TITAN DGA: Uma GAN Otimizada por Divergência KL com Autoencoder Baseado em Transformers para Geração de Domínios Maliciosos

Rafael C. Pregardier[1], Luiz A. C. Bianchi Jr.[1], Alfredo Cossetin Neto[1]
Vinicius Fulber-Garcia[2], Luis A. L. Silva[1], Carlos R. P. dos Santos[1]

[1] Departamento de Computação Aplicada (DCOM)
Universidade Federal de Santa Maria (UFSM)
Av. Roraima, 1000, Bairro Camobi, CEP 97.105-900 – Santa Maria, RS – Brasil

[2]Departamento de Informática
Universidade Federal do Paraná (UFPR)
Jardim das Américas, CEP 81530-900 - Curitiba, PR - Brasil

{rcpregardier, acneto, luisalvaro, csantos}@inf.ufsm.br,
bianchijr@gmail.com, vinicius@inf.ufpr.br

Abstract. *Conventional DGAs use fixed pseudo-random seeds, whereas adversarial DGAs adapt by incorporating the lexical and statistical features of benign domains. We present TITAN DGA, an adversarial domain-generation GAN that integrates a transformer-based autoencoder and Kullback–Leibler divergence to stabilize training. We tokenize benign domains via SentencePiece and use a transformer encoder–decoder to model character dependencies, aligning latent distributions with KL for realistic samples. In evaluations against FANCI, LSTM.MI, and Bilbo—and compared to CDGA, CharBot, Deception DGA, DeepDGA, and MaskDGA—TITAN DGA achieved superior results on evasion.*

Resumo. *DGAs convencionais usam sementes pseudoaleatórias fixas, enquanto os adversariais se adaptam, absorvendo traços léxicos e estatísticos de domínios benignos. Apresentamos o TITAN DGA, uma GAN adversarial que combina autoencoder transformer e divergência de Kullback–Leibler para estabilizar o treinamento. Tokenizamos domínios benignos com SentencePiece e empregamos um encoder–decoder transformer para modelar dependências de caracteres, alinhando distribuições latentes via KL para gerar amostras realistas. Em avaliações com os classificadores FANCI, LSTM.MI e Bilbo, e em comparação com CDGA, CharBot, Deception DGA, DeepDGA e MaskDGA, o TITAN DGA obteve desempenho superior em evasão.*

1. Introdução

Botnets, usadas tanto para planejar ataques de *Distributed Denial-of-Service* (DDoS), quanto para roubar informações dos equipamentos infectados, usam nomes de domínio para dificultar o bloqueio da comunicação entre os *bots* e o servidor de Comando e Controle (C&C) [Alieyan et al. 2017]. Porém, com o aumento do compartilhamento de *Indicators of Compromise (IOCs)* em geral pela comunidade e a adição destes domínios a blacklists [Antonakakis et al. 2012], surgiram os Algoritmos de Geração de Domínios

(DGAs). Tais algoritmos permitem que *malwares* mantenham comunicação com seus servidores (C&C) enquanto evitam a detecção por medidas tradicionais de cibersegurança. Sua expansão tem fomentado a geração e exploração maliciosa de milhares de domínios pseudorrandômicos por hora, sustentando ameaças como o trojan bancário GameOver Zeus e o Mirai que orquestra ataques *DDoS*. Os DGAs também desempenham uma função na facilitação de ataques de ransomware — como o CryptoLocker — e Ameaças Persistentes Avançadas (APTs).

DGAs tradicionais empregam uma semente (*seed*) pré-compartilhada — uma sequência numérica ou outro dado comum [Plohmann et al. 2016] — permitindo que hosts infectados e servidores de C&C compartilhem uma mesma sequência de domínios. Esse avanço expôs a fragilidade das defesas contra *botnets* e motivou a pesquisa de classificadores automáticos de DGAs explorando diferentes técnicas tradicionais de aprendizado de máquina [Schüppen et al. 2018], [Li et al. 2019] e de aprendizado profundo [Tran et al. 2018], [Woodbridge et al. 2016], [Highnam et al. 2020]. Com o aumento da eficácia destes detectores, uma nova ameaça surge: o DGA *adversarial*. Esta técnica é baseada em modelos de aprendizado de máquina que produzem DGAs baseados em domínios benignos, feitos especialmente para burlar os classificadores existentes, alcançando alta evasão em modelos treinados em *datasets* com famílias DGA já definidas. Nos modelos onde o retreinamento *adversarial* é aplicado, o desempenho dos detectores tende a um crescimento. Esse ciclo de ataque e defesa caracteriza uma "corrida armamentista" [Spooren et al. 2019], que evidencia a falta de robustez dos detectores em capturar certas características específicas na relação entre caracteres [Hu et al. 2023], [Zhai et al. 2022].

