A Tale of Two Methods: Unveiling the limitations of GAN and the Rise of Bayesian Networks for Synthetic Network Traffic Generation

Adrien SCHOEN
Inria, Univ. Rennes, IRISA
Rennes, France
adrien.schoen@inria.fr

Gregory BLANC
Samovar, Télécom SudParis, IP Paris
Palaiseau, France
gregory.blanc@telecom-sudparis.eu

Pierre-François GIMENEZ
CentraleSupélec, Univ. Rennes, IRISA
Rennes, France
pierre-francois.gimenez@centralesupelec.fr

Yufei HAN
Inria, Univ. Rennes, IRISA
Rennes, France
yufei.han@inria.fr

Frédéric MAJORCZYK
DGA-MI, Univ. Rennes, IRISA
Rennes, France
frederic.majorczyk@intradef.gouv.fr

Ludovic ME
Inria, Univ. Rennes, IRISA
Rennes, France
Ludovic.Me@inria.fr

2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW) | 979-8-3503-6729-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/EUROSPW61312.2024.00036

***Abstract*—The evaluation of network intrusion detection systems requires a sufficient amount of mixed network traffic, i.e., composed of both malicious and legitimate flows. In particular, obtaining realistic legitimate traffic is hard. Synthetic network traffic is one of the tools to respond to insufficient or incomplete real-world datasets. In this paper, we only focus on synthetically generating high-quality legitimate traffic and we do not delve into malicious traffic generation. For this specific task, recent contributions make use of advanced machine learning-driven approaches, notably through Generative Adversarial Networks (GANs). However, evaluations of GAN-generated data often disregards pivotal attributes, such as protocol adherence. Our study addresses the gap by proposing a comprehensive set of metrics that assess the quality of synthetic legitimate network traffic. To illustrate the value of these metrics, we empirically compare advanced network-oriented GANs with a simple and yet effective probabilistic generative model, Bayesian Networks (BN). According to our proposed evaluation metrics, BN-based network traffic generation outperforms the state-of-the-art GAN-based opponents. In our study, BN yields substantially more realistic and useful synthetic benign traffic and minimizes the computational costs simultaneously.**

1. Introduction

High-quality, fully labeled, and recent network traffic datasets are essential for evaluating network security methods. However, constructing such datasets typically presents three main challenges. Firstly, capturing real network traffic may raise concerns regarding data privacy protocols. Secondly, labeling real network traffic is a time-intensive process that requires significant manual investigation efforts. Additionally, network traffic data can be generated and labeled using well-controlled and configured network testbeds [1], [2]. Labeling network traffic within simulation testbeds is straightforward and does not raise privacy concerns for data sharing. However, simulating network activity can be complex and costly, particularly when simulating many users' network activities. Conversely, synthesizing network traffic [3] avoids the limitations of simulation and emerges as a promising solution to address the scarcity of high-quality network traffic datasets.

There has been a growing interest in applying Machine Learning (ML) techniques to generate synthetic benign traffic in the past few years. Data generation methods – widely successful in creating diverse types of data such as images [4], [5], textual data [6], and tabular data [7], [8] – have inspired researchers to adapt these techniques to generate synthetic network traffic data. Notably, **G**enerative **A**dversarial **N**etworks (**GAN**s) [4] have been applied in numerous studies to create network data [9]–[19] but other approaches that use auto-regression models [20] have been used too. Among those studies, some opt to create synthetic network packets [10], [12], [21], while others chose to generate network flows [11], [14], [19], [22] or network features [15], [16], [18].

In our study, we focus on generating only benign traffic. Attack traffic can often be captured in testbeds with pen-testing or attack tools and then mixed with a dataset of benign traffic [23]. In contrast, benign traffic is more difficult to replicate due to the great diversity of benign network usage. We also chose to generate only **N**etwork **F**lows (**NF**s) because it has been widely used as an input data structure of NIDS [24]. NFs are composed of mixed types of features, including both categorical and numerical, characterizing network traffic. For example, *Transport Protocol* is a categorical feature, whereas *the Duration of the flows* is a numerical one (see Section 2.1 for more details). Network communications follow predefined protocols and programmed activities. As a result, different features in NFs are usually strongly correlated, indicating clear patterns that differentiate network activities. For instance, the Destination port can have a strong influence on the Transport Protocol (NFs targeting port 80 have a high chance of using the TCP transport protocol).

How to encode these inter-feature correlations is hence the key to generate realistic benign NFs. To reach this goal, inspired by Goncalves et al. [25], we propose to use **B**ayesian **N**etworks (**BN**s) [7] as a novel network flows synthesizing solution. BNs are by design a probabilistic generative model, explicitly modelizing the conditional

DOI 10.1109/EuroSPW61312.2024.00036

dependency between various attributes, which in turn achieves a computationally economic estimation to the underlying joint distribution. According to Goncalves et al., BNs have been used successfully in generating patients' medical records.

Compared to the end-to-end data generation methods, e.g. GANs [4], BNs have two merits in synthesizing NFs. First of all, BNs directly capture the correlation between NF features from training data by estimating the inter-feature conditional probability. It provides a lens to investigate and clone the statistical profile of NFs. Second, BNs have significantly lower computational overheads than GAN-based methods. Besides, GAN-based methods suffer from their black-box and non-linear nature of data generation. As reported in [26], GANs are prone to *mode collapse*, where only a subset of the real distribution is generated, due to the non-convex adversarial learning objective. It is, hence, difficult to control the quality of the generated NFs [11].

Our work demonstrates the use of BNs for synthesizing NFs. We organize an empirical comparison of the proposed BN-based and state-of-the-art GAN-based network flow generation methods. To reach a comprehensive comparative study, we propose to build a set of evaluation metrics that cover multiple aspects of synthetic data quality, like its Realism, Diversity, and Novelty, but also its Compliance with network protocols. We employ the measurement metrics to demonstrate the superior quality of the BN-based NFs generation solution over those of the state-of-the-art GAN-based methods, yet only requiring a small fraction of computational costs.

The paper is structured as follows. Section 2 introduces some preliminary concepts. Section 3 presents previous works on generating NFs and evaluating that generation. In Section 4, we unveil a comprehensive set of scoring functions which, to the best of our knowledge, constitutes the most exhaustive and comprehensive benchmark for evaluating the quality of NF generation. Section 5 introduces a novel generation method based on BNs. Finally, in Section 6, we demonstrate the superiority of our model compared to various GAN-based methods.

2. Background

2.1. Network flows (NFs)

Network traffic comprises all the packets exchanged between hosts communicating within a given network during a given period of time. Subsets of packets sharing the same values for 5 common features are grouped into a *network flow*. Those common features are: *source IP address*, *destination IP address*, *source port*, *destination port*, and *transport protocol*. Besides this 5-tuple, a NF can also contain some statistical information about the communication between the two hosts, like, for example, its Duration or the Number of Bytes exchanged.

In this paper, NF generation means the generation of the 5-tuple as well as the statistical information. You can find the information describing each NF in Appendix E. As noted by [27] and [28], network traffic in the format of NFs is, therefore, similar to tabular data, as each line corresponds to a specific NF and each column to a specific network feature.

2.2. Generative Adversarial Networks (GANs)

GAN is a class of deep learning models [4] composed of two neural network modules: a generator and a discriminator, trained adversarially. They are adept at generating data resembling the training dataset. In computer vision, GANs successfully approximate complex distributions of real-world image data. However, their application to generating categorical tabular data is less explored. The discrete nature of categorical data poses an NP-hard challenge in replicating joint distributions [29], [30], compounded by strong statistical correlations among categorical features in tabular data. Moreover, as a black box model, ensuring accurate encoding of inter-feature correlation by the generator module in GANs is challenging [31], [32].

2.3. Bayesian Networks (BNs)

BNs are statistical models that represent the probabilistic relationships among variables [33]. A general introduction to BNs can be found in Appendix A. In contrast to GANs, BNs excel in capturing relational structures inherent in tabular data, explicitly modeling probabilistic dependencies between features. Previous studies [25], [30] have highlighted this advantage. Goncalves et al. [25] extensively compared GANs and statistical methods for generating medical patient data, finding BNs superior across diverse datasets, with lower computational overheads compared to GANs. Additionally, BNs mitigate the mode collapse issue inherent in GANs, ensuring stability during training. In our study, we propose using BNs for synthesizing network flow data.

BN training involves constructing **C**onditional **P**robability **T**ables (**CPT**s) for features, requiring discretization of numerical features like *flow duration* in network traffic data. However, BN scalability is limited by model size, with CPT size scaling exponentially with feature cardinality. High-cardinality network flow features like IP addresses or ports pose challenges in model scalability, as noted by [20].

3. Related works

In this section, we first introduce the previous work on GAN-based generation of synthetic benign NF data and how the quality of generated NFs is evaluated.

3.1. GAN-based generation of network traffic

Since the groundbreaking work of Ring et al. [22], the application of GANs to NF generation has garnered significant attention. Ring et al. proposed addressing discrete distributions and feature-wise dependencies by introducing a specific embedding called IP2Vec. Recognizing the similarities between NF and tabular data (see subsection 2.1), Anande et al. [27] and Bourrou et al. [28] both suggested GAN-based approaches tailored to preserving feature-wise dependencies during generation. More recently, Lin et al. [18] utilized GANs to generate time series corresponding to NFs. Their work laid the foundation for NetShare [19], which improved upon their methods and

extended them to learn not only dependencies between individual flow features but also temporal dependencies among entire datasets.

Beyond the realm of GANs, it is worth mentioning STAN [20], which employs an autoregressive neural network to predict one feature's value based on others, thereby capturing the distribution of NF traffic. This underscores the importance of capturing statistical correlations between network traffic attributes when generating informative network traffic data. However, certain challenges arise when using GANs for this purpose. Firstly, due to the multi-modal distributions of many numerical NF features, GANs are susceptible to mode collapse, wherein the generated data only covers a portion of the training data distribution. Secondly, certain categorical features of NFs, such as *Source Port*, span a large range of categories, leading to high-dimensional and sparse data. Given that GANs rely on a discriminator to distinguish between generated and real examples, this sparsity can lead to discriminators prioritizing sparseness over realism, resulting in suboptimal generation. Furthermore, NFs contain highly correlated features, such as *Protocol* and *Destination Port*, due to network protocol regulations, hardware architectures, and user behavioral patterns. GANs struggle to explicitly capture these intrinsic correlations between different network attributes.

Despite efforts to adapt GANs for NF generation, anomalies persist in the generated data. For example, in Table 2 of [11], multiple NFs exhibit ephemeral ports for both the source and destination and one NF displays multiple packets with null duration, which should only occur when a single packet is transmitted. Similar anomalies can also be found in the results of the most recent GAN-based method, NetShare [19], as demonstrated in Appendix F.

3.2. Quality evaluation of data generation

The evaluation of synthetic tabular data quality spans various domains [6], [34], [35]. Tabular data generation entails synthesizing observations of n features $(X^i_{gen})_{i\leq n}$ to align with the joint distribution $(X^i_{source})_{i\leq n}$. Evaluation typically involves three approaches: comparing marginal distributions, assessing inter-variable correlations, and examining joint distributions. Marginal distribution comparison utilizes metrics like Jensen-Shannon Divergence (JSD), Earth Mover's Distance (EMD), or the Kolmogorov–Smirnov test [25], [30], [36]. Assessing inter-variable correlations ensures that real dataset correlations are captured in the generated dataset [16], [18], [31], [37], often through Pairwise Correlation Matrices or Chi-squared Tests [25], [36]. Comparing joint distributions involves examining sample distributions, typically using precision and recall metrics or utility assessment in subsequent machine learning tasks [34], [38]–[42]. In the latter, the generated data is used to train machine learning algorithms, with performance compared to using training data. A line of research efforts [25], [34], [39], [43] encompasses three major criteria to define a good tabular data generation approach:

- **Realism**: a synthetic sample should be sampled from the same distribution as the real data.
- **Diversity**: the distribution of the synthetic samples should have the same or close variance level as that of the real data.
- **Novelty** (named as authenticity in [43]): a generated sample should be sufficiently different from the samples of the real distribution. To avoid any confusion with the classical definition of authenticity in security, we prefer the term "**novelty**".

Given the resemblance between tabular and NF data, the evaluation methodology for NF generation closely mirrors that of tabular data. This entails utilizing metrics such as JSD [20], EMD [11], [18], [19], [28], or assessing downstream classification model performance [18]–[20], [28], [38]. We have summarized these metrics and their corresponding evaluation criteria in Table 1. Additionally, some studies [20], [22] propose a *Domain Knowledge Check* (*DKC*), where generated NF are evaluated for adherence to fundamental network protocol rules. Consequently, we introduce the "Compliance" criterion to gauge how well a generated NF conforms to protocol specifications, distinct from "Realism", which measures how closely a sample aligns with real data distributions. For instance, a NF consisting of a single packet with a duration close to zero might seem realistic, yet it does not meet Compliance standards as it diverges from protocol specifications (wherein a single-packet NF should indeed have a duration of zero).

As depicted in Table 1, the evaluation of conditional distributions, which ensures the preservation of correlations between attributes in synthetic data, is often overlooked. Only a few studies [18], [28] emphasize the importance of maintaining correlations within generated network traffic flows through dedicated evaluations. Furthermore, a notable gap in current evaluations pertains to Novelty assessment. Specifically in NF generation, only DoppelGANger [18] addresses concerns about overfitting and the risk of duplicating training data. Despite its crucial role in assessing tabular data generation [25], [34], [39], [44], its significance remains underestimated for NFs.

4. Evaluation Metrics

In this section, we present a comprehensive set of evaluation metrics for assessing the quality of generated NF data. Our evaluation framework addresses three key criteria outlined in Section 3.2: Realism, Diversity, and Novelty of the generated NFs. Additionally, we introduce the Compliance measurement in Section 3.2, which evaluates the adherence of generated NFs to network specifications. This evaluation system provides detailed insights into the quality of the generated NF data. Its main features include:

- **Criterion-based assessment**: Our evaluation process incorporates benchmarks for four predefined criteria: Realism, Diversity, Novelty, and Compliance. Each of these criterion is assessed individually to provide a comprehensive evaluation.
- **Attribute-level and Sample-level Evaluation**: Given that NF data is tabular data (refer to Subsection 2.1), we evaluate Realism and Diversity at both the attribute level (examining marginal

TABLE 1: Summary of the functions used to evaluate generated network flows.

Evaluation method	Criteria				Descriptions	
	Realism	Diversity	Novelty	Compliance	Ref.	Input
Euclidian Distance	✓				[22]	Marg. Distr.
Domain Knowledge Check				✓	[20], [22]	Joint Distr.
Classifier performance loss	✓	✓			[18]–[20], [28], [38]	Joint Distr.
EMD	✓	✓			[11], [18], [28], [19]	Marg. Distr.
JSD	✓	✓			[20], [19]	Marg. Distr.
False Negative Test	✓				[28]	Joint Distr.
Correlation Matrix Comparison	✓				[18], [28]	Cond. Distr.
Chi-Squared Test	✓				[28]	Cond. Distr.
Membership disclosure			✓		[18]	Joint Distr.

Marg. Distr.: Marginal Distribution, Cond. Distr.: Conditional Distribution, Joint Distr.: Joint Distribution

TABLE 2: Summary of the function used in our evaluation system.

	Criterion				**Input**			**Data type**	
	Real.	*Div.*	*Nov.*	*Comp.*	*Marg.*	*Cond.*	*Joint*	*Cat.*	*Num.*
JSD	✓	✓			✓			✓	
EMD	✓	✓			✓				✓
CMD	✓					✓		✓	
PCD	✓					✓			✓
Density	✓						✓	✓	✓
Coverage		✓					✓	✓	✓
MD			✓				✓	✓	✓
DKC				✓			✓	✓	✓
Global	✓	✓	✓	✓	✓	✓	✓	✓	✓

Real.: Realism, Div.: Diversity, Nov.: Novelty, Marg. Distr.: Marginal Distribution, Joint Distr.: Joint Distribution

distributions of individual attributes) and the sample level (analyzing joint distributions of all attributes). Additionally, we introduce an assessment of the similarity in inter-feature correlation between synthetic NF data and the underlying correlation patterns present in real data.

- **Data Type Specificity**: Our evaluation protocol includes metrics tailored to each data type to identify potential challenges in modeling specific data types (numerical/categorical).

Assessing the proximity of the joint distributions between generated NF data and real NF data provides insights into their distributional consistency. However, the direct comparison of joint distributions is statistically unstable due to the sparsity of generated data. To address this issue, we advocate for evaluating marginal and conditional distributions as well. When the generated and real data exhibit highly similar conditional and marginal distributions, they are likely to demonstrate close joint distributions.

Table 2 provides an overall view of evaluation functions used in our study. In this table, we can find the evaluation criteria (Realism, Diversity, Novelty, and Compliance), the distributional divergence (marginal, conditional, or joint distribution), and the type of data (categorical or numerical data) that these metrics assess.

4.1. Comparing marginal distributions

First, we assess that the generated data follow the same marginal distribution than the real data. Following the insights in [28], we propose an evaluation strategy that separately measures the distributional closeness of numerical and categorical features. This approach allows us to apply the quality measurement to numerical and categorical features.

We use the *Jensen Shannon Divergence* (*JSD*) for discrete attributes of network traffic (such as *Protocol*) and *Earth Mover's Distance* (*EMD*) for numerical attributes (like *Duration* or *Bytes*). These two metrics have been widely used to compute quantitative measurements of the distribution divergence [18]–[20]. Compared to the statistical tests, like *Kolmogorov-Smirnov Test* or *Chi-squared Test*, *JSD* and *EMD* are derived without posing any assumption over the feature distribution. Beyond that, the statistical tests can only provide a qualitative measurement of the distribution difference, i.e., similar or not similar distributions. In contrast, *JSD* and *EMD* offer a continuous estimation to the distribution closeness. Larger JSD and EMD scores indicate a more significant divergence between the distributions of the generated and real NF data. Therefore, *JSD* and *EMD* can reach more accurate comparisons in the quality of generated data using different methods. A formal definition of *JSD* and *CMD* is given in Appendix B

4.2. Comparing conditional probabilities

However, assessing a match in independent feature distributions is insufficient; we also need to assess that the generated NFs keep the feature-wise dependencies of the real NFs. For this problem, popular correlation metrics like Spearman or Pearson coefficients typically apply to ordered features. However, certain attributes, such as the *Protocol* attribute, belong to an unordered categorical feature. Therefore, we also need a metric for such features.

For numerical features, we can analyze how their ordering correlates using the Spearman correlation coefficient or explore the existence of linear relationships among them using the Pearson correlation coefficient. Since linear correlations exist between features in NFs

between the number of packets, the total size of packets, and the NF duration, we consider the Pearson correlation coefficient more relevant. While the Spearman correlation coefficient can encompass non-linear correlations, it only verifies if two features are monotonically correlated. So, if there is a linear relationship $X = 2Y$ in the actual data and a quadratic relationship $X' = Y'^2$ in the generated data, the Spearman correlation coefficient would be identical, and the generated data would be deemed high quality, even if it is not. This issue would not happen with the Pearson correlation coefficient.

Pairwise conditional distribution (PCD) [25], [28], [45] is the L^2 norm of the difference between two pairwise correlations matrices, as in Eq. 1:

$$PCD(\mathrm{S}, \mathrm{G}) = \|\mathrm{Corr}(S) - \mathrm{Corr}(G)\|_2 \quad (1)$$

where $S = (X^i_{\text{source}})_{i \in [\![1,k]\!]}$ is the set of numerical features of the source data, $G = (X^i_{\text{gen}})_{i \in [\![1,k]\!]}$ is the set of numerical features of the generated data, and Corr(.) is the correlation matrix with Pearson coefficients.

To evaluate the preservation of dependencies among unordered categorical features, we need to study the difference between the contingency matrix in both the real data and the generated data. Therefore, we propose *Contingency Matrix Differences* (CMD), which is the difference of the contingency matrices of a pair of features on the synthetic data or the real dataset:

$$CMD(S,G) = \sum_{(i,j) \in [\![k+1,n]\!]^2} (\sum_{v,u} |\mathrm{P}(X^i_{\text{source}} = u | X^j_{\text{source}} = v) - \mathrm{P}(X^i_{\text{gen}} = u | X^j_{\text{gen}} = v)|) \quad (2)$$

where $S = (\mathrm{X}^j_{source})_{j \in [\![k+1,n]\!]}$ is the set of categorical features of the real data, and $G = (\mathrm{X}^j_{gen})_{j \in [\![k+1,n]\!]}$ is the set of categorical features of the generated data.

4.3. Comparing joint distribution

The metrics based on *PCD* and *CMD* only consider first-order dependencies. However, in the context of NF data, the conditional dependency between features usually includes higher-order information, i.e., the value of one NF feature may depend on multiple other features. For example, the *Number of Bytes* feature depends on both the *Number of Packets* and the *Protocol* type. Therefore, it is crucial to assess the joint distribution to capture high-order dependencies.

There are two methods in the literature for evaluating the joint distribution: comparing distribution manifolds or using the difference in the performance of a machine learning (ML) algorithm on a classification task as a proxy for measuring the similarity between the two distributions [38], [42]. Using the performance of an ML model can be misleading when used as a proxy to measure the closeness of joint distributions. Indeed, the performance of the ML model heavily depends on the definition of the classification task and the concrete choice of the ML model architecture. This can introduce high variability in the final result. Therefore, we opt for the first method, which is to compare real and generated distribution manifolds. To achieve this, we decided to use the *Density*/*Coverage* solution, implemented by Naeem et al. [40], because it evaluates Realism and Diversity independently, thus enforcing the granularity requirement of our benchmark.

Realism is evaluated by a function called for each synthetic sample, which counts how many real-sample neighborhood spheres contain the synthetic sample. The neighborhood spheres are computed using k-nearest neighbors. A low score indicates that real and synthetic data are far from each other, implying that the synthetic data is unrealistic.

Similarly, Diversity is evaluated by a function called *Coverage*. To compute *Coverage*, we count the number of synthetic neighborhood spheres that include this sample for each real sample. A low score suggests that several real samples lack a synthetic counterpart in their vicinity, indicating that the synthetic distribution does not adequately capture the variance of the real distribution.

Coverage and *Density* both rely on a specified number k of neighbors. Thankfully, Naeem et al. [40] led a hyperparameter optimization: according to their results, when we consider a generated dataset and a real dataset of both 10000 samples, the optimal k should be set to 5. We, therefore, use this value for k and this number of NFs in our comparative study.

4.4. Novelty evaluation

Inspired by Goncalves et al. [25], we decide to use the *Membership Disclosure* (*MD*) for measuring the Novelty criterion. This score has been previously employed for NFs [18], but was initially conceived for medical data. The objective of *MD* is to identify synthetic samples exhibiting characteristics that suggest they were copied from training instances.

To compute the *MD* score, we need a generated set, a training set, and a testing set (the last two sets being subsets of the real dataset). We begin by calculating the matrix of Hamming distances between every pair of generated and real samples. If a synthetic sample has a Hamming distance to a real sample below a certain threshold r, we flag the corresponding real sample as a leaked trained sample. Since we know which real samples are part of the training set or the testing set, for each r, we obtain a detector of training samples. Consequently, we can calculate the F1-score of such a detector and compute the integral of that F1-score depending on r. We do this calculation for all r. If the generated data includes instances copied from the training set, these copied samples would be detected even with a low threshold r, leading to an increase in the classifier's F_1 integral.

The interpretation of the value of this metric differs between the medical and network contexts. Medical data, being highly sensitive, necessitates synthetic data generation that respects the privacy of the patients whose data is used for learning. Thus, the *MD* score should be as low as possible. On the other hand, in a network context, encountering duplicated NFs, such as common DNS or NTP requests, is normal. Thus, we argue that Novelty in synthetic data should be close to Novelty observed in real data: synthetic data with a low *MD* score fails to capture the inherent duplication of network data.

4.5. Compliance evaluation

To evaluate the Compliance of the generated NFs, we adapt the *Domain Knowledge Check* (*DKC*) proposed

TABLE 3: List of Tests carried out by the metric **DKC**, with the network rules they are assessing and the Experiment they are associated with

Rule	Features	Experiments		
		Short	Long	UGR
If the flow has flags, then the Protocol is TCP	Flags, Protocol	✓	✓	
At least one IP Address of the flow must be private	Src IP Addr, Dst IP Addr	✓	✓	
If one of the ports is either 80, 443 or 8080, then the Protocol is TCP	Dst Port, Src Port, Protocol	✓	✓	✓
If one of the ports is 53, then the Protocol is UDP	Dst Port, Src Port, Protocol	✓	✓	✓
If Source Port is 53, then the Destination IP Address is private	Dst IP Addr, Source Port	✓	✓	
If one IP Address is public, then Destination Port is not 137/138	Src IP Addr, Dst IP Addr, Dst Port	✓	✓	
If Destination IP Address is public, then Source Port is not 80/443/8080	Src Port, Dst IP Addr	✓	✓	
If one port is ephemeral, then the other is not an application port	Src Port, Dst Port	✓	✓	✓
UDP or TCP flows have more than 0 byte	Protocol, Bytes	✓	✓	✓
An ICMP flow have 0 byte	Protocol, Bytes	✓	✓	✓
If the number of Packets is greater than 1, then Duration is greater than 0	Packets, Duration	✓	✓	✓
The Duration is not greater than the sum of the inter arrival times	Packets, IATs, Duration		✓	

Short: CICShortFeatureSet, Long: CICLongFeatureSet, UGR: UGR

in [22] to our context by adapting the test to our dataset. It consists in a set of test that the generated NFs should pass, which verify that the generated NF does respect some common network rules (Flags only on TCP NFs, ICMP NFs should be empty...). It is essential to emphasize that the set of tests, or set of rules, is feature-dependent and should be customized accordingly for each set of features one might want to generate. The specific test that we have used on the different feature sets described in Section 6 are presented in Table 3. For each test, the table indicates what are the feature tested on on which dataset we have performed it. This list is not an absolute list, depending on the feature on is generating some test might varies.

5. BNs for NF generation: strategic choices

We present now how a BN can addresses the challenges of NF generation, as raised in subsection 2.3.

5.1. Choice of the Structure Learning Algorithm

For choosing the type of structure learning algorithm, we test various algorithms from Python's `bnlearn` library; we have implemented an experiment detailed in Appendix C. Based on the result, we chose Hill-Climbing as our study's structure algorithm.

5.2. Discretization of numerical features

As said in Subsection 2.3, applying BNs to NF generation requires discretizing the numerical features of NFs. To this end, we introduce two discretization strategies. The first strategy, named $\mathbf{BN_{bins}}$, regroups values in quantiles. The second strategy, named $\mathbf{BN_{GM}}$, leverages a Variational Gaussian Mixture Model (VGMM) [46] fitted on the marginal distribution of the numerical features to map the numerical values to the index of their nearest Gaussian kernel. Both strategies are experimentally compared in Section 6. To convert the discretized values back into the original continuous space: with $\mathbf{BN_{bins}}$, the values are drawn uniformly from the corresponding bin's range; with $\mathbf{BN_{GM}}$, the value is sampled from the corresponding Gaussian kernel.

5.3. Reducing cardinalities of categorical features

As said in Subsection 2.3, cardinalities of discrete features greatly impact the size of a BN. However, some NFs features have high cardinalities, such as IP addresses (4 billion values) and ports (65536 values). It is, therefore, necessary to reduce the cardinality of those features.

Ports. While port values vary from 0 to 65536, most high values hold negligible information regarding communication content. This is especially relevant for ephemeral ports. We assign a single value to all ephemeral port instances to reduce the number of distinct port values. Even among non-ephemeral ports, the distribution of ports is highly concentrated on a few ports, which appear much more frequently in practice than others (like ports 80, 443, and 53, for example). In our work, we consider the minimal number n of the most frequent port values such that the set of ports up to the n-th covers a majority of NFs (above 50%). In UGR'16 and CIC-IDS2017, n is 30. The distribution of NFs according to the 30 most frequent port values is displayed in Appendix D

IP Addresses. IP addresses can be divided into public and private categories. For NIDS evaluation, private IP addresses hold intrinsic significance, denoting internal hosts requiring protection. Conversely, besides designating external hosts, public IP addresses do not convey additional information. Therefore, public IP addresses are often anonymized in the NIDS research [47]–[49]. We assign a single value to all public IP addresses to reduce the cardinality of IP values.

6. Experiments

In this section, we present the quality evaluation results on three benchmark datasets using the proposed metric system. We aim to compare the quality of the synthetic NF data produced by the proposed BN-based method and state-of-the-art GAN-based baselines.

6.1. Experimental protocol

To comprehensively evaluate the quality of the synthetic NFs, we introduce seven data generation methods in the comparative study:

- The two variants of the proposed BN-based NF generation method, which differ on the discretization preprocessing stage. These variants are the two introduced in Section 5: $\mathbf{BN_{GM}}$ and $\mathbf{BN_{bins}}$
- Three state-of-the-art GAN-based approaches[1]: **E-WGAN-GP** [22] because this method holds significance as one of the foundational works in the realm of GAN-based NF generation; **NetShare** [19] because it utilizes a unique tool of sequence generation for NF data, in addition to being the most recent work; and **CTGAN** [32] because, despite it being initially designed for general tabular data generation, it was used in several NF generation studies [27], [28].
- A naive approach (called **Naive Sampler** or **Naive** in the rest of the article). This **Naive Sampler** allows to assess the need for more complex and costly methods. The Naive Sampler draws a value from the training set for each attribute to generate data. The sampling process of each attribute is independent. Consequently, the generated data is not realistic nor compliant with network protocols. However, we may expect them to have a high Novelty level.
- Finally, a Real test set of real samples is different from the one used in training but captured in the same environment and, therefore, with the same characteristics.

The comparative test between these seven approaches is organized using the evaluation metrics we propose in Section 4 and based on three datasets with different features.

Firstly, We build two datasets from CIC-IDS2017 [49], using an updated version of CICFlowMeter [50], extracting unidirectional features. We opted to use the CIC-IDS2017 dataset as the reference [50] points out, allowing us to correct errors in that particular dataset. This is not the case with more recent datasets. We prioritized the quality of the training data over its freshness. The first dataset is named **CICSmallFeatureSet**. It is characterized by a limited number of network features (11). In contrast, the second dataset, CICLongFeatureSet, encompasses a larger feature set (30 features). By comparing the results on **CICSmallFeatureSet** and **CICLongFeatureSet**, we have two objectives. First, we aim to verify the impact of feature dimensions over the quality of generated NFs. Second, as **CICLongFeatureSet** embodies a more intricate dataset characterized by more numerical features (as shown in Appendix E), this inherent complexity poses a greater challenge for BNs.

Lastly, we consider a third dataset from UGR'16 [47], which contains real-world NFs (represented using 8 features) recorded by a Spanish ISP. We chose to work with a subset preprocessed by NetShare authors and ranges within the third week of March 2016 [19]. In our study, we denote this subset as UGR. We choose this specific subset to facilitate a more equitable and meaningful comparison with NetShare. Comparing over that dataset will show how the data generation methods perform with real-world traffic compared to testbed-produced simulations.

Before using these three datasets, we apply to them the data preprocessing technique described in Section 5. Therefore, we restrict the possible values of ports to the 30 most frequent ones: if a port value is not among them, we assign the arbitrary value *99999* to it. This value will serve all port values not frequent enough in our dataset. Similarly, we apply the default IP *0.0.0.0* to all public IP addresses. This value is the default route so that it could represent all external IP addresses. This is done for **CICSmallFeatureSet** and **CICLongFeatureSet**. Since **UGR** is solely composed of external IP NFs, we decide not to consider the IP features for this dataset.

6.2. Experimental results

This section presents our results on the three datasets: **CICSmallFeatureSet**, **CICLongFeatureSet**, and **UGR**. For each of them, we report the data quality measurement using the 4 criteria (Realism, Diversity, Novelty, and Compliance) and discuss the corresponding observations.

6.2.1. Results on CICSmallFeatureSet Data. We train the 7 models on **CICSmallFeatureSet** and use these models to produce synthetic NFs. The network traffic generated by each model is evaluated according to the metrics presented in Section 4. The results are presented in Table 4. For each metric and each model, the evaluation was done on 20 different generated sets, and the value in Table 4 are the medians of the measurement. The color indicate the ranking in respect to every metric.

Realism. In evaluating the Realism of synthetic NFs, the *Density* metric reveals that Bayesian Networks (BNs) outshine GAN-based methods by more closely aligning synthetic data distributions with those of real data. Further examination using *CMD* and *PCD* metrics, which assess the preservation of statistical correlations among attributes, shows that BNs excel due to their design for learning variable dependencies, evidenced by their consistently lower scores, indicating superior correlation preservation. In contrast, the Naive Sampler and NetShare struggle with higher *CMD* and *PCD* scores, reflecting poor correlation capture, while E-WGAN-GP and CTGAN demonstrate improved but varying performance in these metrics. This establishes a clear ranking across the *Density*, *CMD*, and *PCD* evaluations, with BNs at the forefront for their robustness in maintaining the integrity of real data's distribution and correlations, followed by GAN-based methods, which still manage to surpass the lesser performances of NetShare and the Naive Sampler.

Diversity. For Diversity, *Coverage* offers an initial global perspective, with BNs notably outperforming other techniques. When delving into the Diversity of feature marginal distributions through *JSD* and *EMD* metrics, the Naive Sampler emerges as exceptionally proficient, achieving low scores due to its strategy of directly sampling from real marginal distributions. Despite $\mathrm{BN_{GM}}$

1. We initially wanted to include STAN [20] in our experiment, but were unable to reproduce the results. Unfortunately, we had to rule it out.

TABLE 4: Comparison between 7 traffic generation methods on CICSmallFeatureSet using all the quality metrics .

	Description	Real data	Naive	BN_{bins}	BN_{GM}	CTGAN	E-WGAN-GP	NetShare
JSD	Realism and Diversity for categorical features (↓)	**0.017**	**0.017**	0.025	0.031	0.148	0.090	*0.23*
EMD	Realism and Diversity for numerical features (↓)	**0.002**	**0.002**	0.014	*0.080*	0.016	0.055	0.062
CMD	Realism of correlation between categorical features (↓)	**0.013**	0.160	**0.018**	**0.018**	0.126	0.071	*0.257*
PCD	Realism of correlation between numerical features (↓)	**0.761**	1.186	**0.630**	0.891	0.949	1.152	*1.191*
Density	Realism of data distribution (↑)	**1.000**	*0.079*	**0.906**	0.887	0.867	0.862	0.391
Coverage	Diversity of data distribution (↑)	**0.967**	*0.161*	0.952	**0.955**	0.903	0.655	0.214
MD	Novelty (=)	**7.175**	5.766	6.929	**6.987**	6.862	6.864	*5.247*
DKC	Compliance (↓)	**0.003**	*0.088*	0.004	0.003	0.008	**0.002**	0.023
Global Rank	Average Ranking (↓)	**-**	4.375	**1.75**	2.375	3.625	3.375	*5.5*

The Test columns serves as a standard. For each metric : *Red* indicates the worst model, **Orange** the second-best model and **Green** the best model. (↑): Higher is better, (↓): Lower is better, (=): Closest to the real data is better. The last line gives the average rank given by all metrics to each model, and is here just as an indication of overall performance.

showing commendable performance with a *JSD* of 0.031 for discrete features, it falls behind GAN-based methods in accurately capturing the distribution of numerical features, evidenced by an *EMD* score of 0.080, possibly attributed to its GMM-based discretization process not adequately learning the marginal distribution of numerical features. However, the limited number of numerical features in the dataset (3 out of 11) minimizes the impact on the overall Density performance of BN_{GM}, which still achieves a *Coverage* score of 0.955. This highlights the effectiveness of BN_{GM} in generating synthetic data closely resembling the real distribution, albeit with some limitations in capturing the diversity of numerical feature distributions.

Novelty. *MD* is employed to detect that some generated samples are copies of source samples. A higher *MD* value suggests a greater likelihood of copied training data in the generated NFs. For the Real data, the *MD* is 7.175, indicating that some copying should be expected in generated data. The BNs have thus the *MD* value closest to the Real data (6.929 and 6.987 for BN_{bins} and BN_{GM}, respecitvely). E-WGAN-GP and CTGAN seem less prone to data-copying. NetShare has the lowest *MD*: this is due to the generation of unrealistic samples, which therefore have a lower chance to be copied from the source data.

Compliance. We evaluate the Compliance by submitting the generated NFs of our different models to a series of tests defined in subsection 4.5. The *DKC* score is defined as the percentage number of failed tests over the whole set of Compliance tests. A higher *DKC* score, such as for NetShare (2.3%) and the Naive Sampler (8.8%), indicates that the generated traffic is less compliant to the network protocols. Except for NetShare, all models have a pretty low *DKC* score. We can also notice that the *DKC* score of the real data is not null. That means that some abnormal NFs are present in the initial dataset.

General Observation. The analysis of the **CICSmallFeatureSet** data reveals that BN-based methods outperform the GAN ones in terms of both generation quality and computational efficiency, with NetShare underperforming even compared to the Naive Sampler. Despite the occasional superiority of GAN models in specific metrics like *EMD*, BNs demonstrate robust performance across a range of evaluations, securing a favorable average - the mean of the ranks across all the metrics- ranking highlighted in the global assessment. The struggle of GAN-based methods to maintain strong performance in metrics that evaluate the preservation of variable dependencies, such as *CMD* and *PCD*, underscores the ongoing challenge these models face in accurately capturing correlations among variables, which BN-based approaches seem better equipped to handle.

6.2.2. Results on CICLongFeatureSet Data. This experiment illustrates how the data generation methods behave with more network features in the training dataset. **CICLongFeatureSet** data has 20 more numerical features than **CICSmallFeatureSet** (see Appendix E). This is expected to challenge the data generation methods involved, especially BNs. The results are shown in Table 5.

Realism. The *Density* score shows that CTGAN and E-WGAN-GP learned the joint distribution of the training data point well. Looking at *PCD* and *CMD*, we can see that the BNs and CTGAN learned the correlation between features well. Surprisingly, the Naive Sampler reproduced a correlation among discrete features, as indicated by its *CMD* of 0.079. *JSD* indicates that the BN methods learned the marginal distribution better than the others. *EMD* shows that BN_{GM} did not handle the numerical features, surely due to its discretization (0.087, the highest of all the models).

Diversity. BN_{GM} and CTGAN have the highest *Coverage* score (0.778 and 0.809, respectively). They managed to capture the important variance of the data distribution. Other methods are not as good, especially NetShare and the Naive Sampler (0.022 and 0.014, respectively).

Novelty. As per the *MD* metric, all the involved models produce synthetic samples that are not direct clones of training data. Compared to the previous experiment with **CICSmallFeatureSet**, which contains fewer features to learn, our different models introduce little novelty compared to the real data.

Compliance. The increased number of numerical features does not impact the DKC metric. This shows producing unrealistic values for the features do not hinder the compliance of the data, as discussed in Subsection 4.5.

General Observation. In our experiment with the **CICLongFeatureSet** Data, BN_{GM}, while proficient in learning discrete features, generates less realistic samples than CTGAN, which is attributed to the challenge of handling numerous numerical features. NetShare demonstrates that

TABLE 5: Comparison, according to all our metrics, of 7 data generation methods using CICLongFeatureSetDat.

	Description	Real data	Naive	BN_{bins}	BN_{GM}	CTGAN	E-WGAN-GP	NetShare
JSD	Realism and Diversity for categorical features (↓)	**0.018**	**0.018**	0.116	**0.078**	0.139	0.109	*0.359*
EMD	Realism and Diversity for numerical features (↓)	**0.001**	**0.001**	**0.014**	*0.087*	0.017	0.065	0.031
CMD	Realism of Correlation between categorical features (↓)	**0.010**	**0.079**	0.175	**0.032**	0.124	0.096	*0.384*
PCD	Realism of Correlation between numerical features (↓)	**2.345**	6.517	**2.761**	**3.456**	3.700	4.843	*8.404*
Density	Realism of data distribution (↑)	**0.997**	*0.051*	0.320	0.486	**0.647**	**0.492**	0.263
Coverage	Diversity of data distribution (↑)	**0.969**	*0.085*	0.647	**0.778**	**0.809**	0.416	0.120
MD	Novelty (=)	**7.612**	5.297	4.251	**5.663**	5.522	**5.835**	*2.539*
DKC	Compliance (↓)	**0.003**	*0.128*	0.060	**0.051**	**0.052**	0.092	0.055
Global Rank	Average Ranking (↓)	**-**	3.875	3.5	**2.375**	**2.75**	3.375	*5.125*

TABLE 6: Comparison, according to all our metrics, of 7 data generation methods using UGR Data.

	Description	Real data	Naive	BN_{bins}	BN_{GM}	CTGAN	E-WGAN-GP	NetShare
JSD	Realism and Diversity for categorical features (↓)	**0.067**	**0.0068**	**0.066**	0.070	0.218	0.105	*0.399*
EMD	Realism and Diversity for numerical features (↓)	**0.002**	**0.002**	0.018	0.007	*0.029*	*0.029*	**0.003**
CMD	Realism of Correlation between categorical features (↓)	**0.037**	0.223	**0.031**	**0.040**	0.209	0.050	*0.578*
PCD	Realism of Correlation between numerical features (↓)	**0.373**	*1.222*	**0.452**	0.738	0.863	1.219	**0.542**
Density	Realism of data distribution (↑)	**0.951**	0.355	0.701	**0.855**	0.486	**0.702**	*0.027*
Coverage	Diversity of data distribution (↑)	**1.000**	0.805	0.792	**0.998**	0.802	**0.996**	*0.076*
MD	Novelty (=)	**8.692**	7.519	8.312	**8.316**	7.447	**8.341**	*5.675*
DKC	Compliance (↓)	**0.006**	0.079	**0.005**	**0.005**	0.019	**0.004**	*0.129*
Global Rank	Average Ranking (↓)	**-**	3.75	**2.625**	**2.25**	4.375	3.0	*5.0*

TABLE 7: Computing costs of the three experiments.

Model	Duration										
	1: CICShortFeatureSet			2: CICLongFeatureSet			3: UGR				
	Prep.	Train.	Samp.	Prep.	Train.	Samp.	Prep.	Train.	Samp.	**Epochs**	**Hardware**
BN_{GM}	00:22	00:36	-	1:48	00:44	-	00:19	00:32	-	-	(1)
BN_{bins}	-	00:39	-	-	00:45	-	-	00:32	-	-	(1)
E-WGAN-GP	-	00:11	00:46	-	00:35	0:55	-	00:20	0:21	100	(1)
CTGAN	-	15:02	-	-	19:30	-	-	13:27	-	300	(2)
NetShare	-	20:42	00:39	-	27:31	00:56	-	(≃100:00)	-	100	(2)

The format is hours:minutes. Prep.: Preprocessing of the data, Train.: Training of the model, Samp.: Sampling from the model. Hardware configurations: (1) Laptop CPU / 32 GB RAM; (2) A40 GPU / 48 GB VRAM. The training of NetShare on UGR is the order of magnitude given by the authors in their paper. A cell with "-" denotes that either this step does not occur or that it is so short that we consider its time to be negligible

generating unrealistic samples does not compromise Diversity or Compliance. This underscores the need to assess the four evaluation criteria individually. Comparing BN_{GM} and CTGAN reveals that evaluating Realism and Diversity solely based on marginal and conditional distributions is insufficient. Despite outperforming CTGAN on many metrics, BN_{GM} fails to model joint distributions due to its struggle with numerical feature distribution learning. Overall, BN_{GM} excels across more metrics, affirming Bayesian Networks' superiority in this experiment, with CTGAN showing promising results albeit at a high computational cost. Additionally, BN_{GM} yields higher-quality samples at a lower training expense.

6.2.3. Results on UGR Data. In the two previous experiments, the training dataset is composed of traffic generated inside the same physical testbed. By using a real-world dataset, we want to observe if the evaluation results of the synthetic traffic are consistent with real traffic as the training data. The results are presented in Table 6

Realism. Aligned with the experiment on **CICSmallFeatureSet**, the BN-based methods exhibit high Realism in *Density*, *CMD* and *PCD* scores. Among GAN approaches, NetShare reaches good results in *EMD* (0.003), while E-WGAN-GP manages to achieve good global Realism on *Density* (0.702).

Diversity. Globally, BN-based methods are better than the GAN-based methods in learning the marginal distribution of features, as we can see with *JSD* and *EMD*. The *Coverage* metric ranks E-WGAN-GP first and NetShare last. Most of the evaluated models (apart from NetShare) produced diverse samples, with BN_{GM} on top. Same as for the first experiment, this might be due to the simpler distribution, which is, therefore, easier to cover. It can be worth noting that CTGAN while having good Coverage (0.802), has a rather low Density (0.486). This implies that while managing to cover pretty much all of the real distribution, CTGAN does create unrealistic samples: a phenomenon known as *mode invention* [39]. A more extensive illustration of this phenomenon is given in Appendix G.

Novelty. Due to the simpler distribution (fewer features, and smaller cardinality per discrete feature) in the

UGR data, it is easier for a data generation model to produce synthetic data close to a training data sample. As a result, the *MD* scores of all models except NetShare are close to the Real data's *MD*. This experiment shows that the simpler the dataset is, the more likely the generated model will produce a copy of the original data.

Compliance. All the data generation methods, except for NetShare and Naive Sampler, have a good *DKC* score. It shows that they are able to produce traffic NF compliant to network protocols. NetShare fails to generate valid traffic, due to its inability to encode correlation between numerical features.

General Observation. Our results show that BN-based approaches are better at preserving Realism, Diversity, and Compliance in generated NFs compared to GAN-based methods, particularly when dealing with datasets with smaller feature dimensions. Independent evaluation of each criterion revealed that CTGAN is particularly prone to mode invention, a common GAN problem detailed in Appendix G. Moreover, NetShare showed subpar performance in this experiment, even when assessed using data provided directly by its creators, highlighting the strengths of BN-based methods in terms of efficiency and effectiveness in data generation tasks.

6.2.4. Computational Costs. Table 7 summarizes the computational costs across three key steps: preprocessing, training, and sampling, along with the hardware configurations used in our experiments. In our analysis, BN_{bins} consistently emerges as the most efficient model for synthetic sample generation. In Experiment 1, BN_{bins}, BN_{GM}, and E-WGAN-GP exhibited the fastest performance. However, due to the complex IP2Vec embedding reconstruction in E-WGAN-GP and BN_{GM}'s Gaussian Mixture Models (GMMs) training requirement for numerical features, BN_{bins} proved to be the most efficient in terms of time and resources. Conversely, CTGAN and NetShare demanded significantly higher computational resources. This trend persisted in Experiment 2, where the rise in numerical features notably extended preprocessing times for BN_{GM} and training times for all GAN models, with CTGAN's training time increasing by approximately 50%.

7. Key takeaways and conclusion

We provide in this paper a comprehensive and interpretable benchmark system to evaluate the quality of synthesized network flows from multiple aspects: in Section 4, we introduce a comprehensive evaluation framework assessing *Realism, Diversity, Novelty, and Compliance* of the generated network flows. This benchmark system encompasses 8 distinct metrics for network flow generation assessment, which allows pinpointing the limitations and statistical bias existing in the synthesized network flows. Empirical observations across popularly used network traffic datasets highlight the necessity of evaluating synthetic network flows using multiple criteria to avoid bias towards specific metrics yet ignoring the others. For instance, while E-WGAN-GP shows the lowest *DKC* score on **CICShortFeatureSet**, it performs worse than our BN-based methods in terms of Realism and Diversity, as seen in *JSD*, *EMD*, and *PCD* scores. Focusing on a single criterion may lead to erroneous choices among different network flow data generation methods. Our novel benchmark helps to prevent such misleading model selections.

We propose a novel approach for network flow generation using BNs and show their superiority over GAN-based approaches. Empirical comparisons in Section 6 consistently demonstrate that GANs, despite their high model complexity and intensive training cost, are outperformed by our BN-based network flow generation methods. Despite their success in computer vision, GANs appear less effective in generating tabular data with mixed feature types, e.g. network flows. Besides, as a black-box model, GANs can not capture the correlation between different network features explicitly. In top of that, despite many upgrades, GANs are still prone to mode collapse phenomenon.

Our study advocates for BNs as the preferred solution to network flow generation. As network traffic is usually produced by programmed activities and restricted by network protocols, the conditional dependency between network flow features characterizes network traffic patterns. The interpretability of BN also allows us to directly estimate these conditional dependency relations, facilitating human users to monitor and understand the data generation process.

The results obtained from the two CIC-IDS-2017 and UGR'16 datasets are promising, but remain on datasets with low cardinality, as discussed in Section 5 . Future studies could extend our findings by incorporating IPv6 addresses and additional Ports. Moreover, while this initial study does not incorporate temporal dependencies, we hypothesize that our BN-based approach has the potential to capture such dynamics if well encoded. Addressing temporal correlations explicitly in network traffic synthesis will be a direction for future research. Lastly, another interesting future work could consist in using the NFs generated by our method in an evaluation pipeline for a real NIDS system. This will assess the "usability" of our generated data in a specific cybersecurity context.

8. Acknowledgements

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations[2].

This work has been partially realized thanks to a doctoral grant from Creah Labs and partially supported by the French National Research Agency under the France 2030 label (Superviz ANR-22-PECY-0008). The views reflected herein do not necessarily reflect the opinion of the French government.

2. see https://www.grid5000.fr

References

[1] R. Uetz, C. Hemminghaus, L. Hackländer, P. Schlipper, and M. Henze, "Reproducible and adaptable log data generation for sound cybersecurity experiments," in *Annual Computer Security Applications Conference*, 2021, pp. 690–705.

[2] M. Landauer, F. Skopik, M. Frank, W. Hotwagner, M. Wurzenberger, and A. Rauber, "Maintainable log datasets for evaluation of intrusion detection systems," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[3] S. Abt and H. Baier, "A plea for utilising synthetic data when performing machine learning based cyber-security experiments," in *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*, ser. AISec '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 37–45. [Online]. Available: https://doi.org/10.1145/2666652.2666663

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," New York, NY, USA, p. 139–144, oct 2020. [Online]. Available: https://doi.org/10.1145/3422622

[5] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks, "Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models," *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[6] A. B. Sai, A. K. Mohankumar, and M. M. Khapra, "A survey of evaluation metrics used for nlg systems," *ACM Comput. Surv.*, vol. 55, no. 2, jan 2022. [Online]. Available: https://doi.org/10.1145/3485766

[7] D. Heckerman, *A Tutorial on Learning With Bayesian Networks*, 10 2008, vol. 156, pp. 33–82.

[8] E. Choi, S. Biswal, B. Malin, J. Duke, W. F. Stewart, and J. Sun, "Generating multi-label discrete patient records using generative adversarial networks," in *Proceedings of the 2nd Machine Learning for Healthcare Conference*, ser. Proceedings of Machine Learning Research, F. Doshi-Velez, J. Fackler, D. Kale, R. Ranganath, B. Wallace, and J. Wiens, Eds., vol. 68. PMLR, 18–19 Aug 2017, pp. 286–305. [Online]. Available: https://proceedings.mlr.press/v68/choi17a.html

[9] A. Meddahi, H. Drira, and A. Meddahi, "Sip-gan: Generative adversarial networks for sip traffic generation," in *2021 International Symposium on Networks, Computers and Communications (ISNCC)*, 2021, pp. 1–6.

[10] A. Cheng, "Pac-gan: Packet generation of network traffic using generative adversarial networks," in *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2019, pp. 0728–0734.

[11] L. D. Manocchio, S. Layeghy, and M. Portmann, "Flowgan-synthetic network flow generation using generative adversarial networks," in *2021 IEEE 24th International Conference on Computational Science and Engineering (CSE)*. IEEE, 2021, pp. 168–176.

[12] P. Wang, S. Li, F. Ye, Z. Wang, and M. Zhang, "Packetcgan: Exploratory study of class imbalance for encrypted traffic classification using cgan," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7.

[13] Z. Lin, Y. Shi, and Z. Xue, "Idsgan: Generative adversarial networks for attack generation against intrusion detection," in *Advances in Knowledge Discovery and Data Mining: 26th Pacific-Asia Conference, PAKDD 2022, Chengdu, China, May 16–19, 2022, Proceedings, Part III*. Berlin, Heidelberg: Springer-Verlag, 2022, p. 79–91. [Online]. Available: https://doi.org/10.1007/978-3-031-05981-0_7

[14] L. Han, Y. Sheng, and X. Zeng, "A packet-length-adjustable attention model based on bytes embedding using flow-wgan for smart cybersecurity," *IEEE Access*, vol. 7, pp. 82 913–82 926, 2019.

[15] M. R. Shahid, G. Blanc, H. Jmila, Z. Zhang, and H. Debar, "Generative deep learning for internet of things network traffic generation," in *2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC)*, 2020, pp. 70–79.

[16] S. Hui, H. Wang, Z. Wang, X. Yang, Z. Liu, D. Jin, and Y. Li, "Knowledge enhanced gan for iot traffic generation," in *Proceedings of the ACM Web Conference 2022*, ser. WWW '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 3336–3346. [Online]. Available: https://doi.org/10.1145/3485447.3511976

[17] R. H. Randhawa, N. Aslam, M. Alauthman, and H. Rafiq, "Evasion generative adversarial network for low data regimes," *IEEE Transactions on Artificial Intelligence*, 2022.

[18] Z. Lin, A. Jain, C. Wang, G. Fanti, and V. Sekar, "Using gans for sharing networked time series data: Challenges, initial promise, and open questions," in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 464–483. [Online]. Available: https://doi.org/10.1145/3419394.3423643

[19] Y. Yin, Z. Lin, M. Jin, G. Fanti, and V. Sekar, "Practical gan-based synthetic ip header trace generation using netshare," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, pp. 458–472.

[20] S. Xu, M. Marwah, M. Arlitt, and N. Ramakrishnan, "Stan: Synthetic network traffic generation with generative neural models," in *International Workshop on Deployable Machine Learning for Security Defense*. Springer, 2021, pp. 3–29.

[21] B. Dowoo, Y. Jung, and C. Choi, "Pcapgan: Packet capture file generator by style-based generative adversarial networks," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, 2019, pp. 1149–1154.

[22] M. Ring, D. Schlör, D. Landes, and A. Hotho, "Flow-based network traffic generation using generative adversarial networks," *Computers & Security*, vol. 82, pp. 156–172, may 2019.

[23] C. G. Cordero, E. Vasilomanolakis, A. Wainakh, M. Mühlhäuser, and S. Nadjm-Tehrani, "On generating network traffic datasets with synthetic attacks for intrusion detection," *ACM Transactions on Privacy and Security (TOPS)*, vol. 24, no. 2, pp. 1–39, 2021.

[24] N. Elmrabit, F. Zhou, F. Li, and H. Zhou, "Evaluation of machine learning algorithms for anomaly detection," in *2020 international conference on cyber security and protection of digital services (cyber security)*. IEEE, 2020, pp. 1–8.

[25] A. Goncalves, P. Ray, B. Soper, J. Stevens, L. Coyle, and A. P. Sales, "Generation and evaluation of synthetic patient data," *BMC medical research methodology*, vol. 20, no. 1, pp. 1–40, 2020.

[26] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, p. 214–223.

[27] T. J. Anande, S. Al-Saadi, and M. S. Leeson, "Generative adversarial networks for network traffic feature generation," *International Journal of Computers and Applications*, vol. 45, no. 4, pp. 297–305, 2023. [Online]. Available: https://doi.org/10.1080/1206212X.2023.2191072

[28] S. Bourou, A. E. Saer, T.-H. Velivassaki, A. Voulkidis, and T. Zahariadis, "A review of tabular data synthesis using gans on an ids dataset," *Information*, vol. 12, no. 09, p. 375, 2021.

[29] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," *Information Fusion*, vol. 81, pp. 84–90, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253521002360

[30] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci, "Deep neural networks and tabular data: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[31] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim, "Data synthesis based on generative adversarial networks," *Proc. VLDB Endow.*, vol. 11, no. 10, p. 1071–1083, jun 2018. [Online]. Available: https://doi.org/10.14778/3231751.3231757

[32] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, *Modeling Tabular Data Using Conditional GAN*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[33] J. Pearl, *Causality*. Cambridge university press, 2009.

[34] A. Borji, "Pros and cons of gan evaluation measures: New developments," *Computer Vision and Image Understanding*, vol. 215, p. 103329, 2022.

[35] S. Zhou, S. Ermon, A. Y. Ng, M. S. Bernstein, and S. U. C. S. Department, *On the Evaluation of Deep Generative Models*. Stanford University, 2021. [Online]. Available: https://books.google.fr/books?id=2qGDzgEACAAJ

[36] N. Patki, R. Wedge, and K. Veeramachaneni, "The synthetic data vault," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2016, pp. 399–410.

[37] G. Marti, "Corrgan: Sampling realistic financial correlation matrices using generative adversarial networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8459–8463.

[38] P. Zingo and A. Novocin, "Can gan-generated network traffic be used to train traffic anomaly classifiers?" in *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2020, pp. 0540–0545.

[39] A. Alaa, B. V. Breugel, E. S. Saveliev, and M. van der Schaar, "How faithful is your synthetic data? sample-level metrics for evaluating and auditing generative models," in *International Conference on Machine Learning*. PMLR, 2022, pp. 290–306.

[40] M. F. Naeem, S. J. Oh, Y. Uh, Y. Choi, and J. Yoo, "Reliable fidelity and diversity metrics for generative models," in *International Conference on Machine Learning*. PMLR, 2020, pp. 7176–7185.

[41] M. Lucic, K. Kurach, M. Michalski, O. Bousquet, and S. Gelly, "Are gans created equal? a large-scale study," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 698–707.

[42] P. Zingo and A. Novocin, *Introducing the TSTR Metric to Improve Network Traffic GANs*, 04 2021, pp. 643–650.

[43] A. Schoen, G. Blanc, P.-F. Gimenez, Y. Han, F. Majorczyk, and L. Mé, "Towards generic quality assessment of synthetic traffic for evaluating intrusion detection systems," in *RESSI 2022-Rendez-Vous de la Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information*, 2022.

[44] C. Meehan, K. Chaudhuri, and S. Dasgupta, "A non-parametric test to detect data-copying in generative models," Apr 2020.

[45] S. Molnár, P. Megyesi, and G. Szabó, "How to validate traffic generators?" in *2013 IEEE International Conference on Communications Workshops (ICC)*, 2013, pp. 1340–1344.

[46] D. M. Blei and M. I. Jordan, "Variational inference for dirichlet process mixtures," 2006.

[47] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro, and R. Therón, "Ugr'16: A new dataset for the evaluation of cyclostationarity-based network idss," *Computers & Security*, vol. 73, pp. 411–424, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404817302353

[48] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho, "Creation of flow-based data sets for intrusion detection," *Journal of Information Warfare*, vol. 16, pp. 41–, 12 2017.

[49] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *International Conference on Information Systems Security and Privacy*, 2018.

[50] M. Lanvin, P.-F. Gimenez, Y. Han, F. Majorczyk, L. Mé, and E. Totel, "Errors in the cicids2017 dataset and the significant differences in detection performances it makes," in *Risks and Security of Internet and Systems*, S. Kallel, M. Jmaiel, M. Zulkernine, A. Hadj Kacem, F. Cuppens, and N. Cuppens, Eds. Cham: Springer Nature Switzerland, 2023, pp. 18–33.

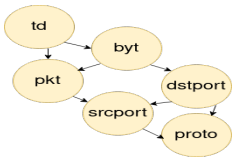
A. Formal definition of Bayesian Networks

BNs are statistical models that represent the probabilistic relationships among variables [33]. They are composed of a directed acyclic graph, where each node N is labelled by one feature X_N and is associated with a conditional probability table that describes the distribution $P(X_N \mid X_{Pa(N)})$ where $X_{Pa(N)}$ are the features of the parents of N. A BN can represent any probability distribution due to Bayes' theorem:

$$P(X) = \prod_N P(X_N \mid X_{Pa(N)}) \tag{3}$$

In order to learn the dependencies among the features of a given dataset, there are multiple families of BN learning algorithms: constraints-based (that relies on statistical tests), score-based (that optimizes a likelihood-based score), and hybrid. On Figure 1 we can see an example of BN (BN_{bins}) that was trained on **UGR'16** with the Hill-Climbing structure learning algorithm.

Figure 1: Example of Bayesian Network trained on UGR'16 with Hill-Climbing



Nodes are features, and arrows are dependencies.

B. JSD and EMD

JSD quantifies the similarity between the probability mass functions of real and synthetic data for a given discrete feature. It is a symmetric variant of the Kullback-Leibler divergence. Importantly, *JSD* is calculated independently for each variable, thus focusing solely on individual variables without capturing inter-variable dependencies. Eq. 4 gives the formulation of *JSD* for a discrete feature.

$$JSD(\mathrm{X_{source}} \parallel \mathrm{X_{gen}}) = \frac{D_{\mathrm{KL}}(X_{source} \parallel M) + D_{\mathrm{KL}}(X_{gen} \parallel M)}{2} \tag{4}$$

where $M = \frac{1}{2}(\mathrm{X_{source}} + \mathrm{X_{gen}})$ and $D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$.

EMD is calculated by analyzing a specific variable's real and synthetic Cumulative Density Functions (CDFs). This metric quantifies the *amount of mass* required to be displaced to transform the source CDF into the generated CDF. Eq. 5 gives the *EMD* between the value distributions of a continuous feature for source NF data and generated NF data.

$$EMD(\mathrm{X_{source}}, \mathrm{X_{gen}}) = \int |\mathrm{P}(\mathrm{X_{source}} \leq x) - \mathrm{P}(\mathrm{X_{gen}} \leq x)| \, \mathrm{d}x \tag{5}$$

C. Selection of a structure learning algorithm

To choose the structure learning algorithm type, we test various algorithms on the UGR'16 dataset and compute their respective BIC scores. Our Bayesian Network will try to represent the different dependencies among the features of that dataset with the lowest possible number of parameters.

TABLE 8: Comparison of different structure learning algorithms from Python's `bnlearn` library on UGR'16.

Structure Learning Method	BIC score (Lower is better)
Naive Bayes	1.94×10^7
Chow-Liu	1.80×10^7
Hill climbing	$\mathbf{1.68 \times 10^7}$

TABLE 9: Distribution of the 30 most frequent port numbers in CIC-IDS-2017 and UGR16 dataset

#	Port Numbers	Occurrence
1	53	24.193 %
2	80	12.427 %
3	443	9.601 %
4	123	0.944 %
5	137	0.502 %
6	25	0.434 %
7	0	0.224 %
8	22	0.185 %
9	8080	0.163 %
10	21	0.147 %
11	110	0.144 %
12	389	0.104 %
13	6000	0.083 %
14	88	0.082 %
15	445	0.077 %
16	3306	0.075 %
17	138	0.062 %
18	22001	0.059 %
19	8000	0.052 %
20	22011	0.049 %
21	3268	0.047 %
22	5060	0.045 %
23	143	0.045 %
24	161	0.044 %
25	8888	0.044 %
26	139	0.044 %
27	5353	0.043 %
28	465	0.040 %
29	5210	0.039 %
30	64887	0.038 %
	Sum	**50.036 %**

Given a collection of data points $\Omega = (x_i)_{i \leq n}$, a model θ, and k the number of parameters in θ, BIC is defined by Eq.6:

$$BIC(\theta \mid \Omega) = -2 \sum_{i=1}^{n} \log(\mathrm{P}(x_i|\theta)) + k \log(n) \tag{6}$$

A low BIC score indicates a model that balances good fit with few parameters (avoiding overfitting). As shown in Table 8, Hill-Climbing is the structure learning algorithm that yields the best BIC score on that experiment. We hence chose this algorithm for our study.

D. Global Port Distribution

In Table 9, we have reported the 30 most used port values in both the **CICIDS-2017** and **UGR'16**, alongside their global occurrences, ordered in descending order. We can see that some values (like 53, 443 or 80) are overwhelmingly represented. The 30th most frequent port value is 64887 (an ephemeral port) and occurs only 0.028% of the time.

E. Description of the features of our datasets

In Table 10, we list the different features used to describe network flows in all our experiments. In Experiments 1 and 2, the feature Flags only contain information about URG flags, RST flags, or PUSH flags. Information about other TCP flags like ACK, SYN, or FIN was unavailable in the dataset. On top of having the highest number of features, **CICLongFeatureSet** also has the highest number of numerical ones.

TABLE 10: Description of the features of each flow in the three datasets of Subsection 6.1

Feature	Type	Dataset		
		Short	*Long*	*UGR*
Source IP Address	categorical	✓	✓	
Source Port	categorical	✓	✓	✓
Destination IP Address	categorical	✓	✓	
Destination Port	categorical	✓	✓	✓
Protocol	categorical	✓	✓	✓
Timestamp	numerical		✓	✓
Day of the week	categorical	✓		
Hour of the day	numerical	✓		
Duration	numerical	✓	✓	✓
Number of packets	numerical	✓	✓	✓
Number of bytes	numerical	✓	✓	✓
Maximum length of a packet	numerical		✓	
Minimum lenght of a packet	numerical		✓	
Average length of a packet	numerical		✓	
Standard deviation of the length of packets	numerical		✓	
Sum of inter-arrival times	numerical		✓	
Average inter-arrival time	numerical		✓	
Standard deviation of the inter-arrival time	numerical		✓	
Maximum of the inter-arrival times	numerical		✓	
Minimum of the inter-arrival times	numerical		✓	
Flags inside the flow	categorical	✓		
Number of PUSH flags	numerical		✓	
Number of URGENT flags	numerical		✓	
Number of RESET flags	numerical		✓	
Sum of length of the headers	numerical		✓	
Average Number of Packets per second	numerical		✓	
Average of segment sizes	numerical		✓	
Average of Bytes/Bulk ratios	numerical		✓	
Average of Packets/Bulk ratios	numerical		✓	
Average of Bulk Rates	numerical		✓	
Number of packets inside a Subflow	numerical		✓	
Number of bytes inside a Subflow	numerical		✓	
Number of Bytes of the Init Window	numerical		✓	

Short stands for **CICShortFeatureSet**; *Long* for **CICLongFeatureSet**; and *UGR* for **UGR**

F. Example of generated network flows

In Tables 11 and 12, you can see the network flows generated in the context of Experiment 1 (see 6.2.1) by $\mathbf{BN_{bins}}$ and **NetShare**, respectively.

As said in the previous subsection, the *Flags* feature contains only information on three types of TCP flags in the source dataset. So it is expected that none of our

TABLE 11: Example of network flows generated by BN_{bins} on Experiment 1

	Day	Time	Duration	Proto	Src IP Addr	Src Pt	Dst IP Addr	Dst Pt	Packets	Bytes	Flags
1	4	18:10:02	0.08	UDP	192.168.10.8	51504	192.168.10.3	53	1	88	.....
2	3	17:17:02	5.66	UDP	192.168.10.8	49231	192.168.10.3	53	3	161	.....
3	0	19:56:16	59.36	TCP	192.168.10.19	41967	152.209.204.1	443	16	1409	..P..
4	2	19:47:39	115.85	TCP	192.168.10.15	46219	99.29.138.167	443	22	2758	..P..
5	4	13:28:26	0.03	UDP	192.168.10.1	53	192.168.10.3	61719	1	119	.....
6	4	13:38:46	119.01	TCP	192.168.10.5	53073	191.195.80.34	443	22	1235	..P..
7	1	19:35:24	0.07	UDP	192.168.10.3	53	192.168.10.17	58681	2	187	.....

TABLE 12: Example of network flows generated by NetShare on Experiment 1

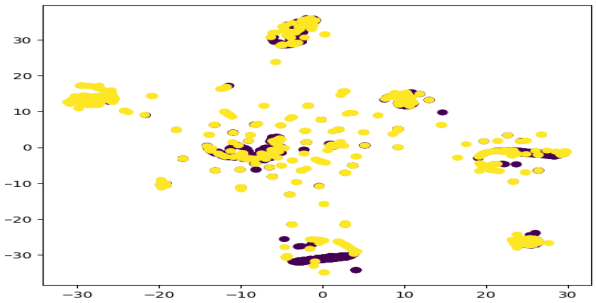
	Day	Time	Duration	Proto	Src IP Addr	Src Pt	Dst IP Addr	Dst Pt	Packets	Bytes	Flags
1	2	16:45:42	0.02	UDP	192.168.10.5	49082	192.168.10.3	53	1	143	.....
2	1	12:12:14	2.49	TCP	157.69.249.80	51243	192.168.10.5	49653	5	1293	...R..
3	2	18:29:28	2.95	TCP	192.168.10.16	47793	87.64.65.152	80	3	0	.....
4	3	12:33:07	0.09	UDP	192.168.10.3	64087	192.168.10.1	53	1	316	.....
5	0	14:09:52	70.28	TCP	192.168.10.25	61487	192.168.10.3	3268	26	1963	..P...
6	4	16:25:51	0.05	UDP	192.168.10.1	53	192.168.10.3	53593	1	351	.....
7	0	17:09:37	6.06	TCP	192.168.10.8	49844	109.182.162.153	443	14	1199	..P...

models produced any *ACK* flag or *SYN* flag for TCP flows. This is a shortcoming of the Dataset, not of our models.

We can see that the **NetShare** traffic exposes some serious shortcomings, like HTTP traffic without any bytes on line 3, or a flow with both *Dst Pt* and *Src Pt* as ephemeral ports on line 2.

G. Illustration of mode invention

Figure 2: Two-dimensional tSNE representation of UGR data and synthetic data generated by CTGAN.



Purple is for UGR data and **Yellow** is for synthetic data

In Subsection 6.2.3, when analyzing the result of CTGAN on the **UGR** dataset, we notice that its generation lacked Realism (*Density* of 0.486) yet had a pretty high Diversity (*Coverage* of 0.802). This behavior is a known pitfall of generative models and is usually labeled as mode invention.

In order to visualize mode invention, we decided to embed both the training data and the synthetic data generated by CTGAN in a tSNE representation. We also plot that representation in a two-dimensional graph in Figure 2.

We can see that every part of the real data distribution is pretty well covered by synthetic data (hence good Diversity), but the model also produces a consequent amount of synthetic samples that are not close to any real one (the little independent dots in the center of Figure 2), indicating low Realism.