DSEF: DNS Synthetic Traffic Evaluation Framework

Jihye Kim
Research Institute CODE
University of the Bundeswehr Munich
Neubiberg, Germany
jihye.kim@unibw.de

***Abstract*—DNS threats are central to cyber threat intelligence (CTI); however, access to real attack telemetry is constrained by privacy controls, operational limitations, and labeling costs—hindering reproducible research and the realistic evaluation of emerging detectors. Although a growing body of tools and ML-based generators can synthesize DNS traffic, the community still lacks a unified methodology to assess its protocol compliance, realism, semantics, and utility for defense. To address this gap, we introduce DSEF, the DNS Synthetic Traffic Evaluation Framework, a modular and generator agnostic framework for measuring the realism and defensive utility of synthetic DNS traffic. DSEF evaluates flows along four complementary axes: (i) *protocol correctness*, (ii) *distributional realism*, (iii) *semantic and behavioral realism*, and (iv) *downstream defensive utility*. By producing standardized, scenario-aware scores, DSEF enables consistent benchmarking across heterogeneous generator families. Using content-driven DNS threat scenarios, our results show that DSEF exposes distinct failure modes across replay, marginal resampling, and latent sampling generators, highlighting where synthetic traffic diverges from the reference distribution. DSEF offers a benchmark-ready foundation for the reproducible evaluation of synthetic DNS traffic and provides practical guidance for the safe and effective use of synthetic data in CTI workflows and security operations.**

I. Introduction

Domain Name System (DNS) is a foundational component of the Internet and a frequent vector or target for abuse, including malware, command-and-control (C2), phishing, and data exfiltration via DNS tunneling. Within cyber threat intelligence (CTI), DNS telemetry provides a valuable observational layer for profiling malicious infrastructure and attack behaviors. However, access to real-world DNS attack traces remains constrained due to privacy regulations, operational sensitivities, and high labeling costs [19], hindering reproducible research and systematic evaluation of detection and mitigation techniques.

Generating synthetic network traffic plays a critical role in addressing these limitations. Synthetic flows provide privacy preserving, controllable substitutes for operational telemetry, enabling detector benchmarking, robustness evaluation, and model development when real data cannot be shared [2], [11], [16], [39]. In the DNS domain in particular, synthetic traffic provides a practical opportunity to study attacker behaviors and to train detection models without requiring access to sensitive production logs. As a result, DNS traffic synthesis—spanning high-speed replay tools [15], [20], statistical and machine learning (ML)-based generators [39], and LLM-driven flow synthesis [11]—has become an increasingly active research direction. In practice, many of these generators are already used in security operations centers (SOCs) for detector testing, analyst training, and proof-of-concept evaluations [26].

Despite this progress, current work focuses primarily on *how to generate* synthetic DNS traffic [24], [35], leaving the open question of *how to evaluate* the fidelity and defensive utility of synthetic flows. Different generators represent different aspects of DNS behavior: some ensure protocol correctness but lack semantic coherence, while others model lexical or behavioral patterns but deviate from realistic statistical distributions. Without rigorous evaluation, synthetic DNS traffic may lead to misleading conclusions, overestimated detector performance, or scenario mismatches in defensive validation.

To address this gap, we introduce **DSEF**, the **DNS Synthetic Traffic Evaluation Framework**. DSEF provides a generator-agnostic validation pipeline that evaluates synthetic DNS flows along four complementary axes: (i) *protocol correctness* via RFC-compliant structural and cross-field checks; (ii) *distributional realism* through statistical divergence metrics; (iii) *semantic and behavioral realism* using lexical and embedding-based representations; and (iv) *downstream utility*, measured via separability probes and Train on Real, Test on Synthetic, Train on Synthetic, Test on Real (TRTS/TSTR)-style tests. Together, these axes quantify how closely synthetic flows approximate real DNS traffic and how useful they are for defensive workflows. This work makes four key contributions:

- We propose a reproducible validation framework that quantifies both the realism and utility of synthetic DNS traffic in a unified and interpretable manner.
- We design a generator-agnostic evaluation interface compatible with tool-, ML-, and LLM-based generators without requiring internal modifications.
- We formalize four complementary validation axes and demonstrate that they enable consistent, scenario-aware

Workshop on Security Operation Center Operations and Construction (WOSOC) 2026
23 February 2026, San Diego, CA, USA
ISBN 978-1-970672-03-9
https://dx.doi.org/10.14722/wosoc.2026.23006
www.ndss-symposium.org

benchmarking across heterogeneous generators.
- We provide standardized schemas, manifests, and containerized evaluation recipes that support reproducible assessments and per-scenario comparisons.

By providing a unified, DNS-specific evaluation pipeline, DSEF establishes a principled basis for benchmarking synthetic DNS traffic and supporting reproducible CTI and network security research.

II. Background and Related Work

A. DNS Traffic in CTI and Security Operations

DNS plays a central role in CTI and security operations because it reveals both infrastructure level and behavioral indicators of compromise that directly inform detection and response workflows. Malicious actors increasingly exploit DNS for domain based malware distribution, phishing, spam delivery, and data exfiltration via DNS tunneling. These activities manifest at the application layer through semantic artifacts—such as domain lexical characteristics, response patterns, and query–response dynamics—that are critical for attribution and detection. As a result, DNS telemetry provides a high-value observational layer for characterizing attacker behaviors and constructing threat intelligence models.

B. Synthetic Traffic Generation Landscape

Table I provides an overview of representative DNS-focused synthetic traffic generators across three categories—tool-based systems, ML-based models, and emerging LLM-based generators. Note that the table reflects the stated capabilities and evaluation focus of each work, rather than the inherent limitations of the underlying generator families. We compare these systems along four dimensions relevant to DNS and CTI—protocol correctness, statistical realism, semantic realism, and downstream utility—as well as qualitative notions of cost and deployment complexity.[1] These families reflect dominant paradigms in DNS synthetic traffic generation, each emphasizing different aspects of realism. In practice, many of these generators are already used in SOC environments for detector testing, analyst training, and proof-of-concept evaluations, often without systematic validation of their realism.

Tool-based generators, such as TRex and MoonGen [15], [20], replay pcap traces or forge packets at line rate with finely grained control over inter-packet timing and burstiness. Because they operate directly on RFC compliant packets and transport headers, they typically achieve near-perfect protocol correctness. However, their statistical realism is bounded by the coverage of the underlying traces or domain lists: they preserve low-level timing and rate properties but do not capture the full variability, correlation structure, or diurnal patterns of large scale DNS telemetry. DNS-specific tools (e.g., dnsperf, dnsgen [5], [14]) primarily replay fixed domain lists or generate deterministic query patterns, which limits their ability to capture behavior rich DNS threat scenarios. In contrast, tunneling frameworks (e.g., dnscat2, iodine [22], [40]) embed application payloads into DNS queries and responses, providing exfiltration semantics. While this gives them higher semantic realism compared to other tool-based generators, their behavior is still constrained by deterministic encoding schemes and fixed tunnel workflows. As a result, the downstream utility of tool-based generators—aside from tunneling tools—has primarily focused on performance benchmarking and protocol-level robustness testing, with more limited applicability to the training or evaluation of content-aware detectors.

ML-based generators learn statistical patterns from captured DNS data. Packet-level models, such as PAC-GAN [9], generate DNS packets by modeling low-level protocol fields and header structures, while DomainGAN and DOLOS [10], [16] focus on the lexical and distributional properties of DNS names and queries to generate benign-looking or evasive DNS traffic. CL-GAN [32] further introduces continual learning for adaptive DGA-style domain generation, but similarly operates at the level of lexical and statistical pattern modeling. With appropriate post processing and constraint enforcement, these models generally produce protocol-valid samples and replicate the marginal and joint feature distributions of the training data, resulting in strong statistical realism. However, their semantic modeling remains limited, as it does not explicitly capture DNS-level intent, protocol semantics, or CTI-relevant behavioral context, often leaving their downstream utility focused on expanding the statistical feature space.

LLM-based generators operate on textual or structured abstractions of network traffic. Generators such as PAC-GPT and GPT-on-the-Wire [25], [29] derive protocol structures or flow templates from JSON-, TLV-, or DSL-like descriptions, whereas TrafficLLM [11] directly emits multi-packet sequences (e.g., DNS, HTTP) guided by scenario-level attack descriptions. By conditioning on high-level prompts that describe campaigns, payload characteristics, or adversary goals, these systems often achieve substantially richer semantic realism than tool- or ML-based generators: synthesized domains, content, and flow patterns can closely mimic realistic phishing, exfiltration, or C2 activity. However, protocol correctness is not inherent to these models and typically depends on external builders or careful prompt engineering. Likewise, temporal structure is not explicitly modeled and remains largely prompt dependent, which can affect statistical realism.

Despite the proliferation of DNS traffic generators, the field lacks a credible way to determine whether these systems actually produce flows that resemble real DNS behavior or meaningfully support defensive evaluation. This absence of standardization has created a fundamental problem: synthetic DNS traffic is widely used in both research and operational settings, yet it is almost never rigorously validated for its impact on downstream security workflows. DSEF directly addresses this gap by establishing a unified benchmark for measuring the realism and defensive value of synthetic DNS flows. By providing a common validation interface, DSEF enables apples-to-apples comparisons across generator families.

[1]Cost and deployment are evaluated qualitatively based on required hardware (e.g., DPDK-capable NICs), configuration overhead, dependency footprint, and operator expertise, as reported in documentation and prior evaluations.

TABLE I: Synthetic traffic generators across Tool / ML / LLM families. (Legend: ✓ = yes, ˜ = partly, ✗ = no.)

Family	Name	Role / capability (DNS-specific)	Cost	Deployment	Protocol correctness	Statistical realism	Semantic realism	Downstream utility
Tool	TRex [20]	High-speed PCAP replay; accurate low-level packets; supports burstiness and timing control.	Med–High	Med	✓	˜	✗	˜
	MoonGen [15]	Scriptable DPDK-based generator with precise timestamps and custom packet emission.	Med	High	✓	˜	✗	˜
	dnsperf / resperf [14]	DNS query generator based on fixed domain lists; primarily used for performance benchmarking of recursive/authoritative servers.	Low–Med	Med	✓	˜	✗	˜
	dnsgen [5]	Raw socket DNS generator; supports large source-port ranges and adaptive sending patterns.	Low	Med	✓	˜	✗	˜
	dnscat2/iodine [22], [40]	DNS tunneling generators; true exfiltration semantics; widely used as ground truth tunnel traffic.	Low	Low	✓	˜	✓	✓
ML	PAC-GAN [9]	CNN GAN that generates raw IP packets, including DNS, by learning byte-level structure.	Med–High	Med	✓	✓	✗	˜
	DOLOS [16]	GAN-guided DNS exfiltration generator mimicking benign lexical and statistical query distributions.	Med–High	Med	˜	✓	˜	✓
	DomainGAN [10]	GAN-based generator for benign-like domains; learns Alexa lexical/n-gram structure.	Med–High	Med	˜	✓	˜	˜
	CL-GAN [32]	Continual-learning GAN + Transformer for adaptive DGA string generation.	Med–High	Med	˜	✓	˜	✓
LLM	PAC-GPT [25]	LLM that outputs structured protocol layouts and semantically coherent headers.	Med–High	Med	˜	✓	✓	˜
	GPT-on-the-Wire [13]	LLM for multi-protocol packet and conversation generation using Scapy builders and Mixture-of-Experts.	Med	Low–Med	˜	˜	✓	✓
	TrafficLLM [11]	Sequence-modeling LLM for multi-packet (e.g., DNS, HTTP) flows with scenario-driven behaviors.	Med–High	Med	˜	✓	✓	✓

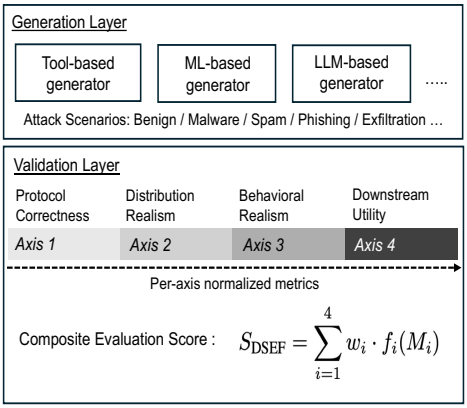

Fig. 1: DSEF overview. Three *Generation Layer* generator families feed a sequential *Validation Layer*: Axis 1 (Protocol Correctness), Axis 2 (Distributional Realism), Axis 3 (Semantic & Behavioral Realism), and Axis 4 (Downstream Utility).

III. DSEF Design

A. Framework Overview

DSEF is a modular, scenario-adaptive framework that evaluates synthetic DNS traffic using a multi-axis validation pipeline. Rather than enforcing a single end-to-end pipeline, DSEF treats generation and validation as loosely coupled components that communicate through a unified feature schema. This decoupling is deliberate: the *Validation Layer* can score flows produced by DSEF's own backbones, by external generators, or by real measurements, enabling *cross-source benchmarking* under a single set of realism criteria. The framework is organized into two conceptual layers: a *Generation Layer* and a *Validation Layer*. The *Generation Layer* accepts traffic from multiple generator families, following the taxonomy presented in Table I. Each generation module emits DNS flows encoded in a shared tabular schema that captures statistical (e.g., length, entropy, counts), lexical (e.g., n-grams, label structure), and temporal (e.g., inter-arrival times, burst patterns) attributes. This schema abstracts away generator-specific idiosyncrasies and preserves interoperability with downstream analysis tools and CTI workflows.

The *Validation Layer* quantifies realism along four complementary axes: (i) protocol correctness, (ii) distributional realism, (iii) semantic and behavioral realism, and (iv) downstream utility. Each axis aggregates a set of calibrated metrics M_i whose values are mapped to a common $[0, 1]$ scale by axis-specific normalization functions $f_i(\cdot)$. These functions are parameterized using held-out partitions of the reference dataset so that the resulting scores represent interpretable deviations from real DNS behavior. To integrate realism across axes, DSEF computes a scenario-aware composite score:

$$S_{\text{DSEF}} = \sum_{i=1}^{4} w_i \, f_i(M_i),$$

where w_i encodes the salience of axis i for a given DNS abuse category. For example, content-driven categories (e.g., phishing, exfiltration) emphasize semantic realism, whereas benign or malware traffic places greater weight on protocol and distributional fidelity. Figure 1 illustrates how heterogeneous generators plug into the *Generation Layer* and are evaluated in a unified manner by the sequential *Validation Layer*, yielding per-axis scores and a composite S_{DSEF} that can be compared across methods and scenarios.

B. Attack Scenarios

DSEF is instantiated on five activity categories: *Benign*, *Malware*, *Phishing*, *Spam*, and *Exfiltration*. While *Benign* and *Malware* traffic primarily require protocol and distributional fidelity, phishing and spam traffic exhibit campaign-level behaviors (e.g., repeated domain templates, attacker-specific lexical patterns) that should be preserved for realistic synthesis. *Exfiltration* is modeled using six payload types (i.e., Text, Image, Video, Compressed, Audio, Executable) in both light and heavy variants, determined by the underlying encoding and chunking strategy. This setup enables DSEF to evaluate whether generators can preserve a realistic QNAME structure, timing, and traffic volume while embedding rich payloads. At the same time, it allows the *Validation Layer* to assess its ability to distinguish between semantically empty DNS abuse and content-bearing tunneling activity.

C. Validation Interface

To make DSEF usable as a benchmark, the *Validation Layer* exposes a standardized interface and logging model.

a) Manifests: Each run produces a `manifest.json` file capturing the random seed, backbone configuration, hyperparameters, software versions, and dataset splits. This manifest enables the reproduction of a run or its replay under updated validation metrics.

b) Auto-validator: Given a set of flows encoded in the DSEF feature schema, the auto-validator computes: (i) protocol checks via DNS parsers with RFC 1035 cross-field constraints; (ii) distributional distances using standard divergence metrics (e.g., Jensen–Shannon, Wasserstein, MMD); (iii) temporal fidelity metrics such as autocorrelation (ACF/-PACF) over inter-arrival times; (iv) semantic and behavioral metrics, including a Fréchet Traffic Distance (FTD) computed in an embedding space; and (v) downstream utility indicators (e.g., TRTS/TSTR-style evaluations), which we outline as a direction for future studies. This automated validation module provides a unified execution point for all axis-level metrics, ensuring consistent scoring across generators and scenarios.

IV. Validation Methodology

A central goal of DSEF is to evaluate whether synthetic DNS traffic preserves the *functional*, *statistical*, and *semantic* properties present in a reference corpus of real-world DNS activity. In our instantiation, the BCCC-CIC-Bell-DNS-2024 dataset serves as this reference distribution: all generators are evaluated against its benign and malicious flow partitions. DSEF employs a four-axis validation methodology that scores synthetic traffic along: (i) protocol correctness, (ii) statistical and temporal fidelity, (iii) semantic and behavioral plausibility, and (iv) downstream defensive utility. All axes are calibrated using disjoint splits of the reference dataset, enabling distances to be interpreted as *how far a generator deviates from the realism exhibited within real DNS traffic distributions*. In principle, each axis can be instantiated with multiple metrics; in this work, we focus on a lightweight subset that can be computed reliably from the available flow-level features.

Let $\mathcal{D}_{\text{ref}}$ denote the reference flow set, and let $\mathcal{D}_{\text{syn}}$ denote the synthetic flows. Each flow f is mapped to a DNS feature vector $\phi(f)$ containing protocol fields, lexical attributes (e.g., entropy, n-grams), and temporal features (e.g., inter-arrival times, burst patterns). All validation splits use stratified partitioning to preserve class balance while preventing label leakage across training, validation, and test sets.

A. Axis 1: Protocol Correctness

Protocol correctness evaluates whether each synthetic flow conforms to DNS and EDNS(0) format specifications. DSEF measures correctness by applying a DNS/EDNS(0) parser to every flow and flagging any structural violations defined in RFC 1035 and EDNS(0) [1], [12], [23]. Let ν denote the proportion of flows that fail one or more constraints. The checks include: (i) QTYPE–RDATA mismatches (e.g., A/AAAA queries paired with incompatible RDATA formats), (ii) invalid QCLASS or RCODE assignments, (iii) NXDOMAIN inconsistencies such as ANCOUNT > 0, (iv) QNAME label-length or total-length violations (label ≤ 63, total ≤ 255), and (v) EDNS(0) DO/CD bit incoherence. The protocol-correctness score is defined as:

$$s_{\text{prot}} = 1 - \nu, \tag{1}$$

This axis isolates structural validity: even if a synthetic query is semantically plausible, it is penalized if it violates fundamental DNS format constraints. Therefore, Axis 1 penalizes only RFC-level structural errors; abnormal but syntactically valid values commonly found in attack traffic (e.g., high-entropy QNAMEs or unusual TTLs) do not reduce the score.

B. Axis 2: Distributional Realism

This axis measures whether synthetic flows preserve the statistical structure observed in real DNS telemetry. We evaluate distributional fidelity along three complementary dimensions: (a) feature-wise univariate similarity, (b) multivariate joint structure, and (c) temporal dynamics.

a) Univariate Feature Fidelity: For each feature k, let p_k^{ref} and p_k^{syn} denote the empirical marginal distributions computed from the reference and synthetic datasets. To quantify the deviation between these marginals, we adopt a combined divergence based on the Jensen-Shannon divergence (JSD) [17], [28], [34] and the 1-Wasserstein distance (W_1) [4], [31], [36], as these metrics capture distinct and complementary aspects of distributional drift.

Jensen-Shannon divergence: Let $H(P)$ denote the Shannon entropy of a discrete distribution P. For two distributions P and Q, the JSD is defined as:

$$\text{JSD}(P\|Q) = H\left(\frac{P+Q}{2}\right) - \frac{1}{2}\big(H(P) + H(Q)\big).$$

JSD is a symmetric, bounded f-divergence that measures the mismatch in distributional *shape* between two histograms. It is particularly sensitive to changes in modality, entropy, or concentration of probability mass—a useful property for DNS

features whose categorical or quasi-discrete distributions may shift under synthetic generation (e.g., RCODE frequencies, QTYPE usage). Small perturbations in frequency mass induce small increments in JSD, whereas mode creation or deletion yields substantially larger divergence values.

1-Wasserstein distance: For real-valued DNS features (e.g., TTL, query length, or response size), the absolute magnitude of feature values is operationally meaningful. The 1-Wasserstein distance (W_1), defined as:

$$W_1(P,Q) = \inf_{\gamma \in \Pi(P,Q)} \mathbb{E}_{(x,y)\sim\gamma}[\,|x-y|\,],$$

quantifies the minimum transport cost required to transform distribution P into Q. Unlike JSD, which is largely insensitive to uniform shifts, W_1 increases linearly with changes in the characteristic scale or location of a distribution (e.g., systematically larger TTL values or inflated packet sizes). This property makes W_1 particularly effective at detecting magnitude distortions that commonly arise in synthetic DNS traffic, even when the overall distributional shape is preserved. Because a generator may accurately match the shape of a distribution while distorting its scale, neither JSD nor W_1 alone provides a complete measure of feature fidelity. We therefore define the following composite univariate distance:

$$d_k^{\text{uni}} = \lambda_{\text{JSD}}\,\text{JSD}\big(p_k^{\text{ref}} \| p_k^{\text{syn}}\big) + \lambda_{\text{W}}\,W_1\big(p_k^{\text{ref}}, p_k^{\text{syn}}\big)\,,$$

where λ_{JSD} and λ_{W} normalize the relative contributions of the two terms based on reference-split variability. This combined metric yields a robust univariate drift signal that captures both structural changes (e.g., mode shifts, entropy changes) and systematic magnitude shifts (e.g., inflation or compression of DNS feature values).

b) Multivariate Joint Fidelity: Univariate divergences capture marginal alignment, but DNS flows also exhibit strong cross-feature dependencies. To evaluate whether a generator preserves this joint correlation structure, we compare the feature sets $X = \{\phi(f) \mid f \in \mathcal{D}_{\text{ref}}\}$ and $Y = \{\phi(f) \mid f \in \mathcal{D}_{\text{syn}}\}$ using the Maximum Mean Discrepancy (MMD) [18], [27], a kernel-based, nonparametric test of distributional equality:

$$d_{\text{MMD}}(X,Y) = \left\| \frac{1}{|X|}\sum_{x\in X} k(x,\cdot) - \frac{1}{|Y|}\sum_{y\in Y} k(y,\cdot) \right\|_{\mathcal{H}}^2, \quad (2)$$

where k is a characteristic kernel (typically Gaussian) and $\mathcal{H}$ is the associated reproducing kernel Hilbert space [7]. MMD is particularly suitable for DNS flows because it remains stable under high-dimensional tabular features and captures nonlinear dependencies without assuming a specific parametric form. In practice, we apply an RBF kernel to standardized numeric features and select the bandwidth using the median heuristic, following standard practice in kernel two-sample testing [18]. In this setting, $d_{\text{MMD}}(X,Y) = 0$ if and only if the underlying multivariate distributions coincide, while remaining computationally tractable at our flow-level sample sizes. Accordingly, DSEF adopts MMD as the default multivariate fidelity metric in Axis 2, with additional distances (e.g., Energy Distance [33]) available for diagnostic analysis.

c) Temporal Fidelity: DNS activity exhibits distinctive timing behaviors governed by caching dynamics, resolver burst patterns, and attacker workflows. To assess whether synthetic flows preserve such temporal signatures, we evaluate the inter-arrival time (IAT) series using two complementary diagnostics: (i) the empirical autocorrelation function (ACF) and (ii) the partial autocorrelation function (PACF) up to a fixed lag L [8]. Let $\rho^{\text{ACF}}\text{ref}(\ell)$ and $\rho^{\text{ACF}}\text{syn}(\ell)$ denote the lag-ℓ ACF of the reference and synthetic IAT sequences, and analogously $\rho^{\text{PACF}}\text{ref}(\ell)$ and $\rho^{\text{PACF}}\text{syn}(\ell)$ for the PACF. ACF captures short-range dependence and burstiness, while PACF isolates direct lag-ℓ effects by removing intermediate correlations. Together, they provide a lightweight but expressive summary of temporal structure, enabling DSEF to detect generators that match marginal IAT statistics yet fail to reproduce realistic timing dependencies. We summarize temporal deviation as:

$$d_{\text{temp}} = \frac{1}{2}\Big(\big\|\rho_{\text{ref}}^{\text{ACF}} - \rho_{\text{syn}}^{\text{ACF}}\big\|_2 + \big\|\rho_{\text{ref}}^{\text{PACF}} - \rho_{\text{syn}}^{\text{PACF}}\big\|_2\Big), \quad (3)$$

with vectors taken over lags $\ell = 1, \ldots, L$. A generator, therefore, receives high temporal deviation when it matches marginal IAT statistics but fails to reproduce burstiness or short-range dependence patterns.

Normalizing Distances: Different traffic classes and datasets exhibit inherently different levels of variability: certain flows (e.g., heavy exfiltration) naturally induce higher multivariate or temporal divergence than others, even when comparing disjoint subsets of the same dataset. To ensure comparability across scenarios and to avoid penalizing generators for variability intrinsic to the underlying traffic, we normalize all distance measures using class-specific reference variability estimates:

$$g(d) = \text{clip}\left(\frac{d_{R/S} - d}{d_{R/S} - d_{ref}}, 0, 1\right), \quad (4)$$

where d_{ref} denotes the divergence observed between reference partitions, representing the *best achievable fidelity* under dataset-intrinsic variability, and $d_{R/S}$ is a scenario-specific upper reference bound chosen to reflect the largest divergences observed across reference and stress partitions. This normalization is applied to d_k^{uni}, d_{MMD}, and d_{temp}, yielding realism scores that are interpretable and comparable across generators, classes, and scenarios.

C. Axis 3: Semantic & Behavioral Realism

Many DNS abuse behaviors manifest at the lexical and flow-structure levels—patterns that cannot be captured by protocol checks or marginal statistical divergences alone [3].

a) Fréchet Traffic Distance (FTD): To quantify semantic realism, DSEF adopts FTD [21], [38], conceptually analogous to the Fréchet Inception Distance used in generative vision models. Let E_r and E_s denote the embedding sets

extracted from an AutoEncoder (AE) trained exclusively on real flows. AE-based representations provide a compact semantic manifold that captures co-occurring lexical, structural, and temporal patterns [6], [30], and they have been widely used as unsupervised feature extractors for network traffic and security telemetry. FTD measures the discrepancy between the Gaussian approximations of these embedding distributions:

$$\mathrm{FTD}(E_r, E_s) = \|\mu_r - \mu_s\|_2^2 + \mathrm{Tr}\Big(\Sigma_r + \Sigma_s - 2(\Sigma_r^{1/2}\Sigma_s\Sigma_r^{1/2})^{1/2}\Big). \tag{5}$$

A low FTD indicates that synthetic flows align with real DNS activity in the learned semantic manifold, capturing high-level patterns (e.g., phishing-style lexical templates, structured payload-encoding behavior in exfiltration traffic, or the algorithmic regularities characteristic of DGA-generated domains).

Unlike Axis 1 and Axis 2, which evaluate structural validity and statistical alignment, Axis 3 focuses on the *behavioral meaning* encoded in DNS queries and their broader flow context. FTD serves as the sole semantic metric used in the composite realism score, providing a stable and interpretable measure of semantic alignment across heterogeneous generator families. For content-driven categories (e.g., phishing, exfiltration), LLM-based generators typically achieve lower FTD values and stronger embedding-space alignment, producing flows that cluster near real malicious activity in the learned semantic space.

D. Axis 4: Downstream Utility

Realism is necessary but insufficient: synthetic DNS traffic should also be useful for downstream tasks such as training, robustness evaluation, and red teaming [37]. Axis 4 therefore evaluates whether synthetic flows can support detection and classification pipelines in a manner consistent with real-world training or deployment settings. In principle, downstream utility can be assessed using any defender model. DSEF supports a range of evaluation settings, including rule-based systems, traditional ML classifiers, and LLM-based detectors. Utility is measured using standard metrics such as ROC-AUC or changes in TPR at low FPR levels; however, a comprehensive defender comparison is left to future work.

Remarks on Diversity (Future Extension): While the core DSEF score relies solely on utility-based metrics, the framework can optionally incorporate diversity indicators (e.g., normalized entropy, Simpson index) to diagnose mode collapse or limited coverage in synthetic traffic generators. Such indicators provide useful diagnostic signals; however, their interpretation is inherently scenario dependent. Accordingly, DSEF treats them as optional extensions.

E. Composite Score

For a given scenario c (e.g., benign, malware, phishing, spam, or exfiltration), each axis produces a scenario-specific subscore:

$$\bar{s}_i^{(c)} = \sum_{m \in M_i} \alpha_{im}\, g\big(m^{(c)}\big), \qquad \sum_m \alpha_{im} = 1, \tag{6}$$

where M_i is the set of metrics on axis i and α_{im} controls the internal weighting of metrics within that axis (fixed across generators). The overall DSEF realism score for generator G on scenario c is then:

$$S_{\mathrm{DSEF}}(G, c) = \sum_{i=1}^{4} w_i^{(c)}\, \bar{s}_i^{(c)}(G), \qquad \sum_i w_i^{(c)} = 1, \tag{7}$$

where $\mathbf{w}^{(c)}$ encodes the relative importance of the four axes for scenario c and is shared across all generators. We emphasize that DSEF does not rely on a fixed choice of $\mathbf{w}^{(c)}$; weights can be adjusted to reflect analyst priorities or operational constraints while preserving the same axis level validation pipeline and metrics.

V. Evaluation

We evaluate DSEF on content-driven DNS abuse using the four-axis methodology introduced in Section IV. Our goals are twofold: (i) to demonstrate that the DSEF validation pipeline can be instantiated end-to-end on a benchmark DNS dataset and yields stable, interpretable scores, and (ii) to show that the four axes expose meaningful differences between synthetic generators with varying levels of realism. To this end, we evaluate three representative *generator types*: a Tool-style replay generator (class-wise resampling of real flows), an IMS (Independent Marginal Sampler) generator that preserves per-feature marginals but destroys joint structure, and a lightweight ML-AE (ML-based latent sampling) generator that samples from an AutoEncoder-derived latent space. These generators span a spectrum from highly realistic (Tool) to deliberately structure-free (IMS) to moderately coherent but semantically weak (ML-AE). We then analyze axis-wise realism across the five activity labels present in the dataset.

A. Experimental Setup

Dataset: We use the BCCC–CIC–Bell–DNS–2024 dataset, which contains benign, malware, phishing, spam, and content-driven exfiltration scenarios. Each record exposes more than 120 DNS statistical, lexical, and timing features in tabular CSV format. We instantiate DSEF on the flow-level schema described in Section III and restrict exfiltration to the "heavy" variants of six payload types. Table II reports the resulting class counts.

Environment: All experiments are conducted in a containerized Python environment with fixed random seeds (42–46). For each run, we log the generator type, configuration, and seed identifiers in JSON files alongside the per-axis metrics, enabling full replay and inspection of the evaluation settings.

TABLE II: Dataset class counts and exfiltration variants (BCCC–CIC–Bell–DNS–2024).

Traffic class		Exfiltration file type	
Benign	3,764,933	Text	55,514
Malware	81,698	Image	56,702
Spam	30,371	Audio	52,544
Phishing	43,348	Video	58,846
Exfiltration	322,127	Compressed	45,136
		Exe	53,385

TABLE III: Synthetic generators used in DSEF and their evaluation roles.

Generator	Mechanism	Evaluation Role
Tool (Replay)	Resamples real flows (class-wise), preserving the original joint structure and semantics.	Upper-bound realism; verifies that DSEF returns near-perfect scores for near-real data.
IMS	Independently resamples each feature from empirical marginals; thereby destroying cross feature dependencies and semantics.	Tests DSEF's ability to distinguish marginal alignment from joint/semantic fidelity (Axis 2 vs. 3 separation).
ML-AE	Samples from an AE-derived latent space using class-conditional Gaussians; QNAMEs are generated by a simple Markov chain.	Represents intermediate-quality synthesis; reveals subtle distributional and semantic drift that simpler baselines may miss.

Stratified splits: Because our focus is to validate the DSEF validation layer, we adopt a *stratified* split. We partition the reference dataset into the train (70%), validation (15%), and test (15%) splits using stratification over the labels, ensuring that all classes are represented in each split and that axis-wise scores remain stable across seeds. Representation learning components (e.g., Axis 3 AutoEncoder) are trained only on the training split.

Synthetic generators: We evaluate three synthetic generators that emit flows in the same tabular schema described above, chosen to probe the effectiveness of DSEF's validation design and to expose distinct failure modes across the four axes. Details of the generators are summarized in Table III.

B. Generator Comparison: Tool, IMS, and ML-AE

We now ask whether DSEF can distinguish between generators with different realism regimes. To this end, we compare axis-wise scores for the Tool, IMS, and ML-AE generators under identical splits, seeds, and normalization baselines, and report per-class scores averaged over five seeds. Table IV highlights several coherent results.

Axis 2 (Distributional realism): For benign, malware, and phishing traffic, Tool and IMS both achieve high distributional realism ($\bar{s}_2 \approx 0.96$–0.99), indicating that matching per-feature marginals is often sufficient to preserve global numeric statistics in these scenarios. Exfiltration flows are more sensitive: Tool's class-wise replay yields a moderate average score ($\bar{s}_2 \approx 0.61$) with substantial seed-to-seed variability, while IMS reaches higher but still imperfect realism ($\bar{s}_2 \approx 0.79$).

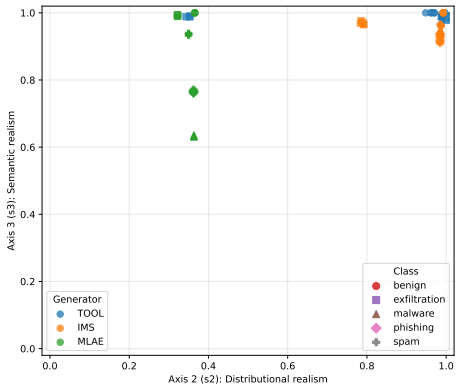

Fig. 2: Axis 2 vs. Axis 3 realism plane. Each point corresponds to a (generator, class, seed) run. The plane separates distinct failure regimes: Tool clusters near high distributional and semantic realism, IMS preserves marginal statistics but exhibits semantic degradation, and ML-AE shows strong distributional drift with class-dependent semantic collapse.

TABLE IV: Axis-wise realism scores per backbone and class (mean over five seeds).

Backbone	Class	$\bar{s}_2$	$\bar{s}_3$	H(QNAME)	u
Tool	benign	0.96	1.00	6.88	0.99
Tool	exfiltration	0.61	0.99	5.11	0.69
Tool	malware	0.99	0.99	6.88	0.98
Tool	phishing	0.99	0.99	6.55	0.83
Tool	spam	1.00	1.00	6.20	0.57
IMS	benign	0.99	1.00	6.88	0.99
IMS	exfiltration	0.79	0.97	5.11	0.69
IMS	malware	0.99	0.94	6.88	0.98
IMS	phishing	0.98	0.93	6.55	0.83
IMS	spam	0.99	0.96	6.20	0.57
ML-AE	benign	0.37	1.00	6.91	1.00
ML-AE	exfiltration	0.32	0.99	6.91	1.00
ML-AE	malware	0.36	0.64	6.91	1.00
ML-AE	phishing	0.36	0.77	6.91	1.00
ML-AE	spam	0.35	0.93	6.91	1.00

In contrast, ML-AE exhibits substantially lower Axis 2 scores across all classes ($\bar{s}_2 \approx 0.32$–0.37), revealing pronounced distributional drift induced by latent space sampling. Overall, Axis 2 highlights that marginal alignment alone can yield high realism scores, while more expressive generators may significantly distort global flow statistics.

Axis 3 (Semantic realism): Tool achieves near-perfect semantic realism for all classes ($\bar{s}_3 \approx 0.99$–1.00), as expected from class-wise replay. IMS also remains semantically close to the real manifold for benign and exfiltration flows, but its Axis 3 scores for malware and phishing slightly drop to $\bar{s}_3 \approx 0.94$ and 0.93, respectively. ML-AE displays a more nuanced pattern: benign and exfiltration flows remain near-perfect in Axis 3 ($\bar{s}_3 \approx 0.99$–1.00), whereas malware and phishing degrade substantially ($\bar{s}_3 \approx 0.64$ and 0.77). This suggests that the AE manifold for malicious and phishing traffic is more brittle: small latent perturbations already move samples away from the real embedding distribution. Spam shows a similar pattern: Tool and IMS remain close to the

reference ($\bar{s}_3 \gtrsim 0.96$), while ML-AE is marginally lower ($\bar{s}_3 \approx 0.93$), though the difference is small given the limited coverage of the class. Figure 2 visualizes generator behavior in the joint Axis 2–Axis 3 space.

Axis 4 (Utility & diversity): Because IMS and Tool reuse real QNAMEs from the corresponding class, their lexical entropy and unique domain ratios are nearly identical to the reference dataset (e.g., benign $H \approx 6.88$, $u \approx 0.99$; spam $H \approx 6.20$, $u \approx 0.57$). In contrast, ML-AE's Markov chain QNAME generator produces almost maximally diverse domains ($H \approx 6.91$, $u = 1.00$ for all classes): every synthetic flow effectively receives a distinct domain name drawn from a high entropy character model. This diversity is largely orthogonal to the other axes of realism.

Taken together, the results show that DSEF consistently assigns high realism scores to Tool replay, markedly lower scores to the deliberately unrealistic IMS generator along the semantic axis, and intermediate, regime-dependent scores to the ML-AE generator. Axes 2 and 3 are complementary: Axis 2 detects global distributional drift, while Axis 3 is sensitive to the breakdown of joint structure and higher-order semantics. Overall, these experiments demonstrate that our methodology is both stable under replay and sensitive to qualitatively different failure modes in synthetic DNS traffic, supporting DSEF as a benchmark-ready evaluation harness for synthetic traffic generators.

VI. Discussion

A. Implications for CTI and Security Operations

DSEF provides a practical and reproducible foundation for benchmarking synthetic DNS data in CTI workflows. From the operational perspective, modern SOCs increasingly rely on synthetic traffic for analyst training, proof of concept evaluations, and red-team exercises. However, unvalidated synthetic traffic datasets can mislead triage workflows. DSEF can directly address this risk by quantifying traffic realism along protocol, statistical, semantic, and utility dimensions, enabling SOC engineers to select or tune generators that align with concrete operational objectives. Analysts can therefore select generators whose DSEF profiles align with task-specific CTI objectives—for instance, prioritizing semantic fidelity for scenario modeling or distributional realism for detector benchmarking. In this sense, DSEF supports safe and effective synthetic data driven testing for security operations.

B. Ethics, Safety, and Reproducibility Considerations

Synthetic traffic used in security research must be handled responsibly to avoid enabling offensive misuse. Any generator intended for use within DSEF should avoid producing resolvable domains, operationally harmful artifacts, or payload-bearing content. Public manifests record hyperparameters and configuration details, enabling third parties to audit, replay, and extend evaluations while maintaining a transparent and trustworthy analysis pipeline.

VII. Conclusion

This work introduces **DSEF**, a framework that unifies DNS synthetic traffic evaluation under a reproducible, benchmark-oriented design. DSEF scores synthetic flows along four complementary axes—protocol correctness, distributional fidelity, semantic and behavioral realism, and downstream defensive utility—each normalized using reference-based baselines to provide interpretable measures of realism. Our preliminary study with three different generators shows that DSEF can distinguish qualitatively different generator regimes. Tool replay, as expected, achieves near-real scores along distributional and semantic axes; IMS preserves marginal feature statistics but degrades joint structure and semantic coherence; and ML-AE occupies an intermediate region, maintaining coarse benign structure while drifting for malicious classes. These findings validate DSEF as a practical, standardized lens for assessing the reliability and utility of synthetic DNS traffic.

Future releases of DSEF will expand both the breadth of evaluated generators and the dimensions of realism captured by the framework. We plan to evaluate additional generator families—including more advanced LLM-based and hybrid models—and to extend the assessment to a wider set of DNS attack types (e.g., reflection/amplification attacks, DNSSEC-related behaviors). In addition, we aim to deepen Axis 4 by conducting utility- and diversity-oriented studies. This includes reproducibility experiments in which real detection or mitigation systems are evaluated under controlled conditions using synthetic traffic generated by DSEF. Beyond benchmarking generators, DSEF highlights a broader operational risk: synthetic DNS traffic that is not properly validated can distort detector behavior and analyst decision making. By making realism and utility explicit and measurable, DSEF enables security teams to use synthetic traffic more safely and deliberately in training, evaluation, and testing workflows.

Acknowledgments

We thank Roland van Rijswijk-Deij for his constructive feedback on an earlier version of this manuscript. We also thank the anonymous reviewers for their helpful comments.

References

[1] Domain names - implementation and specification. RFC 1035, November 1987.

[2] Dure Adan Ammara, Jianguo Ding, and Kurt Tutschku. Synthetic network traffic data generation: A comparative study. *arXiv preprint arXiv:2410.16326*, 2024.

[3] L Torrealba Aravena, Pedro Casas, Javier Bustos-Jiménez, Germán Capdehourat, and Mislav Findrik. Dom2Vec-detecting DGA domains through word embeddings and AI/ML-Driven lexicographic analysis. In *2023 19th International Conference on Network and Service Management (CNSM)*, pages 1–5. IEEE, 2023.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[5] Ray Bellis and ISC. DNSGEN: DNS packet generator. https://github.com/isc-projects/dnsgen, 2025. Accessed: 2025-12-08.

[6] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[7] Alain Berlinet and Christine Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.

[8] Peter J. Brockwell and Richard A. Davis. *Time Series: Theory and Methods*. Springer, 2016.

[9] Adriel 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)*, pages 0728–0734. IEEE, 2019.

[10] Isaac Corley, Jonathan Lwowski, and Justin Hoffman. DomainGAN: generating adversarial examples to attack domain generation algorithm classifiers. *arXiv preprint arXiv:1911.06285*, 2019.

[11] Tianyu Cui, Xinjie Lin, Sijia Li, Miao Chen, Qilei Yin, Qi Li, and Ke Xu. TrafficLLM: Enhancing Large Language Models for Network Traffic Analysis with Generic Traffic Representation, 2025.

[12] Joao Luis Silva Damas, Michael Graff, and Paul A. Vixie. Extension Mechanisms for DNS (EDNS(0)). RFC 6891, April 2013.

[13] Javier Aday Delgado-Soto, Jorge E López de Vergara, Iván González, Daniel Perdices, and Luis de Pedro. Gpt on the wire: Towards realistic network traffic conversations generated with large language models. *Computer Networks*, page 111308, 2025.

[14] DNS-OARC. dnsperf: A dns performance and load testing tool. https://github.com/DNS-OARC/dnsperf. Accessed: 2025-12-08.

[15] Paul Emmerich, Sebastian Gallenmüller, Daniel Raumer, Florian Wohlfart, and Georg Carle. Moongen: A scriptable high-speed packet generator. In *Proceedings of the 2015 Internet Measurement Conference*, IMC '15, page 275–287, New York, NY, USA, 2015. Association for Computing Machinery.

[16] Abdulrahman Fahim, Shitong Zhu, Zhiyun Qian, Chengyu Song, Evangelos Papalexakis, Supriyo Chakraborty, Kevin Chan, Paul Yu, Trent Jaeger, and Srikanth V Krishnamurthy. Dns exfiltration guided by generative adversarial networks. In *2024 IEEE 9th European Symposium on Security and Privacy (EuroS&P)*, pages 580–599. IEEE, 2024.

[17] Bent Fuglede and Flemming Topsoe. Jensen-shannon divergence and hilbert space embedding. In *International symposium onInformation theory, 2004. ISIT 2004. Proceedings.*, page 31. IEEE, 2004.

[18] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. 13(null), 2012.

[19] Jorge Luis Guerra, Carlos Catania, and Eduardo Veas. Datasets are not enough: Challenges in labeling network traffic. *Computers Security*, 120:102810, 2022.

[20] Hanoch Haim. Trex: Realistic traffic generator. https://trex-tgn.cisco.com/, 2017. Accessed: 2025-10-25.

[21] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[22] iagox86. dnscat2 — dns tunnelling tool. https://github.com/iagox86/dnscat2, 2025. Accessed: 2025-12-08.

[23] Internet Engineering Task Force. Domain Name System (DNS). https://www.ietf.org/technologies/dns/. Accessed: 2025-12-17.

[24] Xi Jiang, Shinan Liu, Aaron Gember-Jacobson, Arjun Nitin Bhagoji, Paul Schmitt, Francesco Bronzino, and Nick Feamster. Netdiffusion: Network data augmentation through protocol-constrained traffic generation. *Proc. ACM Meas. Anal. Comput. Syst.*, 8(1), February 2024.

[25] Danial Khosh Kholgh and Panos Kostakos. PAC-GPT: A novel approach to generating synthetic network traffic with GPT-3. *IEEE Access*, 11:114936–114951, 2023.

[26] Dong-Wook Kim, Gun-Yoon Sin, Kwangsoo Kim, Jaesik Kang, Sun-Young Im, and Myung-Mook Han. Network traffic synthesis and simulation framework for cybersecurity exercise systems. *Computers, Materials and Continua*, 80(3):3637–3653, 2024.

[27] Jan Kohout and Tomáš Pevnỳ. Network traffic fingerprinting based on approximated kernel two-sample test. *IEEE Transactions on Information Forensics and Security*, 13(3):788–801, 2017.

[28] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 2002.

[29] Xuying Meng, Chungang Lin, Yequan Wang, and Yujun Zhang. NetGPT: Generative pretrained transformer for network traffic. *arXiv preprint arXiv:2304.09513*, 2023.

[30] Umberto Michelucci. An introduction to autoencoders. *arXiv preprint arXiv:2201.03898*, 2022.

[31] Victor M Panaretos and Yoav Zemel. Statistical aspects of wasserstein distances. *Annual review of statistics and its application*, 6(1):405–431, 2019.

[32] Yimo Ren, Hong Li, Peipei Liu, Jie Liu, Hongsong Zhu, and Limin Sun. CL-GAN: A GAN-based continual learning model for generating and detecting AGDs. *Computers & Security*, 131:103317, 2023.

[33] Maria L Rizzo and Gábor J Székely. Energy distance. *wiley interdisciplinary reviews: Computational statistics*, 8(1):27–38, 2016.

[34] Reza Sharifnya and Mahdi Abadi. Dfbotkiller: Domain-flux botnet detection based on the history of group activities and failures in dns traffic. *Digital Investigation*, 12:15–26, 2015.

[35] Nirhoshan Sivaroopan, Kaushitha Silva, Chamara Madarasingha, Thilini Dahanayaka, Guillaume Jourjon, Anura Jayasumana, and Kanchana Thilakarathna. A comprehensive survey on network traffic synthesis: From statistical models to deep learning. *arXiv preprint arXiv:2507.01976*, 2025.

[36] Cédric Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2008.

[37] Maximilian Wolf, Julian Tritscher, Dieter Landes, Andreas Hotho, and Daniel Schloer. Benchmarking of synthetic network data: Reviewing challenges and approaches. *Computers & Security*, 145:103993, 2024.

[38] Chao-Lun Wu, Yu-Ying Chen, Po-Yu Chou, and Chih-Yu Wang. Synthetic traffic generation with wasserstein generative adversarial networks. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pages 1503–1508. IEEE, 2022.

[39] Shengzhe Xu, Manish Marwah, Martin Arlitt, and Naren Ramakrishnan. STAN: Synthetic network traffic generation with generative neural models. In *International Workshop on Deployable Machine Learning for Security Defense*, pages 3–29. Springer, 2021.

[40] yarrick. dnscat2 — dns tunnelling tool. https://github.com/yarrick/iodine, 2025. Accessed: 2025-12-08.