Nesse contexto, o objetivo deste trabalho é a criação de domínios maliciosos que sejam realistas, além de serem projetados para apresentar alta evasividade perante detectores, independentemente de seu funcionamento. Em específico, investigamos a aplicação de métodos de aprendizado de máquina na geração de dados, especialmente Redes Generativas Adversárias (GANs), com a incorporação de arquiteturas baseadas em *transformers*, de forma a permitir modelagem precisa de relações semânticas e estruturais complexas em sequências de texto. Propomos o TITAN DGA, cujas principais contribuições são:

- A proposição de uma nova arquitetura para geração de domínios maliciosos que une GANs com um *autoencoder* baseado em *transformers* – substituímos codificadores LSTM por um *autoencoder transformer* com estabilização por divergência de Kullback-Leibler (KL), com treinamento conjunto da GAN e do autoencoder, a fim de incentivar maior diversidade nas amostras geradas;
- Ao utilizar *tokenização* com *SentencePiece* e uma arquitetura *transformer*, o modelo é capaz de capturar dependências de curto e longo alcance entre *tokens*, gerando domínios mais difíceis de distinguir de domínios reais, demonstrando modelagem semântica e estrutural aprimorada;
- Avaliamos extensivamente o TITAN DGA contra três classificadores de referência de naturezas distintas (FANCI, LSTM.MI e Bilbo) e comparamos com cinco DGAs *adversários* estado-da-arte (CDGA, CharBot, DeceptionDGA, DeepDGA e MaskDGA). A arquitetura proposta apresentou os menores valores de F1-Score encontrados e altos índices de diversidade (Self-BLEU baixo), evidenciando sua superior capacidade de evadir a detecção e gerar domínios altamente distintos.

Este artigo está organizado da seguinte forma: Seção II revisa trabalhos relacionados e desafios na aplicação de GANs a DGAs; Seção III detalha o TITAN DGA; a Seção IV apresenta experimentos e resultados; e, por fim, a Seção V discute conclusões e aponta direções para trabalhos futuros.

2. Revisão da Literatura

2.1. Geração de Domínios Sintéticos

Redes Generativas Adversárias (GANs) vêm se destacando por sua capacidade de sintetizar tráfego de rede realista, fundamental para treinar e validar modelos de aprendizado de máquina em detecção de anomalias, otimização de desempenho e avaliação de robustez [Afifi et al. 2024]. O PAC-GAN [Lin et al. 2018] adapta arquiteturas de visão computacional: cada pacote é mapeado numa matriz 28×28 (bytes divididos em "nibbles") e reconstruído por deconvolução (com checksums recalculados). O [Bianchi et al. 2025] emprega transfer learning, pré-treinando um WGAN-GP em um protocolo "fonte" volumoso e, em seguida, afinando-o (dropout, redução de taxa de aprendizado, penalidade de gradiente) em um protocolo "alvo"; a qualidade é avaliada pelo Fréchet Inception Distance. Esses trabalhos mostram como métodos utilizados em outras áreas de pesquisa, como imagens, podem ser reaproveitados para gerar tráfego em múltiplos protocolos. Já trabalhos como o TITAN DGA utilizam GANs para criar domínios sintéticos com diferentes enfoques, explorando a afinidade léxica entre nomes de domínio e texto comum através de técnicas NLP que se assemelham aos padrões reais.

Arquiteturas que integram *transformers* e GANs têm sido exploradas para geração de texto. O Style Transformer-GAN [Zeng et al. 2020] utiliza codificadores distintos para estilo e conteúdo, combinando essas representações em um gerador baseado em *transformer*, avaliado por um discriminador que considera fluência e fidelidade ao estilo. O modelo aprende estilos implicitamente, sem dados paralelos, e apresenta bons resultados em BLEU [Papineni et al. 2002], perplexidade e acurácia de estilo. Já o TIL-GAN [Diao et al. 2021] adota uma estrutura latente implícita, na qual um *autoencoder* gera *embeddings* intermediários processados por um decodificador *transformer*, permitindo maior controle e diversidade na geração. Esses avanços motivam a adoção de estratégias semelhantes no TITAN DGA proposto.

O DeepDGA [Anderson et al. 2016] adota um *autoencoder* para compactar domínios reais e um gerador treinado adversariamente para produzir representações semelhantes a domínios legítimos, mas não captura dependências de longo alcance entre caracteres. O Khaos [Yun et al. 2020] propõe uma WGAN com técnicas baseadas em *n-grams* e um filtro similar ao do TITAN DGA, mas com geração mais restrita, favorecendo evasão em detrimento da diversidade. O CDGA [Zhai et al. 2022] emprega uma WGAN-GP com blocos ResNet, tokenização via SentencePiece, *embeddings* Word2Vec e redes LSTM, com treinamento separado entre *autoencoder* e GAN. Em contraste, o TITAN DGA integra ambos os treinamentos e adota *transformers* para modelar dependências longas entre *tokens*, incorporando ainda a divergência de Kullback-Leibler para estabilizar e diversificar o espaço latente, gerando domínios difíceis de detectar.

Abordagens alternativas à geração direta também exploram a manipulação de domínios existentes para criar DGAs adversariais. O Deception DGA [Spooren et al. 2019] modifica domínios gerados por DGAs ajustando padrões

linguísticos, como o equilíbrio entre vogais e consoantes. De forma similar, o CharBot [Peck et al. 2019] substitui aleatoriamente dois caracteres em domínios benignos para enganar classificadores, sem recorrer a técnicas de aprendizado de máquina. CLETer [Liu et al. 2021] e MaskDGA [Sidi et al. 2019] refinam essa ideia ao aplicar substituições baseadas em informações extraídas dos próprios classificadores: o CLETer altera os caracteres mais relevantes para a decisão, enquanto o MaskDGA usa transferibilidade adversária a partir de uma CNN substituta. Embora eficazes, essas abordagens dependem do acesso ao modelo de detecção ou da similaridade entre modelos. Em contraste, o TITAN DGA gera domínios com GANs, assegurando diversidade e realismo sem depender de informações internas dos classificadores.

Abordagens baseadas em séries temporais também são exploradas. O ReplaceDGA [Hu et al. 2023] utiliza uma rede BiLSTM para substituir caracteres de domínios benignos de forma informada, gerando domínios semanticamente similares, porém adversários. Apesar da alta taxa de evasão contra classificadores tradicionais, sua dependência de domínios legítimos limita a diversidade estrutural e a adaptabilidade. Além disso, ao alterar apenas dois caracteres por domínio, tende a produzir padrões previsíveis. Em contraste, o TITAN DGA gera domínios inteiramente novos, com maior complexidade e variedade, alcançando evasão robusta mesmo frente a múltiplos classificadores.

Após anos de desenvolvimento de DGAs, cada vez mais mostrou-se que os métodos de detecção baseados em block-lists e correlação entre requisições DNS estavam defasados. Então, juntamente com a criação de DGAs baseados em machine learning, surgiram também detectores DGA com a mesma base, buscando minimizar o tempo necessário para a classificação de um domínio. Esses classificadores são divididos em dois subgrupos: baseados em machine learning e baseados em deep learning.

2.2. Detectores

O FANCI [Schüppen et al. 2018] utiliza uma Random Forest com 21 características linguísticas para classificar domínios como benignos ou DGAs, com alta precisão e baixa taxa de falsos positivos, permitindo detecção em tempo real. No entanto, conforme [Spooren et al. 2019], seu desempenho é inferior ao de modelos baseados em aprendizado profundo. O B-RF [Sivaguru et al. 2018], também com Random Forest, adota 26 características, incluindo entropia, consoantes consecutivas e mediana de *n-grams* circulares. Embora eficaz em cenários simples, enfrenta limitações sob variações nas sementes dos DGAs. Avaliações comparativas revelam que seu recall é inferior ao de modelos como o B-LSTM.MI, que detectam o dobro de domínios maliciosos [Sivaguru et al. 2018], evidenciando a superioridade e a robustez dos métodos baseados em aprendizado profundo, que dispensam engenharia manual de características.

O trabalho em [Tran et al. 2018] propôs um detector de DGA baseado em redes LSTM, capaz de lidar com desbalanceamento de classes e alcançar altos F1-Scores em domínios benignos. Complementarmente, o Bilbo [Highnam et al. 2020] introduziu uma arquitetura híbrida com LSTM e CNN em paralelo, voltada à detecção de DGAs baseados em dicionário, com desempenho consistente em métricas como AUC, F1-Score e acurácia, superando modelos clássicos em generalização e resiliência temporal. Embora ambos explorem o potencial do aprendizado profundo, nosso trabalho gera domínios sintéticos mais diversos e evasivos, elevando a adaptabilidade frente a classificadores heterogêneos.

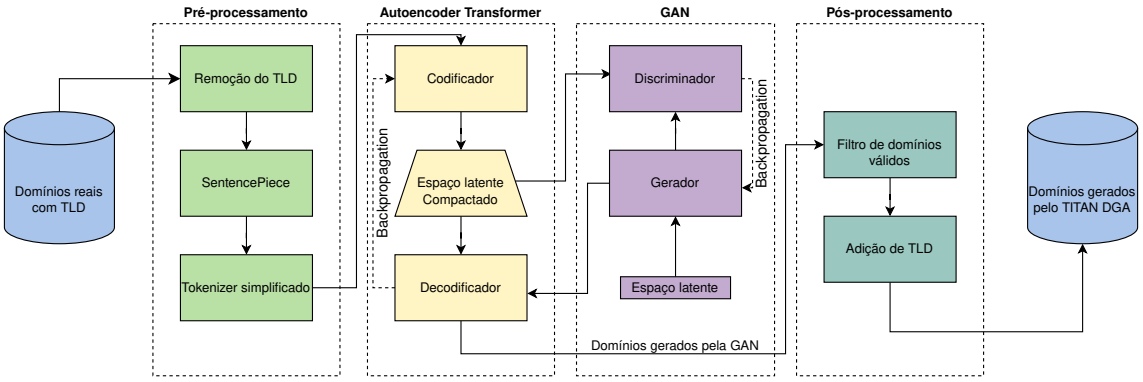

Figura 1. TITAN DGA - Arquitetura Geral

2.3. Discussão

Em conclusão, trabalhos apresentados na literatura revelam a evolução das técnicas de geração de domínios maliciosos, desde abordagens baseadas em LSTM e manipulação de domínios legítimos até modelos adversariais com GANs. Trabalhos como o DeepDGA [Anderson et al. 2016] e o CDGA [Zhai et al. 2022] avançaram ao integrar autoencoders e tokenização para emular domínios legítimos, mas ainda sofrem com a separação entre geração e reconstrução e dependências de longo alcance mal capturadas [Hu et al. 2023]. O TITAN DGA, por sua vez, propõe uma integração direta entre autoencoder e GAN, utilizando transformers e divergência KL para estabilização, gerando domínios mais coesos e evasivos sem ajustes específicos para cada classificador, superando limitações anteriores.

3. Geração de Domínios Maliciosos Usando Transformers

Esse trabalho propõe a geração de DGAs utilizando uma GAN baseada em *transformers*. O emprego dessa abordagem possibilita aprender relações de curto e longo alcance entre os diferentes *tokens*, o que possibilita a maior evasão e diversidade de domínios gerados. Este trabalho utiliza datasets de DGAs disponíveis, junto a listas de domínios benignos, onde o modelo proposto é capaz de enganar classificadores existentes na literatura.

O *framework* proposto do TITAN DGA, ilustrado na Figura 1, é composto por quatro módulos principais:

1. ***Dataset* de treinamento:** essa etapa é composta pela escolha e preparação dos dados a serem utilizados no treinamento da GAN, juntando DGAs de diferentes famílias com domínios benignos, provenientes de *datasets* conhecidos na literatura.
2. **Pré-processamento:** neste estágio, os domínios benignos têm seus TLDs removidos. Em seguida, são *tokenizados* utilizando dois métodos: o *tokenizador SentencePiece* e um *tokenizador* simplificado que insere os *tokens* especiais. Isso prepara os dados para entrada no *autoencoder* e reduz a complexidade do vocabulário.
3. ***Autoencoder*:** este módulo é responsável por codificar os domínios *tokenizados* em um espaço latente de menor dimensionalidade. O codificador (*encoder*) transforma os domínios em vetores latentes compactados, que são utilizados pelo discriminador da GAN como amostras reais. O decodificador (*decoder*), por sua vez,

reconstrói os domínios a partir desses vetores, tanto os oriundos do codificador quanto os gerados pelo gerador da GAN. O treinamento do *autoencoder* ocorre de forma conjunta com o da GAN, sendo ambos otimizados com base na divergência de Kullback-Leibler.

4. **GAN:** a GAN é composta por um gerador e um discriminador. O gerador é treinado para produzir vetores no espaço latente que se assemelhem aos gerados pelo codificador, enquanto o discriminador tenta distinguir entre os vetores latentes reais (vindos do codificador) e os sintéticos (produzidos pelo gerador). O treinamento adversário aprimora progressivamente a qualidade dos vetores sintéticos.
5. **Pós-processamento:** após a reconstrução dos domínios pelo decodificador, é aplicado um filtro para garantir que os domínios gerados estejam em conformidade com as RFCs 1034 e 1035. Além disso, um TLD válido é selecionado a partir de uma lista pré-definida, eliminando a necessidade de geração de TLDs pela GAN e assegurando a validade estrutural dos domínios resultantes.

3.1. Datasets de Treinamento e Teste

No total, três *datasets* foram utilizados neste trabalho: um para o treinamento da GAN, outro para o treinamento dos classificadores, e um conjunto de datasets para a inferência dos geradores de DGA da literatura, incluindo o TITAN DGA.

Para a seleção de domínios benignos, este trabalho utiliza o Tranco [Le Pochat et al. 2019], contendo uma lista dos domínios mais acessados mundialmente, elaborada para evitar manipulações comuns em listas tradicionais como a Alexa Top-n Domains. Para os domínios gerados por DGAs, o DGArchive [Fraunhofer FKIE 2020] é utilizado. Disponibilizado pela Fraunhofer FKIE, esse dataset contém mais de 9GB de domínios de diferentes famílias.

O primeiro dataset, denominado **GT**, foi utilizado para o treinamento da GAN. Ele é composto por 140 mil domínios benignos extraídos do Tranco e 140 mil domínios DGA retirados do DGArchive, abrangendo sete famílias distintas: *Conficker*, *Gozi*, *Matsnu*, *Pykspa*, *Simda*, *Suppobox* e *Ud2*.

O segundo dataset, chamado **DB1**, foi utilizado para o treinamento dos classificadores FANCI [Schüppen et al. 2018], LSTM.MI [Tran et al. 2018] e Bilbo [Highnam et al. 2020]. Ele é composto por 500 mil domínios benignos (**B1**) extraídos do Tranco e 500 mil domínios DGA (**D1**) provenientes de 10 famílias diferentes, sendo duas baseadas em *hash* (*Bamital* e *Dyre*), cinco em aritméticas (*Banjori*, *Conficker*, *Cryptolocker*, *Nymaim* e *Pykspa*) e três em listas de palavras (*Gozi*, *Matsnu* e *Suppobox*).

Na fase de inferência dos modelos construídos, foram utilizados datasets individuais para cada gerador de DGA testado, bem como para o TITAN DGA. Cada conjunto é composto por 10 mil domínios benignos e 10 mil domínios gerados por cada respectivo DGA, organizados da seguinte forma: **CDGA_IN** (10k CDGA + 10k benignos), **CHAR_IN** (10k CharBot + 10k benignos), **DEEP_IN** (10k DeepDGA + 10k benignos), **DECE_IN** (10k DeceptionDGA + 10k benignos), **MASK_IN** (10k MaskDGA + 10k benignos) e **TITA_IN** (10k TITAN DGA + 10k benignos).

3.2. Pré-processamento e Tokenização dos Dados

O pré-processamento dos dados é similar a [Zhai et al. 2022], utilizando o *SentencePiece* para a tokenização dos domínios. Após a criação dos *datasets*, os domínios passam

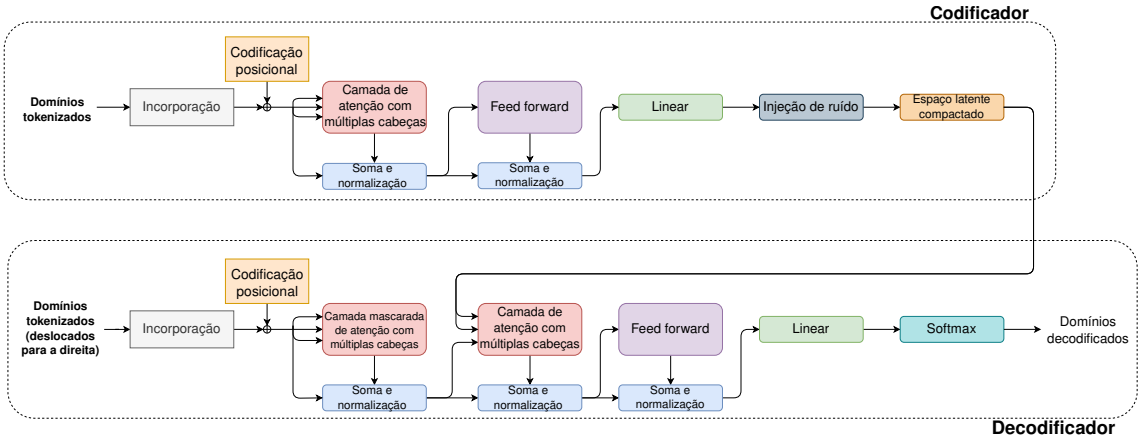

Figura 2. Arquitetura do modelo de *autoencoder*.

por uma etapa de remoção dos *Top Level Domains* (TLDs). Essas terminações pertencem a um conjunto pequeno e fixo de possibilidades, dispensando sua geração direta pela GAN. