrusion detection system (following the research direction presented in this work) that is able to detect zero-day unknown attacks.

Author Contributions: Conceptualization, M.M., G.M.M. and J.-Y.J.; Methodology, M.M., G.M.M. and J.-Y.J.; Software, M.M.; Validation, M.M., G.M.M. and J.-Y.J.; Formal analysis, M.M., G.M.M. and J.-Y.J.; Investigation, M.M., G.M.M. and J.-Y.J.; Resources, M.M.; Data curation, M.M.; Writing—original draft, M.M., G.M.M. and J.-Y.J.; Writing—review & editing, M.M., G.M.M. and J.-Y.J.; Visualization, M.M. and J.-Y.J.; Supervision, G.M.M. and J.-Y.J.; Project administration, G.M.M. and J.-Y.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data used in this paper is available in: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099, accessed on 1 August 2023.

Acknowledgments: At the time of the realization of this work, Matthieu Mouyart was a research intern at LyRIDS, ECE Paris, France, and his internship was funded by ECE Paris.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. The Comparison of the Probability Mass Functions (PMFs) of the Original CMU-CERT Attacks to the PMFs of the Attacks Generated with the CTGAN

Appendix A.1. Logon Attacks

Appendix A.1.1. For the Case of the Attacks Generated Using the CTGAN with Default Hyperparameter Values

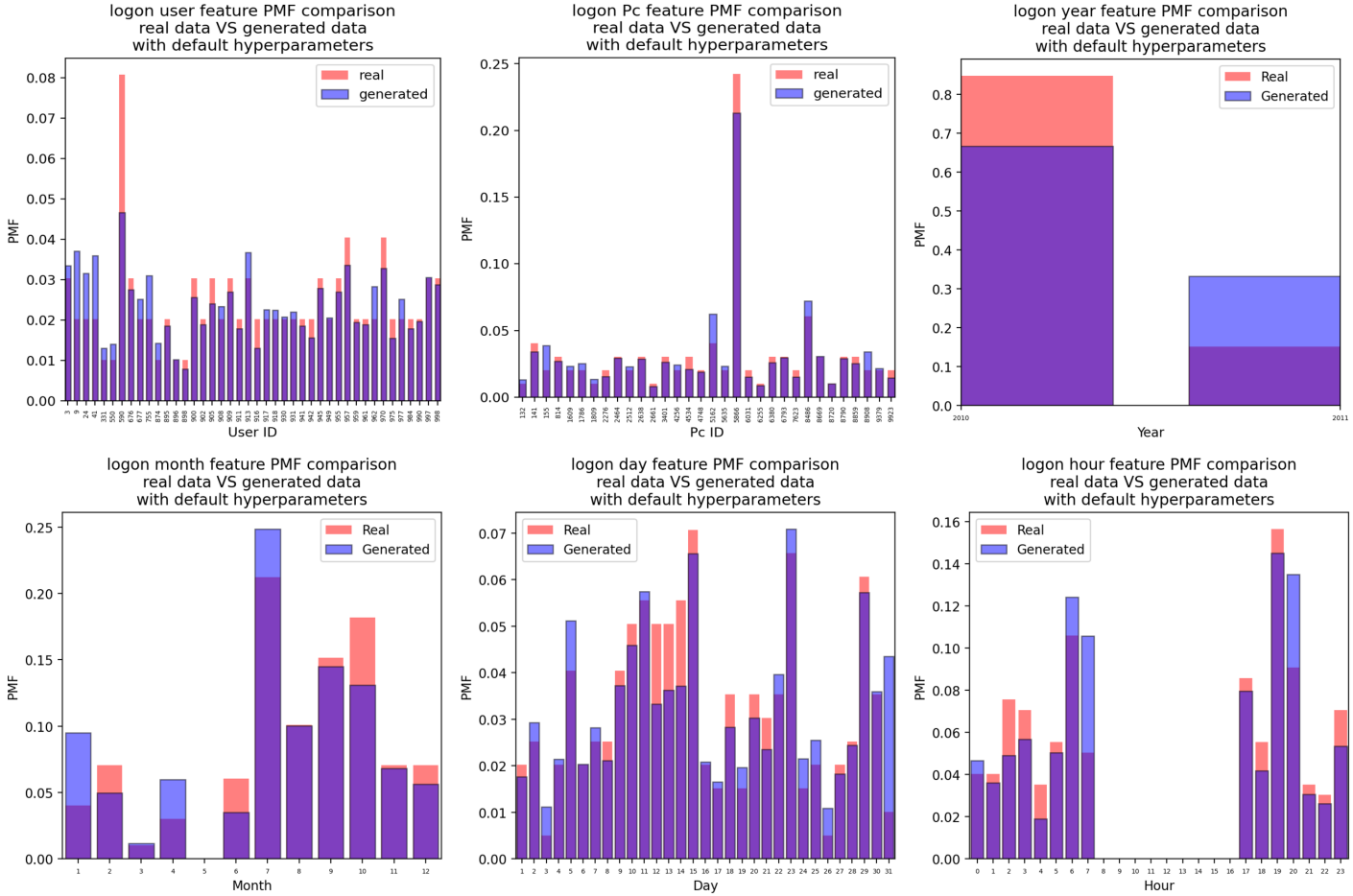


Figure A1. The PMFs of the actual logon attacks and those of the logon attacks generated by the CTGAN with default hyperparameter values.

Appendix A.1.2. For the Case of the Attacks Generated Using the CTGAN with Optimized Hyperparameter Values

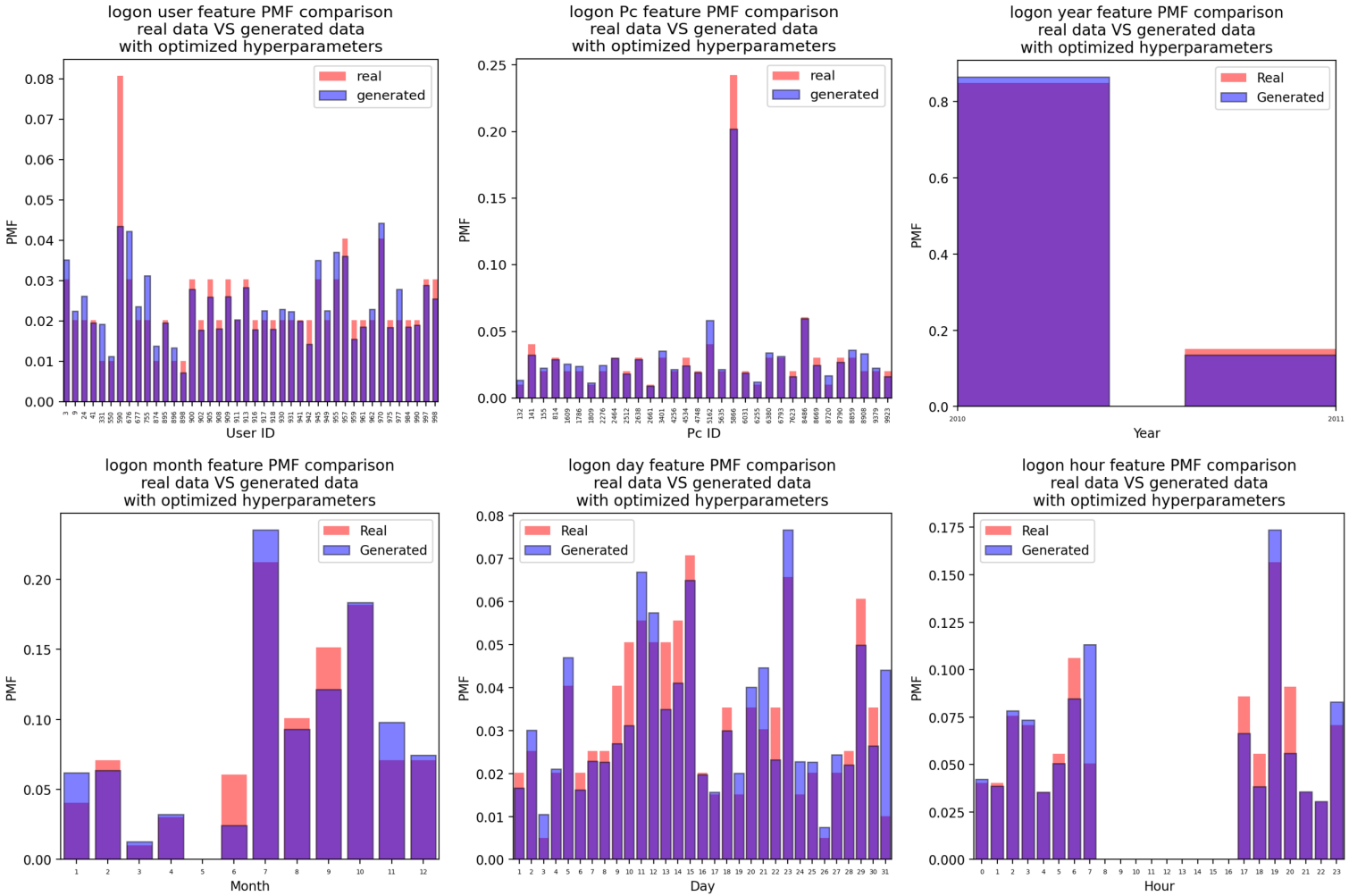


Figure A2. The PMFs of the actual logon attacks and those of the logon attacks generated by the CTGAN with optimal hyperparameter values.

Appendix A.2. Device Attacks

Appendix A.2.1. For the Case of the Attacks Generated Using the CTGAN with Default Hyperparameter Values

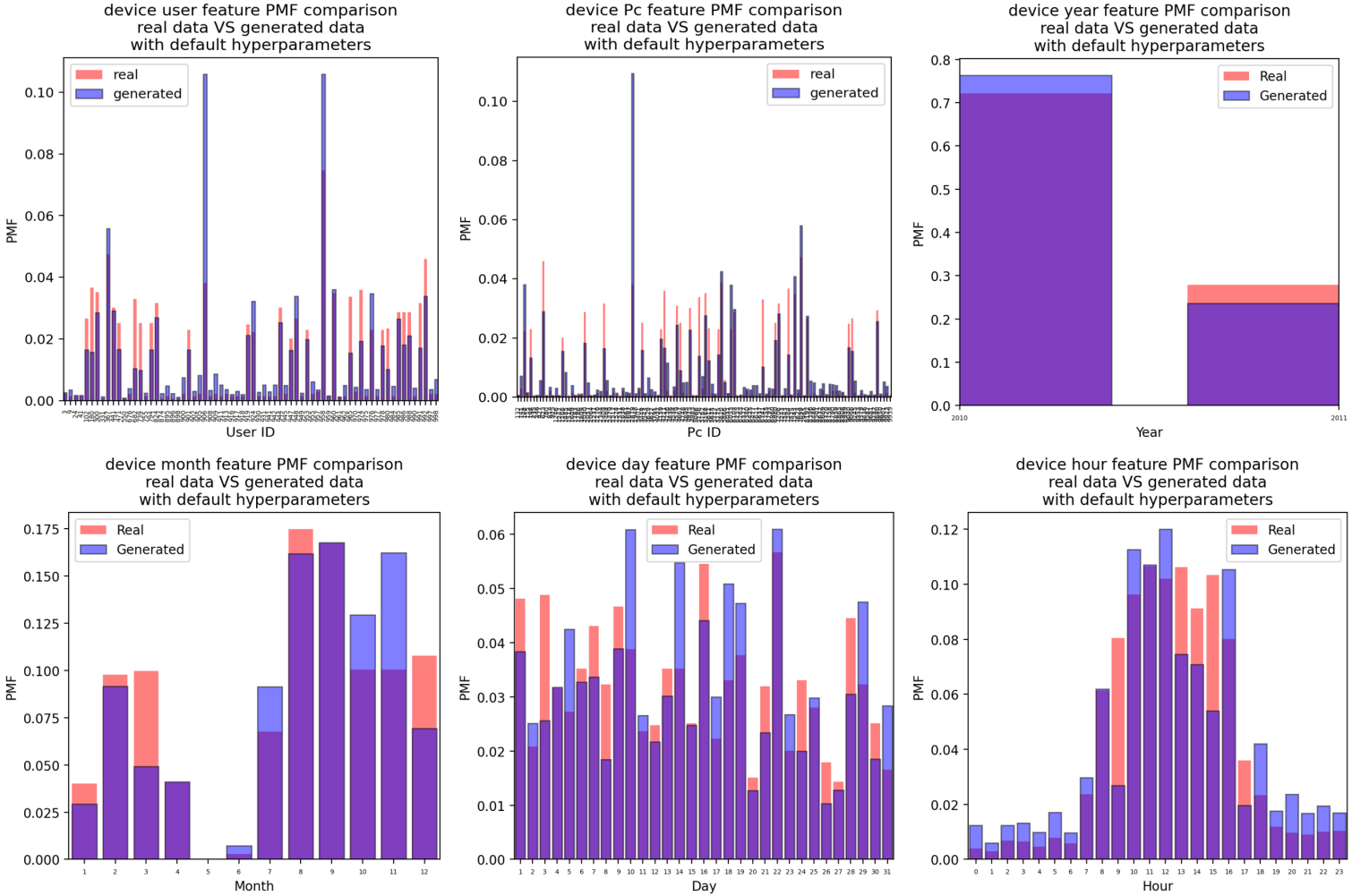


Figure A3. The PMFs of the actual device attacks and those of the device attacks generated by the CTGAN with default hyperparameter values.

Appendix A.2.2. For the Case of the Attacks Generated Using the CTGAN with Optimized Hyperparameter Values

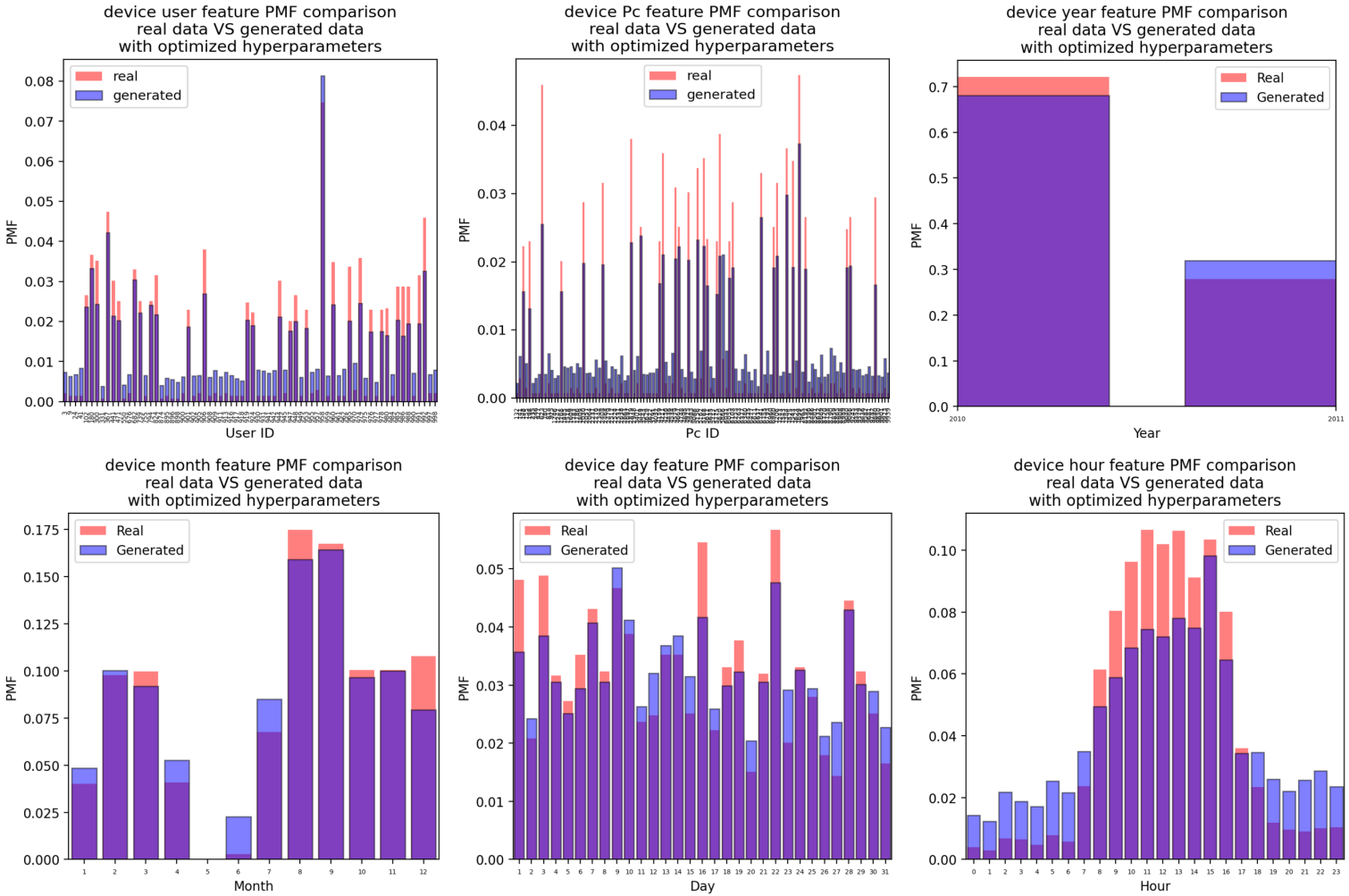


Figure A4. The PMFs of the actual device attacks and those of the device attacks generated by the CTGAN with optimal hyperparameter values.

Appendix A.3. Email Attacks

Appendix A.3.1. For the Case of the Attacks Generated Using the CTGAN with Default Hyperparameter Values

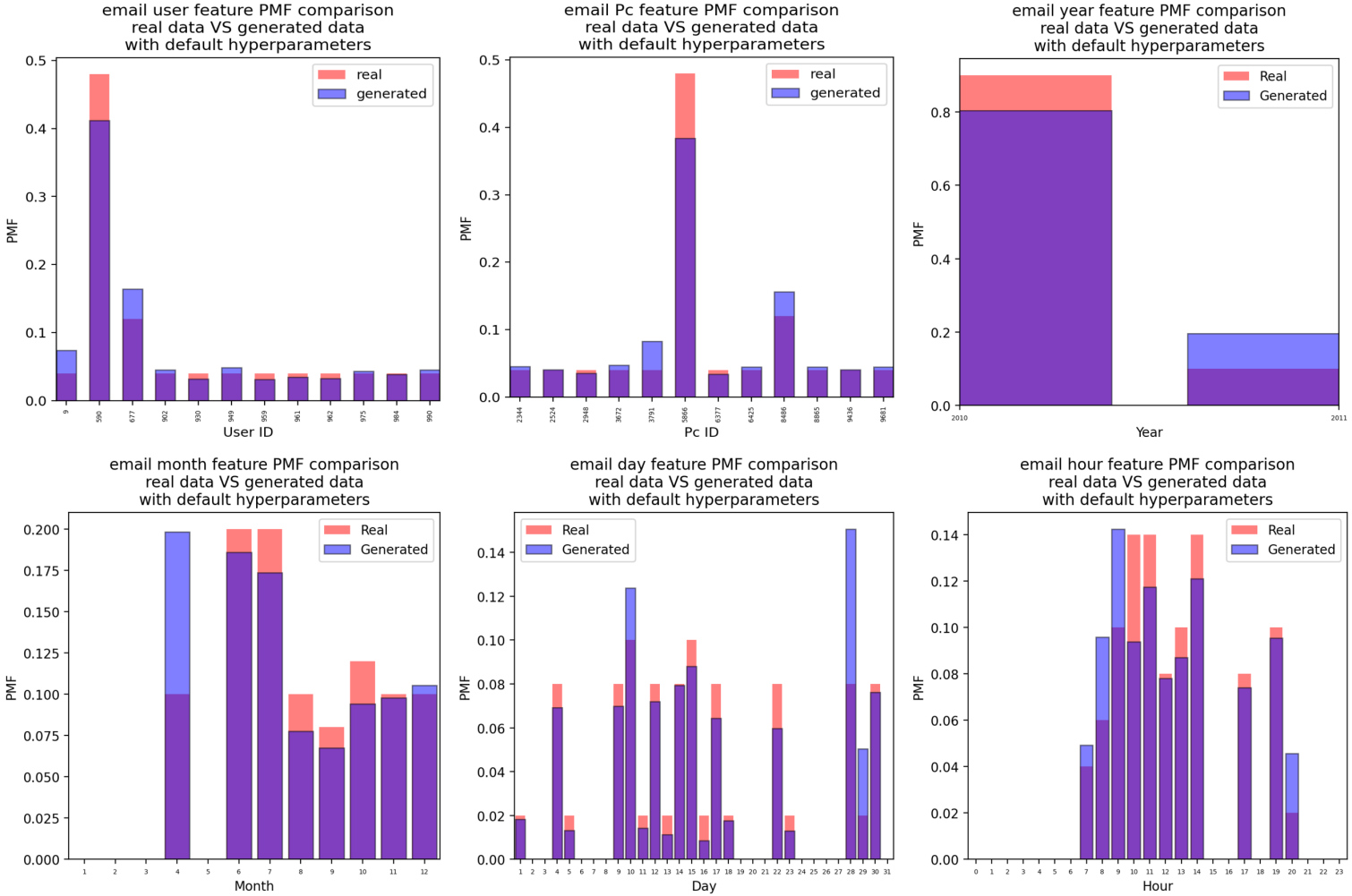


Figure A5. The PMFs of the actual email attacks and those of the email attacks generated by the CTGAN with default hyperparameter values.

Appendix A.3.2. For the Case of the Attacks Generated Using the CTGAN with Optimized Hyperparameter Values

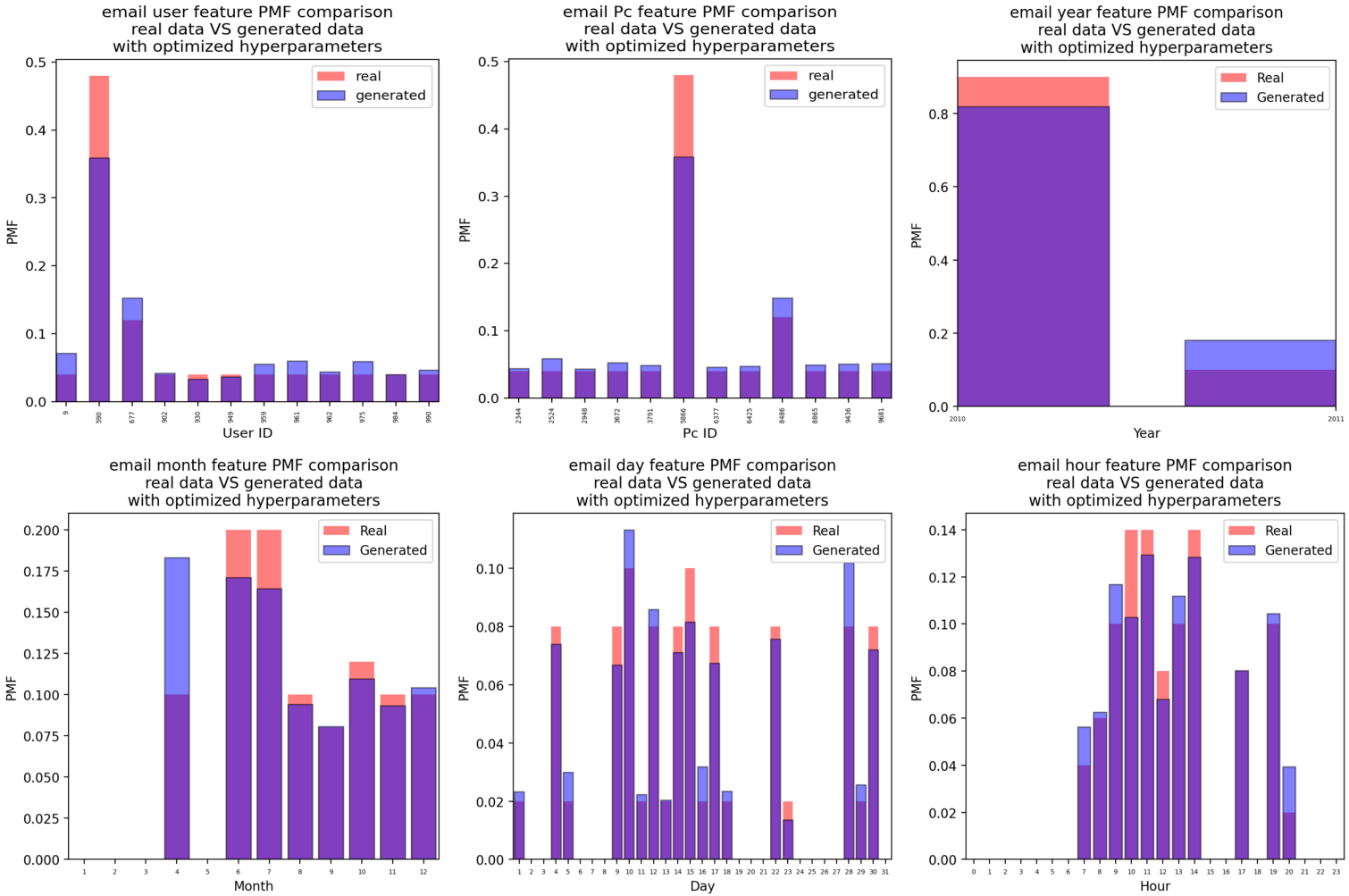


Figure A6. The PMFs of the actual email attacks and those of the email attacks generated by the CTGAN with optimal hyperparameter values.

Appendix A.4. HTTP Attacks

Appendix A.4.1. For the Case of the Attacks Generated Using the CTGAN with Default Hyperparameter Values

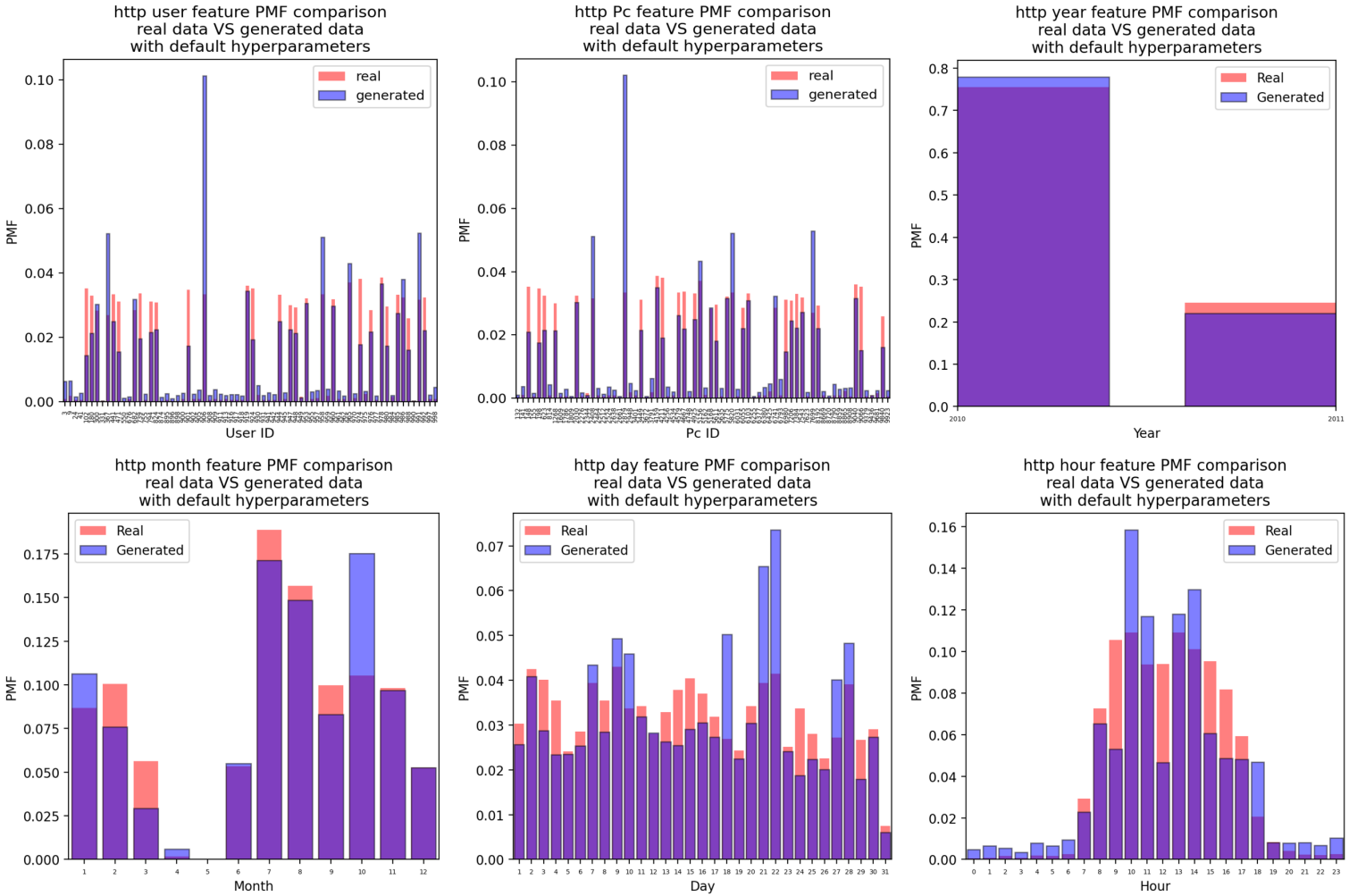


Figure A7. The PMFs of the actual http attacks and those of the http attacks generated by the CTGAN with default hyperparameter values.

Appendix A.4.2. For the Case of the Attacks Generated Using the CTGAN with Optimized Hyperparameter Values

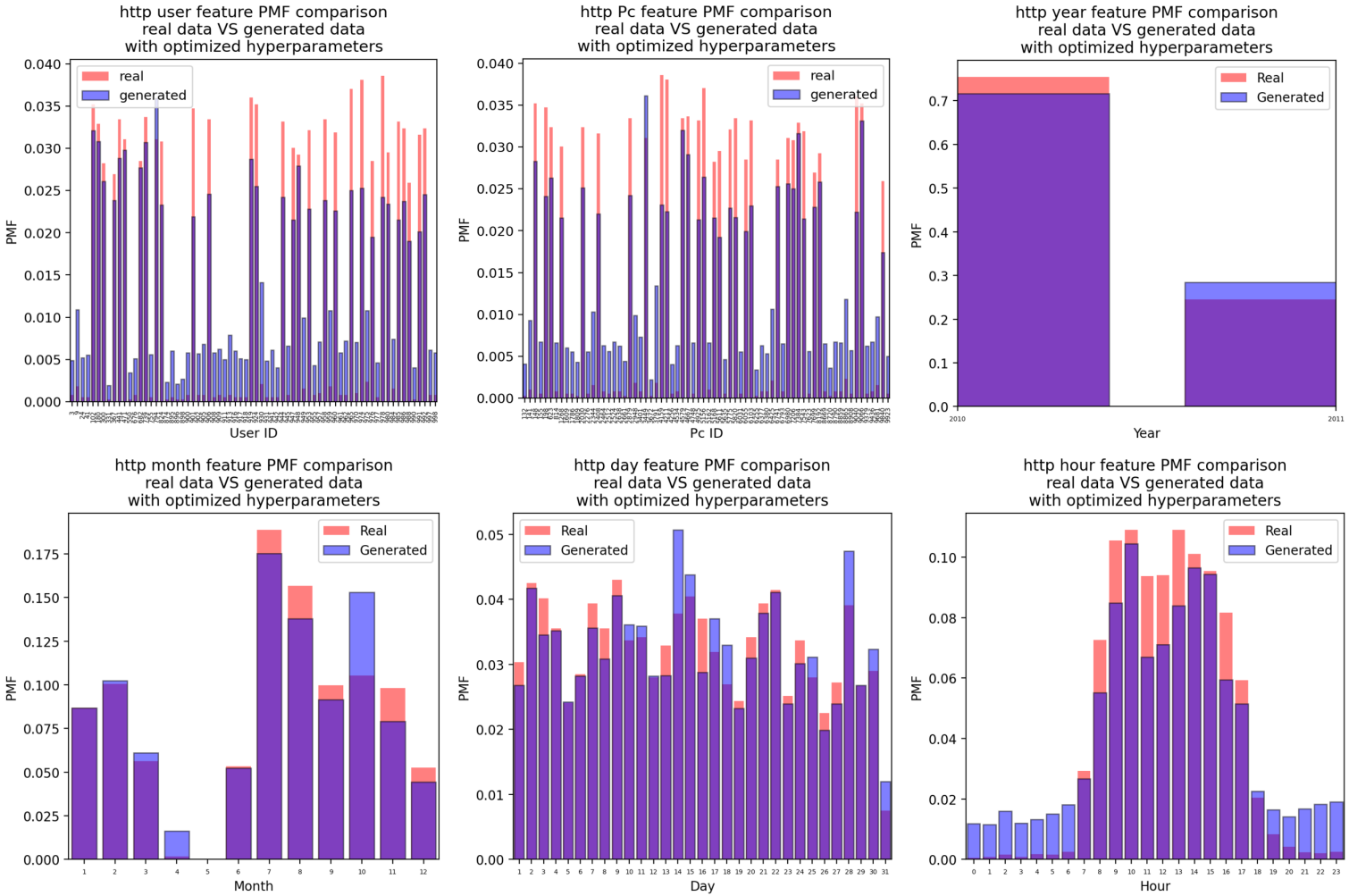


Figure A8. The PMFs of the actual http attacks and those of the http attacks generated by the CTGAN with optimal hyperparameter values.

Appendix A.5. File Attacks

Appendix A.5.1. For the Case of the Attacks Generated Using the CTGAN with Default Hyperparameter Values

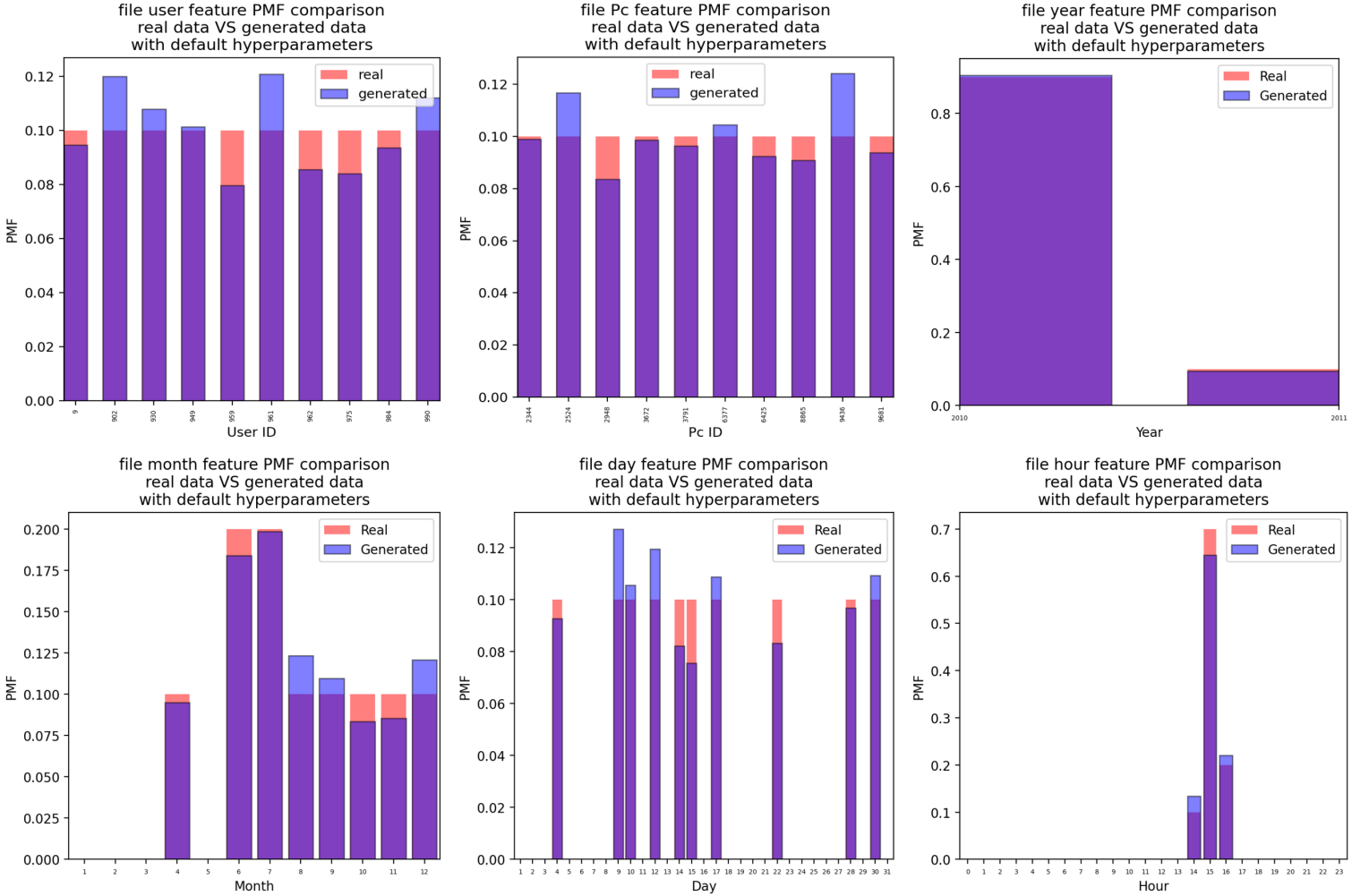


Figure A9. The PMFs of the actual file attacks and those of the file attacks generated by the CTGAN with default hyperparameter values.

Appendix A.5.2. For the Case of the Attacks Generated Using the CTGAN with Optimized Hyperparameter Values

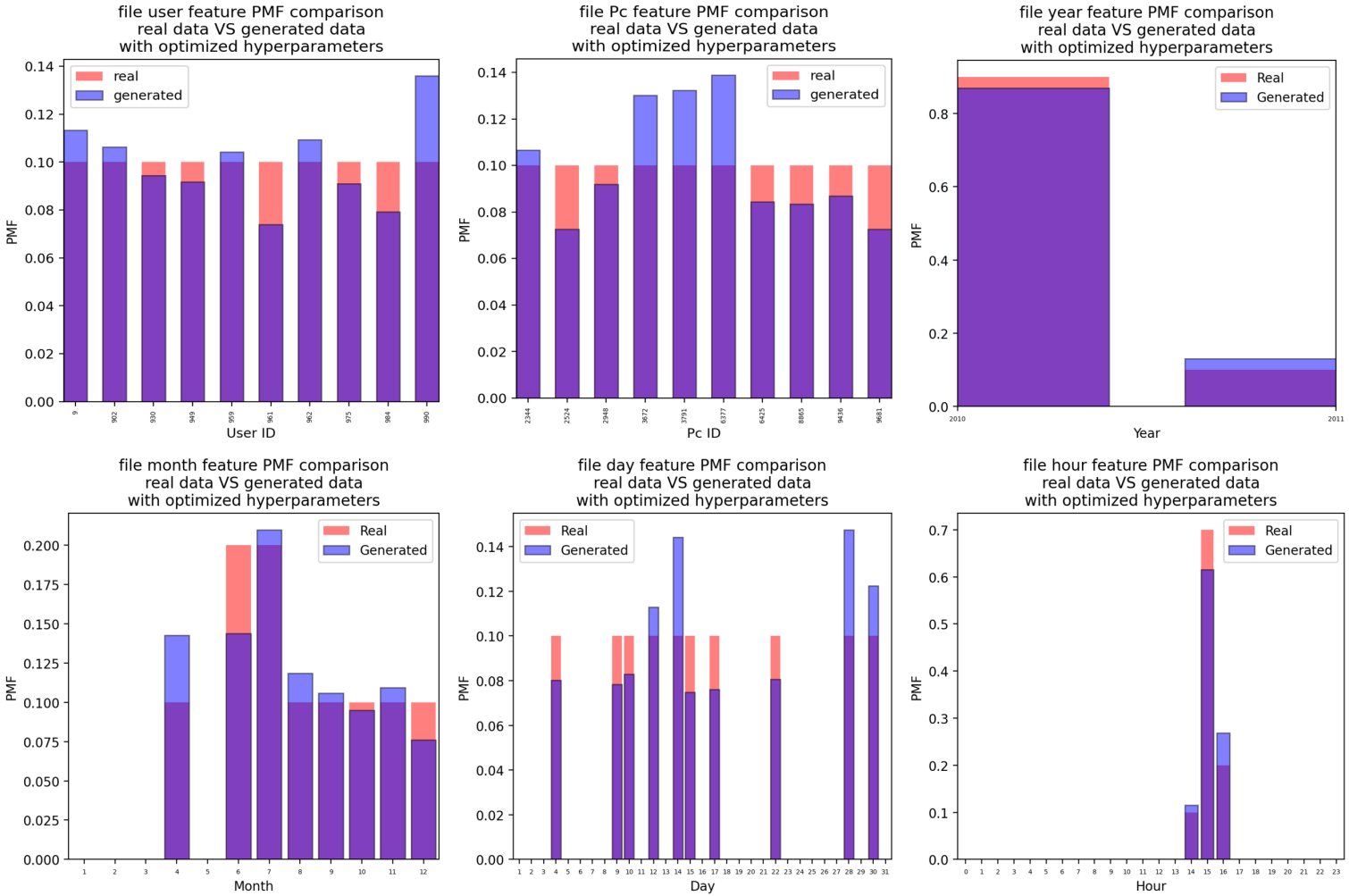


Figure A10. The PMFs of the actual file attacks and those of the file attacks generated by the CTGAN with optimal hyperparameter values.

Appendix B. The Mean Absolute Error (MAE) (between the Probability Mass Functions (PMFs) of the Original Attacks and of the Attacks Generated Using the CTGAN with Optimized Hyperparameter Values) as Function of the Number of Attacks Sampled from the PMFs Estimated Using the CTGAN

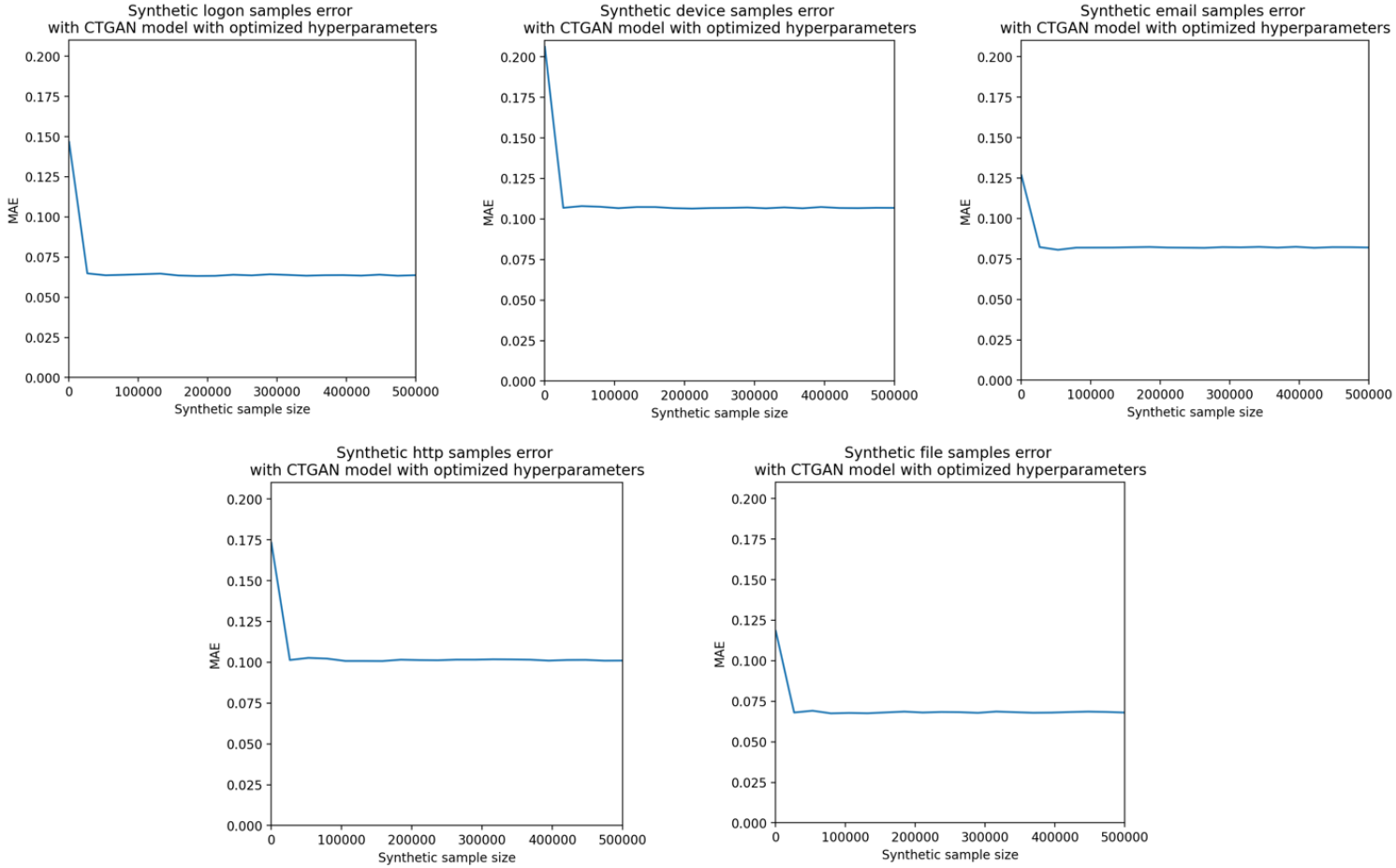


Figure A11. The mean absolute error (MAE) (between the probability mass functions (PMFs) of the original attacks and of the attacks generated using the CTGAN with optimized hyperparameter values) as function of the number of attacks sampled from the PMFs estimated using the CTGAN.

References

1. Azaria, A.; Richardson, A.; Kraus, S.; Subrahmanian, V. Behavioral analysis of insider threat: A survey and bootstrapped prediction in imbalanced data. *IEEE Trans. Comput. Soc. Syst.* **2014**, *1*, 135–155. [CrossRef]
2. Trzeciak, R.; CERT INSIDER THREAT CENTER. The CERT Insider Threat Database. Carnegie Mellon University, Software Engineering Institute's Insights (Blog). 2011. Available online: https://insights.sei.cmu.edu/blog/the-cert-insider-threat-database/ (accessed on 24 July 2023).
3. Glasser, J.; Lindauer, B. Bridging the gap: A pragmatic approach to generating insider threat data. In Proceedings of the Security and Privacy Workshops, San Francisco, CA, USA, 23–24 May 2013; IEEE: New York, NY, USA, 2013; pp. 98–104.
4. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; IEEE: New York, NY, USA, 2009; pp. 1–6.
5. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP), Funchal, Portugal, 22–24 January 2018; SciTePress: Setubal, Portugal, 2018.
6. Xu, L.; Skoularidou, M.; Cuesta-Infante, A.; Veeramachaneni, K. Modeling tabular data using conditional GAN. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 7335–7345.
7. Caminero, G.; Lopez-Martin, M.; Carro, B. Adversarial environment reinforcement learning algorithm for intrusion detection. *Comput. Netw.* **2019**, *159*, 96–109. [CrossRef]
8. Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double Q-learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; AAAI Press: Washington, DC, USA, 2016; Volume 30.
9. Huber, P.J. Robust estimation of a location parameter. In *Breakthroughs in Statistics: Methodology and Distribution*; Springer: New York, NY, USA, 1992; pp. 492–518.
10. Brown, P.; Brown, A.; Gupta, M.; Abdelsalam, M. Online malware classification with system-wide system calls in cloud IaaS. In Proceedings of the 23rd International Conference on Information Reuse and Integration for Data Science (IRI), Virtual Conference, 9–11 August 2022; pp. 146–151.
11. Bergstra, J.; Bardenet, R.; Bengio, Y.; Kégl, B. Algorithms for hyper-parameter optimization. *Adv. Neural Inf. Process. Syst.* **2011**, *24*, 2546–2554.
12. McCarthy, A.; Ghadafi, E.; Andriotis, P.; Legg, P. Functionality-preserving adversarial machine learning for robust classification in cybersecurity and intrusion detection domains: A survey. *J. Cybersecur. Priv.* **2022**, *2*, 154–190. [CrossRef]
13. Gayathri, R.G.; Sajjanhar, A.; Xiang, Y.; Ma, X. Anomaly detection for scenario-based insider activities using CGAN augmented data. In Proceedings of the 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Shenyang, China, 20–22 October 2021; IEEE: New York, NY, USA, 2021; pp. 718–725. [CrossRef]
14. Yuan, F.; Shang, Y.; Liu, Y.; Cao, Y.; Tan, J. Data augmentation for insider threat detection with GAN. In Proceedings of the 32nd International Conference on Tools with Artificial Intelligence (ICTAI), Baltimore, MD, USA, 9–11 November 2020; IEEE: New York, NY, USA, 2020; pp. 632–638. [CrossRef]
15. Gayathri, R.G.; Sajjanhar, A.; Xiang, Y. Adversarial training for robust insider threat detection. In Proceedings of the International Joint Conference on Neural Networks (IJCNN), Padua, Italy, 18–23 July 2022; IEEE: New York, NY, USA, 2022; pp. 1–8. [CrossRef]
16. Ali-Gombe, A.; Elyan, E. MFC-GAN: Class-imbalanced dataset classification using multiple fake class generative adversarial network. *Neurocomputing* **2019**, *361*, 212–221. [CrossRef]
17. Sood, T.; Prakash, S.; Sharma, S.; Singh, A.; Choubey, H. Intrusion detection system in wireless sensor network using conditional generative adversarial network. *Wirel. Pers. Commun.* **2022**, *126*, 911–931. [CrossRef]
18. Chen, T.; Guestrin, C. XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; ACM: New York, NY, USA, 2016; pp. 785–794.
19. OpenAI. ChatGPT: Large Language Model. 2023. Available online: https://chat.openai.com/ (accessed on 20 July 2023).
20. Gupta, M.; Akiri, C.; Aryal, K.; Parker, E.; Praharaj, L. From ChatGPT to ThreatGPT: Impact of generative AI in cybersecurity and privacy. *IEEE Access* **2023**. [CrossRef]
21. Aryal, K.; Gupta, M.; Abdelsalam, M. A survey on adversarial attacks for malware analysis. *IEEE Commun. Surv. Tutor.* **2022**, *25*, 467–496.
22. Elderman, R.; Pater, L.J.J.; Thie, A.S.; Drugan, M.M.; Wiering, M.A. Adversarial reinforcement learning in a cyber security simulation. In Proceedings of the 9th International Conference on Agents and Artificial Intelligence (ICAART), Porto, Portugal, 24–16 February 2017; SciTePress Digital Library: Setubal, Portugal, 2017; pp. 559–566.
23. Sethi, K.; Madhav, Y.V.; Kumar, R.; Bera, P. Attention based multi-agent intrusion detection systems using reinforcement learning. *J. Inf. Secur. Appl.* **2021**, *61*, 102923. [CrossRef]
24. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
25. Jin, X.; Liang, J.; Tong, W.; Lu, L.; Li, Z. Multi-agent trust-based intrusion detection scheme for wireless sensor networks. *Comput. Electr. Eng.* **2017**, *59*, 262–273. [CrossRef]

26. Krishnan Sadhasivan, D.; Balasubramanian, K. A fusion of multiagent functionalities for effective intrusion detection system. *Secur. Commun. Netw.* **2017**, *2017*, 6216078. [CrossRef]
27. Freund, Y.; Schapire, R.E. A desicion-theoretic generalization of on-line learning and an application to boosting. In Proceedings of the European Conference on Computational Learning Theory, Barcelona, Spain, 13–15 March 1995; Springer: Berlin/Heidelberg, Germany, 1995; pp. 23–37.
28. Cohen, W.W. Fast effective rule induction. In *Machine Learning Proceedings*; Elsevier: Amsterdam, The Netherlands, 1995; pp. 115–123.
29. Stolfo, S.; Fan, W.; Lee, W.; Prodromidis, A.; Chan, P. *KDD Cup 1999 Data*; UCI Machine Learning Repository; University of California: Irvine, CA, USA, 1999. [CrossRef]
30. Boyer, S.A. *Supervisory Control and Data Acquisition*; International Society of Automation (ISA): Pittsburgh, PA, USA, 1999.
31. Achbarou, O.; El Kiram, M.A.; Bourkoukou, O.; Elbouanani, S. A new distributed intrusion detection system based on multi-agent system for cloud environment. *Int. J. Commun. Netw. Inf. Secur.* **2018**, *10*, 526. [CrossRef]
32. Suwannalai, E.; Polprasert, C. Network intrusion detection systems using adversarial reinforcement learning with deep Q-network. In Proceedings of the 18th International Conference on ICT and Knowledge Engineering (ICT&KE), Bangkok, Thailand, 18–20 November 2020; IEEE: New York, NY, USA, 2020; pp. 1–7.
33. Ma, X.; Shi, W. AESMOTE: Adversarial reinforcement learning with SMOTE for anomaly detection. *IEEE Trans. Netw. Sci. Eng.* **2020**, *8*, 943–956. [CrossRef]
34. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
35. Bishop, C.M.; Nasrabadi, N.M. *Pattern Recognition and Machine Learning*; Springer: New York, NY, USA, 2006; Volume 4.
36. Jang, E.; Gu, S.; Poole, B. Categorical reparameterization with Gumbel-Softmax. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France , 24–26 April 2017.
37. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A.C. Improved training of Wasserstein GANs. In Proceedings of the 31st Advances in Neural Information Processing Systems , Red Hook, NY, USA, 4–9 December 2017; Volume 30.
38. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, USA , 7–9 May 2015.