# TITAN DGA: Enhancing DGA Evasiveness through a Transformer-based Autoencoder and Adversarial Self-Augmentation

Rafael C. Pregardier[1], Luiz A. C. Bianchi Jr.[2], Vinicius F. Garcia[3], Burkhard Stiller[4],
Luis A. L. Silva[5], Carlos R. P. dos Santos[6]
*Federal University of Santa Maria – UFSM, Santa Maria, RS, Brazil*
[1]rcpregardier@inf.ufsm.br, [2]luiz.bianchi@inf.ufsm.br, [5]luisalvaro@inf.ufsm.br, [6]csantos@inf.ufsm.br
*Federal University of Paraná – UFPR, Curitiba, PR, Brazil*
[3]viniciusfulber@ufpr.br
*University of Zurich – UZH, Zurich, Switzerland*
[4]stiller@ifi.uzh.ch

***Abstract*—Existing adversarial Domain Generation Algorithms (DGAs) often fail to produce realistic synthetic domains because their sequential models cannot adequately capture complex character dependencies within domain names. This paper presents TITAN DGA, a new approach that addresses this limitation by integrating a Generative Adversarial Network (GAN) with a Transformer-based autoencoder, enabling the model to effectively learn long-range dependencies to generate synthetic domains. The solution incorporates adversarial self-augmentation to enhance evasiveness through iterative retraining. The model learns from its own generated samples using two strategies: a broad approach that improves overall generation quality, and a targeted approach that specifically leverages previously successful evasions to refine the model's adversarial capabilities. Comprehensive evaluation against multiple state-of-the-art classifiers and competing adversarial DGAs demonstrates that TITAN DGA achieves superior evasion rates while maintaining realism.**

***Index Terms*—Domain Generation Algorithm, Generative Adversarial Network, Transformers, Adversarial Evasion, Self-Augmentation, Synthetic Data Generation**

## I. Introduction

Domain Generation Algorithms (DGAs) are an important element of cybersecurity, widely employed by sophisticated malwares to evade detection. By generating a large and frequently changing list of domain names, DGAs hinder threat identification and mitigation, thus enabling botnets to maintain communication with their Command and Control (C&C) servers [1]. This technique introduces significant challenges, particularly in critical environments like datacenter networks, cloud environments, and operational technology (OT) systems. The increase in the frequency of Distributed Denial-of-Service (DDoS) attacks, data exfiltration attempts, and ransomware - carried out by malware such as GameOver Zeus, Mirai, and CryptoLocker - highlights the need for more effective detection and response mechanisms.

A major challenge in detecting DGAs is their rapidly evolving nature, which renders traditional signature-based methods like blacklisting ineffective. Attackers continually enhance their algorithms, integrating sophisticated evasion techniques and name variation approaches that make malicious domains appear indistinguishable from legitimate traffic. This evolution has driven security teams to adopt machine learning-based detectors [2], creating a dual arms race [3]. As these detectors have improved, adversarial DGAs have become increasingly common. Modern approaches employ models like Generative Adversarial Networks (GANs) to generate realistic domain names specifically designed to evade even the most advanced classifiers, which struggle to identify complex and subtle character patterns in domain strings.

GANs generate realistic data through a competitive process between two neural networks: a generator, which creates samples, and a discriminator, which distinguishes them from real ones. In networking, they have been applied to generate complete packets [4], as well as metadata and traffic flows for simulations [5]. They have also proven effective for adversarial DGAs. Early models such as DeepDGA [6] used autoencoders to compress domains and generators to replicate their features. Later, DomainGAN employed a Wasserstein GAN with gradient penalty to produce more diverse and realistic domains that better evade detection [7]. More recently, Zhai et al. proposed CDGA [8], a controllable GAN-based framework that enables fine-grained manipulation of domain attributes for improved stealth against modern classifiers [8].

However, despite progress, many GAN-based DGAs still rely on Long Short-Term Memory (LSTM) models that generate domains one character at a time. While effective for local n-gram patterns, this approach limits the learning of global dependencies required for realistic structures. Longer domains further suffer from context loss due to vanishing gradients in RNNs [9]. As a result, these DGAs may capture low-level statistics but fail to preserve higher-order coherence, leaving artifacts exploitable by advanced detection systems [10]. Moreover, their performance is often tightly coupled to the initial training dataset, reducing adaptability against dynamically shifting adversarial strategies [3]. To address these challenges, recent work has explored representation techniques beyond raw character sequences, as effective domain representations are crucial for both generation and detection.

978-3-903176-75-1 

It is essential for the cybersecurity community to continuously analyze modern DGA techniques in order to understand and anticipate adversarial strategies that exploit weaknesses in existing detection systems. Systematic analysis of these advanced DGAs reveals emerging patterns and identifies gaps in current defense mechanisms, ultimately guiding the design of more robust and adaptive detectors. In this context, this work presents an architecture based on GANs integrated with Transformer-based architectures to generate realistic malicious domains, thereby promoting research and development of more sophisticated classifiers. Our approach leverages machine learning methods to model complex semantic and structural relationships in text sequences, enabling the generation of domains that exhibit high evasiveness against various detectors. Specifically, the main contributions of this research are:

- **Novel Architecture.** *TITAN DGA*, a new approach for generating malicious domains that combines a GAN with a Transformer-based autoencoder. This approach replaces traditional LSTM encoders and uses Kullback–Leibler divergence for stabilization, promoting greater diversity in the generated samples;
- **Enhanced Semantic and Structural Modeling.** By employing SentencePiece tokenization together with a Transformer architecture, the model effectively captures both short- and long-range dependencies between tokens. This results in the generation of domains that are more difficult to distinguish from real ones;
- **Progressive Evasion Enhancement through Self-Augmentation.** One training strategy is investigated to progressively improve evasiveness: *Targeted Self-Augmentation*, which retrains the model on a dynamic mixture of real domains and synthetic samples generated in previous stages, but selects synthetic samples exclusively from those misclassified as benign by external classifiers.

This work is organized as follows: Section II provides the necessary background and reviews related work, identifying key challenges in current DGA generation, such as the limitations of sequential models. Section III details TITAN DGA, which integrates a transformer-based architecture with a novel targeted self-augmentation training strategy, and discusses the challenges and considerations inherent in this approach. Section IV presents the experimental setup and evaluates the results of the method against established benchmarks. Finally, Section V offers concluding remarks and outlines potential directions for future work.

## II. Theoretical Review

This section covers the basics behind TITAN DGA: first, a review of the evolution of DGAs from simple seed-based methods to more complex wordlist and permutation schemes; next, a discussion of the generative frameworks that enable today's adversarial domain creation; and finally, an exploration of advanced training paradigms such as self-augmentation and show how TITAN DGA advances the current state of the art.

### A. Background

The Domain Name System (DNS) maps human-readable names to IP addresses and botnets exploit this flexibility to hide their C&C infrastructure. Early malware used hard-coded DNS records, but once discovered, defenders could easily blacklist these endpoints [11]. To address this weakness, DGAs emerged around 2008, enabling bots to compute daily lists of pseudo-random domains from shared seeds and contact whichever domains attackers registered [12].

Traditional DGAs fall into four categories: arithmetic-based, hash-based, wordlist-based, and permutation-based [13]. These early generators produced high-entropy domains, but their randomness made them detectable by machine learning classifiers using statistical distances or models like Random Forests [14], [15]. LSTM-based architectures further improved detection by modeling sequential dependencies in legitimate domains [16]. In response, adversarial DGAs began creating domains that mimic benign linguistic features using two main strategies: perturbation and generative modeling. Perturbation approaches like CharBot introduce minimal character substitutions into known good domains [17], while generative techniques like DeepDGA and WordDGA use GANs trained on legitimate domains to produce novel and evasive names [6], [18].

Given the complexity of these adversarial generation techniques and their potential to evade detection, robust evaluation methodologies are essential to assess both the quality of synthetic domains and the effectiveness of defense mechanisms. Evaluating synthetic data quality and effectiveness goes beyond simple statistical comparisons; it requires measuring machine learning model performance under four distinct training and testing protocols: **TRTR (Train on Real/Test on Real)**, which establishes a baseline by measuring model performance exclusively on genuine data; **TSTR (Train on Synthetic/Test on Real)**, which determines whether synthetic data can replace or augment real data; **TRTS (Train on Real/Test on Synthetic)**, which assesses how realistic and evasive the generated samples are; and **TSTS (Train on Synthetic/Test on Synthetic)**, which measures the internal consistency of the synthetic dataset itself, detecting issues like mode collapse or insufficient diversity [19], [20].

A further variant, **T(R+S) (Test on Real + Synthetic)**, compares a single model against a combined test set of real and synthetic samples, providing a more challenging assessment of its resilience across authentic and generated distributions [21], [22]. Finally, **TR+S (Train on Real + Synthetic)** examines mixed-data training: models are trained on a union of genuine and synthetic samples and then evaluated on a reserved set of real data [23]. This setting determines whether augmenting real datasets with synthetic DGAs enhances performance, robustness, and generalization beyond what real data alone can achieve [24], [25], a critical consideration for developing more effective and resilient DGA classifiers.

### B. Related Work

GANs have been successfully adapted to create realistic network-related data at both packet [4] and domain levels. In

DGAs, several GAN-based architectures produce malicious-looking yet diverse domain names. DeepDGA compresses real domains with an autoencoder and uses a GAN generator, but cannot capture long-range character dependencies [6]. Subsequent works like CDGA introduce ResNet blocks, tokenization and LSTM encoders, yet still train autoencoder and GAN separately [8]. TITAN DGA unifies both stages, employs transformers to model long-term token relationships, and incorporates a Kullback–Leibler divergence term to enrich its latent space. Alternative methods modify existing domains — DeceptionDGA adjusts vowel–consonant ratios [3], CharBot randomly swaps characters [17], and MaskDGA uses adversarial perturbations extracted from substitute classifiers [10] — but depend on access to detection models or on limited structural changes. In contrast, transformer-GAN hybrids like Style Transformer-GAN and TILGAN generate novel sequences by learning style and content representations or latent embeddings without parallel data, demonstrating strong fluency, diversity and control in text generation [26], [27].

As DGAs evolved, so did detection techniques, from blocklist matching to feature-based classifiers and deep models. Random Forest methods such as FANCI use up to 26 handcrafted linguistic features (entropy, n-gram statistics) to deliver real-time performance but lag behind deep solutions in recall and adaptability [15], [28]. LSTM-based detectors like LSTM.MI address class imbalance and achieve high F1-scores [29], while hybrid CNN–LSTM architectures (*e.g.*, BILBO) combine local and sequential feature learning to improve AUC and temporal robustness [30]. These advances underscore the strength of end-to-end deep learning for both generating and detecting adversarial domains, motivating the focus on GAN-driven DGA synthesis that maximizes diversity and evasion against heterogeneous classifiers.

### C. Discussion

In summary, the evolution of malicious domain generation has progressed from simple domain manipulations to sophisticated adversarial methods powered by GANs. Seminal efforts like DeepDGA [6] and CDGA [8] enhanced realism by coupling autoencoders with tokenization schemes, yet they remained limited by distinct encoding/generation stages and an inability to model long-range dependencies effectively [31]. By contrast, TITAN DGA, presented in the next section, tightly integrates its autoencoder and GAN components, leveraging a transformer architecture and the Kullback–Leibler divergence. This unified design produces coherent, highly evasive domains out-of-the-box, without the need for classifier-specific adjustments, thereby directly addressing the shortcomings of prior approaches.

## III. The TITAN DGA Pipeline

The work focuses on generating domain names with a GAN architecture built on transformers and augmented by a self-augmentation module. This design allows the model to capture both short- and long-range token dependencies while leveraging the variability injected by self-augmentation to improve generalization and robustness. Consequently, the model achieves higher evasion rates and produces domains more similar to real ones. The proposed TITAN DGA architecture — shown in Figure 1 — comprises five main stages:

1) **Datasets:** represents the initial dataset of legitimate domains used to pre-train the GAN, as well as the adversarial domains selected during TITAN DGA's retraining stage. Additionally, the datasets used for training the classification models are also processed during this step;
2) **Preprocessing:** in this stage, legitimate domains have their Top-Level Domains (TLDs) removed and are then tokenized with SentencePiece — a subword tokenizer that builds a compact, language-agnostic vocabulary and handles rare or unseen character sequences by segmenting them into statistically derived subword units — while also inserting special boundary tokens. Domains generated by TITAN DGA are also tokenized using a simpler, rule-based tokenizer. Finally, these processed legitimate and generated domains are merged into a single dataset, reducing overall vocabulary complexity and priming the data for the autoencoder;
3) **Transformer-based Autoencoder GAN:** this module combines a Transformer-based Autoencoder with a GAN [27] to learn compact latent representations of domain names. The encoder projects tokenized domains into continuous vectors that capture syntactic and semantic features, which serve as real samples during GAN training. The generator produces synthetic latent vectors while the discriminator distinguishes them from encoder-derived ones. The decoder reconstructs domain names from both real and generated vectors, ensuring consistency and syntactic validity. The system is trained end-to-end with a loss function that incorporates Kullback–Leibler divergence to regularize the latent space and enhance generation quality, producing realistic domains that retain the structure of legitimate examples;
4) **Post-processing:** after the decoder reconstructs the domain names, a validation filter ensures compliance with RFCs 1034 and 1035. Subsequently, a valid TLD from a predefined list is appended to each domain. This approach relieves the GAN of TLD generation responsibilities while guaranteeing the structural correctness of the final domains;
5) **Retraining:** in this final stage, the structurally valid domain names from the previous step undergo TLD removal and are evaluated by three state-of-the-art DGA classifiers: FANCI [28], LSTM.MI [29], and BILBO [30]. Only domains that are misclassified as legitimate by all three classifiers are retained, effectively filtering out easily detectable DGA samples. These remaining domains are then tokenized and incorporated into the GAN's training dataset for the next iteration.

### A. Datasets

Our research uses four datasets for different experimental purposes. The first dataset, "OB" (Only Benigns), contains 240,000 legitimate domains from the Tranco list [32] for initial

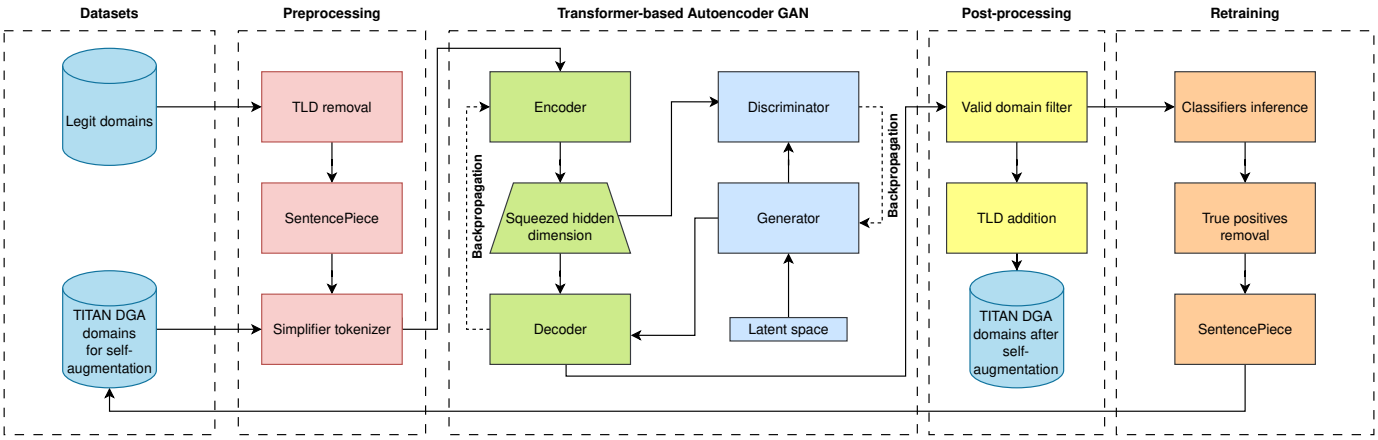


Fig. 1: TITAN DGA - General Architecture

GAN training and providing real domain samples for self-augmentation. The second dataset is a collection of ten datasets that systematically combine the OB dataset with increasing amounts of TITAN DGA-generated domains, with synthetic-to-real ratios from 0.10 to 1.00. The third dataset, "CT" (Classifiers Training), enables classifier training on real-world data by combining 500,000 legitimate domains from Tranco [32] with 500,000 DGA-generated domains from DGArchive [33]. The DGA samples include ten families with diverse generation methods: two hash-based (Bamital and Dyre), five arithmetic-based (Banjori, Conficker, Cryptolocker, Nymaim, and Pykspa), and three wordlist-based (Gozi, Matsnu, and Suppobox). Finally, dedicated evaluation datasets were constructed for each baseline DGA investigated in this study: CDGA [8], CharBot [17], Deception DGA [3], DeepDGA [6], and MaskDGA [10]. Each evaluation dataset contains 10,000 legitimate domains from Tranco [32] and 10,000 domains generated by the corresponding adversarial DGA.

### B. Preprocessing

The data preprocessing methodology varies based on whether self-augmentation techniques are applied. The preprocessing pipeline begins with the systematic removal of TLDs from all domains within the OB dataset. These TLDs, including standardized extensions such as `.com`, `.org`, and `.net`, represent the highest hierarchical level in domain name architecture and are derived from a finite, predefined set maintained by international standards organizations. Given their limited variability and adherence to predetermined structural conventions, TLDs are deliberately excluded from the generation process, as they do not contribute meaningful diversity to the GAN training procedure.

Following TLD removal, the processed OB domains are segmented using SentencePiece [34], Google's tokenization framework for neural text processing. This allows defining the vocabulary prior to training, ensuring consistent tokenization across the dataset. Among the available methods, the Unigram Language Model was chosen for its effectiveness in handling out-of-vocabulary terms and morphological variations common in domain names. This process decomposes domains into subunits, e.g., "facebook" becomes "face" and "_book", where the underscore marks a word boundary.

The preprocessing pipeline concludes with the application of a streamlined tokenizer that systematically maps the previously established tokens to corresponding numerical indices, facilitating neural network processing. This tokenization step incorporates essential special tokens, including start-of-sequence (`sos`) and end-of-sequence (`eos`) markers, which provide crucial structural information for sequence generation tasks. Importantly, this unified tokenization process is applied consistently to both the original OB domains and the synthetically generated domains produced during the retraining phase for self-augmentation, ensuring methodological consistency across all experimental conditions.

### C. Transformer-based Autoencoder GAN

The GAN architecture employed in this work consists of a generator, a discriminator, and a transformer-based autoencoder. In the autoencoder, both the encoder and the decoder are composed by two self-attention layers with 16 attention heads, followed by a feedforward layer with 512 units and an embedding dimension of size 512. Both the generator and discriminator are implemented as single-layer Perceptrons with 128 neurons. The LeakyReLU activation function is used to ensure training stability, and the loss function is based on the Kullback-Leibler (KL) divergence, as it encourages the generation of diverse samples that closely follow the distribution of real data. The generator maps noise samples from a standard normal distribution $\varepsilon \sim \mathcal{N}(0, I)$ through the MLP $g_\beta(\varepsilon)$, resulting in an implicit latent distribution $p_\beta(z)$. The training objective is to align this distribution with the aggregated posterior of the encoder, defined as $q_\phi(z) = \mathbb{E}_{x\sim p_r(x)}[q_\phi(z|x)]$, by minimizing the KL divergence:

$$\mathcal{L}_g(\phi, \beta) = D_{\mathrm{KL}}(q_\phi(z) \,\|\, p_\beta(z)) = \int q_\phi(z) \log\left(\frac{q_\phi(z)}{p_\beta(z)}\right) dz \tag{1}$$

This term is incorporated into the overall loss function with a weighting coefficient, encouraging the generator to capture

the full diversity present in the encoded distributions. Unlike traditional variational autoencoders (VAEs), which regularize each individual latent vector $q(z|x)$ to match a fixed prior (typically $\mathcal{N}(0, I)$), TITAN applies KL divergence at the level of the *aggregate latent distribution*. Specifically, it aligns the overall distribution of latent vectors produced by the generator with that of the encoder, promoting compatibility and diversity in the latent space without imposing a predefined prior. In addition, a second KL divergence term is introduced to enhance the decoder's robustness. This auxiliary term measures the discrepancy between the encoder's distribution over reconstructed samples $\tilde{z} = E(G(g(\varepsilon)))$ and the original latent distribution $q_\phi(z)$:

$$\mathcal{L}_{\text{dec}} = D_{\text{KL}}(q_\phi(z) \,\|\, \tilde{p}_g(z)) \tag{2}$$

This additional penalty ensures that the decoder learns to handle latent representations generated by the GAN, promoting higher fidelity in text reconstruction even when exposed to latent vectors not seen during training.

### D. GAN Training

For model training, the Adam optimizer was used across all components of the GAN architecture, and the Stochastic Gradient Descent (SGD) was used on the autoencoder component. However, different learning rates were assigned to each module to address their specific optimization requirements: the generator was trained with a learning rate of (0.0004), the discriminator with (0.0001), and the autoencoder with a higher rate of (0.06). The choice of these values was made based on an extensive empirical analysis of various hyperparameters, particularly for the autoencoder, which was trained using SGD and therefore required a higher learning rate compared to adaptive optimizers, in order to effectively learn stable representations of the domains.

The latent input to the generator consists of 100-dimensional vectors sampled from a standard normal distribution $\mathcal{N}(0, I)$. A batch size of 256 was adopted for balancing training stability and efficient memory usage. Additionally, a 30% dropout was applied at various points in the autoencoder, including after the attention heads and the feedforward layer. Finally, controlled Gaussian noise was introduced during both training and inference within the autoencoder. This technique aims to improve the model's robustness by encouraging the learning of latent representations that remain stable under small input changes, thereby improving generalization performance. The training environment was based on Python 3.12.3, running on a workstation with an NVIDIA RTX 5070 Ti GPU, 128 GB of RAM, and an Intel Core i5-14600KF processor (3.50 GHz).

### E. Post-processing

The post-processing step involves two main tasks: validating domains generated by the GAN and assigning TLDs. Generated domains are filtered according to the specifications defined in RFCs 1034 and 1035, which establish the syntax rules and naming conventions for domain names in the DNS system. Beyond removing invalid domains, this validation process also prevents the generation of structurally similar variants, improving the overall syntactic quality of the output.

TLD assignment is performed using a fixed list that includes the 50 most common traditional TLDs registered globally, as well as the 50 most frequent new gTLDs, based on statistics provided by [35]. The term "new gTLDs" refers to an expansion introduced by ICANN starting in 2013, which includes a wide variety of more descriptive or brand-oriented domains such as `.shop`, `.tech`, and `.xyz`. This selection strategy ensures structural consistency with real-world distributions and increases the likelihood that the generated domains resemble those commonly observed on the public internet.

Unlike other adversarial DGA models in the literature, TITAN DGA performs a post-processing that enforces compliance with DNS syntax standards and incorporates realistic TLDs. As shown in Table I, domains generated by TITAN DGA exhibit greater lexical coherence and semantic plausibility compared to those produced by other models, many of which generate purely random or malformed strings. This qualitative advantage directly results from post-processing and filtering stages, which are more rigorous than those used in prior approaches.

### F. Retraining

The retraining stage represents the core component of the self-augmentation mechanism. Two training strategies were investigated to progressively improve evasiveness: (*i*) *Iterative Self-Augmentation*, which retrains the model on a dynamic mixture of real domains and synthetic samples generated in previous stages; (*ii*) *Targeted Self-Augmentation*, which uses the same approach but selects synthetic samples exclusively from those misclassified as benign by external classifiers.

In this work we opted for *Targeted Self-Augmentation*, as it provided a significant improvement in model performance compared to the iterative method, which introduced higher noise levels by incorporating data not classified as legitimate domains. Consequently, the filtered domains generated by the GAN — initially trained exclusively on legitimate domains — are evaluated by the FANCI [28], LSTM.MI [29], and BILBO [30] classifiers. Inference is carried out sequentially, with one classifier at a time, following an order determined by their performance on the CT dataset: starting with FANCI, followed by LSTM.MI, and concluding with BILBO.

After each round of inference, domains that are classified as true positives (*i.e.*, correctly identified as DGAs) are removed. Only those categorized as legitimate are passed to the next classifier in the sequence. At the end of this filtering process, a dataset containing 240,000 DGA domains that successfully evaded all three classifiers is created. This dataset serves as the foundation for constructing the various synthetic-to-real ratio datasets. Before being reintegrated into the training pipeline, however, the domains are tokenized using SentencePiece [34] to ensure consistent segmentation, aligned with the preprocessing applied to the OB dataset.

TABLE I: Examples of domain names generated by different Adversarial DGA models (TLDs removed)

| TITAN DGA | CDGA | MaskDGA | CharBot | Deception DGA | DeepDGA |
|---|---|---|---|---|---|
| quickupsports | riwoim | aahrdjjrwfh | provenfvnners | jtrogani–ebuesmeorean-we | firiaps |
| greenservice | presyouryouth | uoruiddvwdwuihkul | kricmbc | trtegogonsc | sirgivrv |
| sebeautyproducts | numberconditions | ggupitmcfz | bxzlerweb | likinkew | laner |
| fitnesslife360 | plsystemlegaledu | jxhwkhmsza | guirpad | yalowltiveraf | mivognit |
| timemarketbox | venproevo | wwypddbcoodl | eicelkran | liydchousyorkv | qiurdeees |

## IV. Experiments and Results

The conducted experiments aimed to assess both the performance and adaptability of the synthetic data produced by TITAN DGA through the targeted self-augmentation process. The generated domains were evaluated using well-established classifiers from the literature to measure how effectively they can evade detection. Additionally, their spatial distribution was analyzed using dimensionality reduction techniques to examine dispersion and overlap with legitimate domains, providing insights into structural similarity and evasion potential. A comprehensive dataset was constructed with sufficient representation from major DGA families to enable robust training of the classification models. The experimental procedure followed these steps:

1) **Training of classification models:** Three classification models were trained based on solutions from the literature [28]–[30]. Each classifier was trained individually using 80% of the CT dataset. This paper implementation of the FANCI model [28] used 16 features instead of the original 21 features due to TITAN DGA not generate TLDs. All classifiers were configured according to the original architectural and hyperparameter specifications described in their respective papers;
2) **Classifier inference:** After training completion, an inference was performed using domains generated by TITAN DGA, both before and after the self-augmentation process. This evaluation measured the evasion success of the generated domains by determining the extent to which they were misclassified or remained undetected by the classifiers;
3) **Comparison with other DGAs under adversarial retraining:** To evaluate the robustness of the TITAN DGA models against existing adversarial approaches, adversarial retraining were performed using synthetic domains generated by different DGA techniques. Five well-established adversarial DGAs were included from the literature: CDGA [8], CharBot [17], DeceptionDGA [3], DeepDGA [6], and MaskDGA [10]. For each adversarial DGA, the classifiers were retrained on a balanced dataset consisting of 50% real domains and 50% synthetic domains generated by that specific model. This experimental setup enabled a direct comparison of evasiveness across different adversarial techniques and revealed how the inclusion of adversarial domains impacts detection performance;
4) **Computation of complementary metrics:** Additional metrics were computed to assess the quality and distribution of generated domains. These included the Wasserstein distance between generated and real domains, and the structural distribution of data points in the vector space. To analyze the latter, two widely used dimensionality reduction techniques were applied: Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE).

The experiments were validated using well-established evaluation metrics. Classifier performance was assessed using five standard metrics: Precision, Accuracy, Recall, F1-Score, and AUC/ROC (Area Under the Curve), which measures the probability that a classifier ranks a randomly chosen positive instance higher than a randomly chosen negative one. All metrics range from 0 to 1, with higher values indicating better performance. Initially, classifiers were evaluated on the training dataset to validate their ability to distinguish between legitimate and generated domains.

These metrics were then used to evaluate TITAN DGA's evasion performance on datasets generated before and after the application of self-augmentation. For evasion assessment, lower metric values indicate better performance, as they reflect the classifiers' inability to detect the generated malicious domains. The performance of the DGA without self-augmentation was compared against the worst-case evasion scenario with self-augmentation, as well as against state-of-the-art DGAs under adversarial retraining. Additionally, PCA and t-SNE were employed to analyze domain distribution in the latent space and to illustrate the separation between legitimate and GAN-generated domains.

TABLE II: Performance of each classifier on CT dataset

| Classifier | Accuracy | Precision | Recall | F1-Score | AUC/ROC |
|---|---|---|---|---|---|
| FANCI | 0.8304 | 0.8394 | 0.8170 | 0.8281 | 0.8304 |
| LSTM.MI | 0.9460 | 0.9344 | **0.9594** | 0.9467 | 0.9879 |
| **BILBO** | **0.9498** | **0.9477** | 0.9521 | **0.9499** | **0.9895** |

As shown in Table II, all classifiers achieved solid performance on the CT dataset, although with varying degrees of effectiveness. Bilbo stood out with the best overall results, reaching an Accuracy of 0.9498 and an AUC/ROC of 0.9895, indicating excellent discrimination between benign and malicious domains. LSTM.MI followed closely, also presenting high performance (AUC/ROC of 0.9879), and balanced precision and recall as reflected in its F1-Score. FANCI, while outperformed by the deep learning-based models, still obtained competitive results, with an Accuracy of 0.8304 and Precision of 0.8394. These findings reinforce the advantage of neural networks in capturing complex DGA patterns, while also showing that traditional classifiers like FANCI remain effective under certain conditions.

TABLE III: Demonstration how different levels of self-augmentation affect classification performance. The synthetic-to-real ratio represents the proportion of synthetic samples added to the real dataset (*e.g.*, 0.10 = 10% additional synthetic data)

| Synthetic-to-Real Ratio | Classifier | Accuracy | Precision | Recall | F1-Score | AUC/ROC |
|---|---|---|---|---|---|---|
| 0.00 (baseline) | BILBO | 0.5055 | 0.5393 | 0.0754 | 0.1323 | 0.5217 |
| 0.10 | FANCI | 0.5206 | 0.5479 | 0.2364 | 0.3303 | 0.5206 |
| 0.20 | BILBO | 0.5099 | 0.5725 | 0.0778 | 0.1370 | 0.5354 |
| 0.30 | FANCI | 0.5201 | 0.5466 | 0.2352 | 0.3289 | 0.5201 |
| 0.40 | LSTM.MI | 0.5014 | 0.5111 | 0.0669 | 0.1183 | 0.5108 |
| **0.50** | **BILBO** | 0.5028 | 0.5226 | **0.0636** | **0.1134** | 0.5261 |
| 0.60 | FANCI | 0.4930 | 0.4814 | 0.1811 | 0.2632 | 0.4930 |
| **0.70** | **FANCI** | **0.4722** | **0.4167** | 0.1394 | 0.2089 | **0.4721** |
| 0.80 | BILBO | 0.5058 | 0.5454 | 0.0697 | 0.1236 | 0.5245 |
| 0.90 | FANCI | 0.4947 | 0.4860 | 0.1845 | 0.2675 | 0.4947 |
| 1.00 | FANCI | 0.5108 | 0.5262 | 0.2167 | 0.3070 | 0.5108 |

Table III shows that classifiers exhibited lower detection performance against domains generated by self-augmented GANs compared to the baseline (0.00 ratio), where GANs were trained solely on benign domains. In this step, classifiers were trained on the same CT dataset as in Table II, and the best AUC/ROC was reported for each synthetic-to-real ratio. For example, Bilbo's AUC/ROC dropped from 0.5217 in the baseline to 0.4930 with self-augmentation using FANCI at a 0.60 ratio. Although these decreases appear modest (5.5%), they represent meaningful gains in evasion capability given the baseline difficulties. FANCI consistently achieved the highest recall and F1-Score across ratios, indicating broader detection but at the cost of increased false positives. AUC/ROC was used as the main metric since it evaluates classifier discrimination independently of thresholds, enabling consistent comparison across classifiers and ratios while revealing subtle shifts in model confidence.

The motivation behind self-augmentation extends beyond simply increasing training data volume to enabling feedback-oriented refinement of the generator's latent space. By incorporating its own successful outputs back into training, the GAN iteratively explores regions of the latent space that produce harder-to-detect domains. This process creates a form of evasion-driven self-adaptation that would be difficult to achieve through conventional data augmentation with benign domains alone. Self-augmentation therefore acts as a mechanism for incremental discovery of evasive structures, guided by the classifier's own detection weaknesses rather than relying on manual data expansion.

Table IV summarizes the evasion performance of different adversarial DGA models against adversarially retrained classifiers. Bilbo was selected for this evaluation due to its consistently superior AUC/ROC performance on the CT dataset. Despite targeting a more robust classifier, TITAN DGA achieved more effective evasion than all the other adversarial DGAs from the literature. Even at its least evasive configuration (0.20 synthetic-to-real ratio), TITAN outperformed its baseline (0.00 ratio) in Recall and F1-Score, demonstrating improved evasion while maintaining realism. Compared to CDGA and CharBot, TITAN with self-augmentation at 0.20 achieved respectively up to 30.0% and 23.7% lower recall, and up to 18.8% and 17.6% lower F1-score. These reductions indicate stronger evasion performance, even in adversarial retraining scenarios, where most models tend to overfit to detection patterns.

To assess the similarity between domains generated by each adversarial DGA and legitimate domains, the Wasserstein distance was computed per model. A Word2Vec Skip-gram embedding of 256 dimensions was first trained on legitimate domains, and tokens from each DGA were mapped into this space. For each embedding dimension, the Wasserstein distance was calculated between legitimate and adversarial distributions, and the final value was obtained by averaging across dimensions. As shown in Table V, TITAN DGA achieved the lowest distance (0.00122), indicating that its synthetic domains are not only structurally coherent but also semantically aligned with legitimate domain patterns.

TABLE V: Wasserstein distance from legit dataset

| Adversarial DGA | Wasserstein Distance |
|---|---|
| **TITAN** | **0.00122** |
| CDGA | 0.00527 |
| DeepDGA | 0.04157 |
| DeceptionDGA | 0.00309 |
| MaskDGA | 0.00989 |
| CharBot | 0.00241 |

This result highlights TITAN's capability to generate domains that are highly evasive and difficult to distinguish from real ones. Such effectiveness is further supported by the recall analysis presented in Tables III and IV, where TITAN DGA — with self-augmentation — consistently achieved the lowest

TABLE IV: Performance of the classifier retrained with adversarial retraining in several DGA families

| DGA Family | Accuracy | Precision | Recall | F1-Score | AUC/ROC |
|---|---|---|---|---|---|
| **TITAN** | **0.6570** | **0.6721** | 0.6130 | 0.6412 | **0.7071** |
| **TITAN 0.20 Synthetic/Real ratio** | 0.6675 | 0.7099 | **0.5665** | **0.6301** | 0.7389 |
| CDGA | 0.7863 | 0.8137 | 0.7425 | 0.7765 | 0.8744 |
| DeepDGA | 0.9995 | 1.0000 | 0.9990 | 0.9995 | 0.9923 |
| DeceptionDGA | 0.7810 | 0.7461 | 0.8520 | 0.7955 | 0.8647 |
| MaskDGA | 0.9555 | 0.9691 | 0.9410 | 0.9548 | 0.9923 |
| CharBot | 0.7515 | 0.7256 | 0.8090 | 0.7650 | 0.8323 |

recall values among all evaluated models. This implies that the generated domains are more likely to bypass detection mechanisms, reinforcing the practical impact of TITAN's latent space modeling and post-processing strategies.

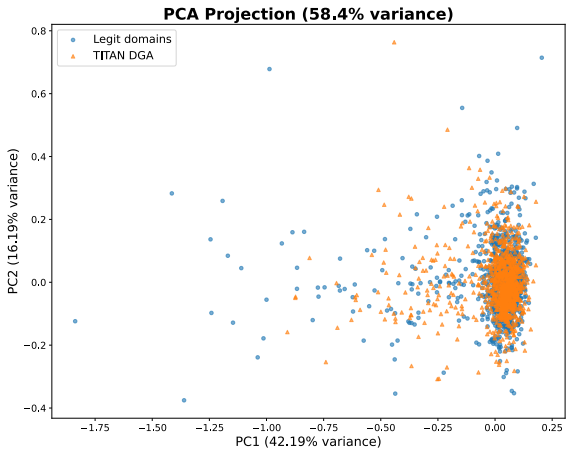


Fig. 2: PCA visualization of generated and real domains

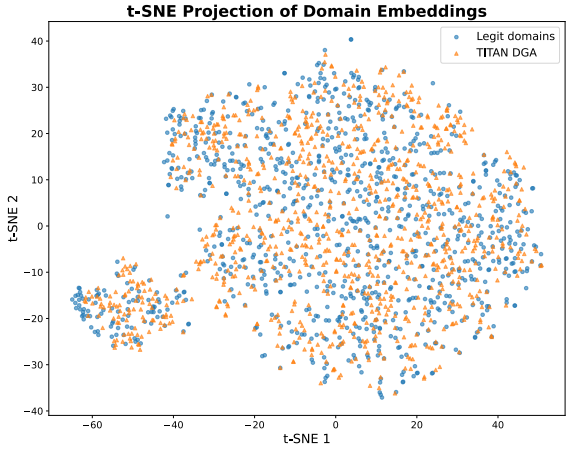


Fig. 3: t-SNE visualization of generated and real domains

To assess the similarity between the domains generated by TITAN DGA and legitimate domains, two dimensionality reduction techniques were employed: PCA and t-SNE. The PCA projection (Figure 2) shows that TITAN DGA domains overlap substantially with the real domains in the reduced space, indicating that their global distributional characteristics are closely aligned. In addition, t-SNE (Figure 3) reveals that TITAN-generated domains are densely embeded with legitimate ones, reflecting strong local structure similarity in the embedding space. Together, these projections confirm that TITAN DGA is capable of producing highly realistic domains that mimic both the overall and fine-grained distributional patterns of legitimate traffic, reinforcing its evasiveness and the challenge it poses for detection mechanisms.

## V. Final Remarks

This work introduces TITAN DGA, a novel architecture that combines Generative Adversarial Networks, a Transformer-based Autoencoder, and an adversarial self-augmentation strategy to generate highly evasive domain names that closely resemble legitimate ones. By integrating these three components, TITAN DGA addresses key limitations of prior adversarial DGA models, including their inability to model long-range token dependencies and adapt to evolving detection mechanisms. The self-augmentation process enables the model to iteratively refine its latent space based on its own evasive outputs, producing domains that are both harder to detect and more lexically and syntactically aligned with real-world traffic.

The evaluations demonstrate that TITAN DGA achieves superior evasion performance even under adversarial retraining scenarios, maintaining low detection rates compared to other adversarial DGAs, even when classifiers are trained on its generated domains. The self-augmentation mechanism shows that TITAN DGA benefits from synthetic sample inclusion, improving overall performance while remaining highly evasive against classifiers. The similarity between TITAN-generated domains and legitimate ones is demonstrated through lower Wasserstein distances, and analysis of both PCA and t-SNE. These results indicate that TITAN DGA domains follow a distribution closely aligned with legitimate domains, confirming the effectiveness of the proposed approach.

Although TITAN DGA was evaluated in a controlled experimental setup based solely on domain string analysis, its applicability extends to real-world cybersecurity pipelines. The model can be integrated into red-teaming environments to simulate highly evasive DGAs, supporting the training and stress-testing of detection systems. Security providers and enterprise SOCs may leverage TITAN-generated domains to improve classifier robustness by including them in adversarial retraining datasets, thus enhancing resilience against future DGA-based malware campaigns. Furthermore, TITAN DGA can serve as a benchmarking tool for evaluating the limits of detection models in realistic Internet-facing scenarios, complementing other synthetic traffic generation methods.

Future work will focus on enhancing the TITAN DGA architecture to address current limitations. First, while the current approach is restricted to domain string generation and does not incorporate contextual or behavioral information—such as DNS query timing, resolution patterns, or WHOIS metadata—future versions will explore the integration of these multi-modal signals to better simulate real-world network scenarios and improve the model's applicability in enterprise detection pipelines. Second, TITAN DGA's effectiveness depends on the quality and diversity of training data (legitimate domains and classifiers); ongoing research will investigate the use of additional input datasets, beyond Tranco, to train the GAN and generate more evasive domains, enhancing generalization across heterogeneous network environments. Third, an adaptive self-augmentation mechanism will be developed, capable of dynamically adjusting synthetic domain injection rates based on feedback metrics such as classifier uncertainty, allowing more effective refinement of evasive behaviors.

## REFERENCES


[1] K. Alieyan, A. ALmomani, A. Manasrah, and M. M. Kadhum, "A survey of botnet detection based on dns," *Neural Computing and Applications*, vol. 28, no. 7, pp. 1541–1558, Jul 2017.

[2] B. C. Cebere, J. L. B. Flueren, S. Sebastián, D. Plohmann, and C. Rossow, "Down to earth! guidelines for dga-based malware detection," in *Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses*, 2024, pp. 147–165.

[3] J. Spooren, D. Preuveneers, L. Desmet, P. Janssen, and W. Joosen, "Detection of algorithmically generated domain names used by botnets: a dual arms race." New York, NY, USA: Association for Computing Machinery, 2019, pp. 1916–1923.

[4] L. A. Bianchi, R. C. Pregardier, L. A. Silva, and C. R. dos Santos, "2pack-gan: Exploring transfer learning to fine-tune generative adversarial networks for network packet generation," in *NOMS 2025-2025 IEEE Network Operations and Management Symposium*. IEEE, 2025, pp. 1–9.

[5] 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.

[6] H. S. Anderson, J. Woodbridge, and B. Filar, "Deepdga: Adversarially-tuned domain generation and detection," in *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*. New York, NY, USA: Association for Computing Machinery, 2016, p. 13–21.

[7] I. Corley, J. Lwowski, and J. Hoffman, "Domaingan: generating adversarial examples to attack domain generation algorithm classifiers," *arXiv preprint arXiv:1911.06285*, 2019.

[8] Y. Zhai, J. Yang, Z. Wang, L. He, L. Yang, and Z. Li, "Cdga: A gan-based controllable domain generation algorithm," in *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2022, pp. 352–360.

[9] L. Nie, X. Shan, L. Zhao, and K. Li, "Pkdga: A partial knowledge-based domain generation algorithm for botnets," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 4854–4869, 2023.

[10] L. Sidi, A. Nadler, and A. Shabtai, "Maskdga: A black-box evasion technique against dga classifiers and adversarial defenses," *arXiv preprint arXiv:1902.08909*, 2019.

[11] C. J. Dietrich, C. Rossow, and N. Pohlmann, "Cocospot: Clustering and recognizing botnet command and control channels using traffic analysis," *Computer Networks*, vol. 57, no. 2, pp. 475–486, 2013.

[12] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon, "From Throw-Away traffic to bots: Detecting the rise of DGA-Based malware," in *21st USENIX Security Symposium (USENIX Security 12)*, Bellevue, WA, 2012, pp. 491–506.

[13] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, and E. Gerhards-Padilla, "A comprehensive measurement study of domain generating malware," in *Proceedings of the 25th USENIX Conference on Security Symposium*. USA: USENIX Association, 2016, pp. 263–278.

[14] S. Yadav, A. K. K. Reddy, A. N. Reddy, and S. Ranjan, "Detecting algorithmically generated domain-flux attacks with dns traffic analysis," *IEEE/Acm Transactions on Networking*, vol. 20, no. 5, pp. 1663–1677, 2012.

[15] R. Sivaguru, C. Choudhary, B. Yu, V. Tymchenko, A. Nascimento, and M. D. Cock, "An evaluation of dga classifiers," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 5058–5067.

[16] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Predicting domain generation algorithms with long short-term memory networks," *arXiv preprint arXiv:1611.00791*, 2016.

[17] J. Peck, C. Nie, R. Sivaguru, C. Grumer, F. Olumofin, B. Yu, A. Nascimento, and M. De Cock, "Charbot: A simple and effective method for evading dga classifiers," *IEEE Access*, vol. 7, pp. 91 759–91 771, 2019.

[18] S. Selvaraj and R. Panjanathan, "Worddga: Hybrid knowledge-based word-level domain names against dga classifiers and adversarial dgas," in *Informatics*, vol. 11, no. 4. MDPI AG, 2024, p. 92.

[19] G.-S. Năstăsescu and D.-C. Cercel, "Conditional wasserstein gan for energy load forecasting in large buildings," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–8.

[20] S. Vecile, K. Lacroix, K. Grolinger, and J. Samarabandu, "Malicious and benign url dataset generation using character-level lstm models," in *2022 IEEE Conference on Dependable and Secure Computing (DSC)*. IEEE, 2022, pp. 1–8.

[21] H. A. Ahmed, J. A. Nepomuceno, B. Vega-Márquez, and I. A. Nepomuceno-Chamorro, "Synthetic data generation for healthcare: Exploring generative adversarial networks variants for medical tabular data," *International Journal of Data Science and Analytics*, pp. 1–16, 2025.

[22] B. Camus, T. Voillemin, C. Le Barbu, J.-C. Louvigné, C. Belloni, and E. Vallée, "Training deep learning models with hybrid datasets for robust automatic target detection on real sar images," in *2024 International Radar Conference (RADAR)*. IEEE, 2024, pp. 1–6.

[23] G. Pasculli, M. Virgolin, P. Myles, A. Vidovszky, C. Fisher, E. Biasin, M. Mourby, F. Pappalardo, S. D'Amico, M. Torchia *et al.*, "Synthetic data in healthcare and drug development: Definitions, regulatory frameworks, issues," *CPT: Pharmacometrics & Systems Pharmacology*, vol. 14, no. 5, pp. 840–852, 2025.

[24] A. Arora and A. Arora, "Machine learning models trained on synthetic datasets of multiple sample sizes for the use of predicting blood pressure from clinical data in a national dataset," *PLoS One*, vol. 18, no. 3, p. e0283094, 2023.

[25] L. Eversberg and J. Lambrecht, "Combining synthetic images and deep active learning: Data-efficient training of an industrial object detection model," *Journal of Imaging*, vol. 10, no. 1, 2024.

[26] K.-H. Zeng, M. Shoeybi, and M.-Y. Liu, "Style example-guided text generation using generative adversarial transformers," *arXiv preprint arXiv:2003.00674*, 2020.

[27] S. Diao, X. Shen, K. Shum, Y. Song, and T. Zhang, "TILGAN: Transformer-based implicit latent GAN for diverse and coherent text generation," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Online: Association for Computational Linguistics, Aug. 2021, pp. 4844–4858.

[28] S. Schüppen, D. Teubert, P. Herrmann, and U. Meyer, "Fanci: feature-based automated nxdomain classification and intelligence," in *Proceedings of the 27th USENIX Conference on Security Symposium*. USA: USENIX Association, 2018, p. 1165–1181.

[29] D. Tran, H. Mac, V. Tong, H. A. Tran, and L. G. Nguyen, "A lstm based framework for handling multiclass imbalance in dga botnet detection," *Neurocomputing*, vol. 275, pp. 2401–2413, 2018.

[30] K. Highnam, D. Puzio, S. Luo, and N. R. Jennings, "Real-time detection of dictionary dga network traffic using deep learning," 2020. [Online]. Available: https://arxiv.org/abs/2003.12805

[31] X. Hu, H. Chen, M. Li, G. Cheng, R. Li, H. Wu, and Y. Yuan, "Replacedga: Bilstm-based adversarial dga with high anti-detection ability," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 4406–4421, 2023.

[32] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczynski, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," in *Proceedings 2019 Network and Distributed System Security Symposium*, ser. NDSS 2019. Internet Society, 2019.

[33] Fraunhofer FKIE, "Dgarchive: Database of domain generation algorithm domains," https://dgarchive.caad.fkie.fraunhofer.de/, 2020.

[34] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," 2018. [Online]. Available: https://arxiv.org/abs/1808.06226

[35] "Domain name stat," https://domainnamestat.com/.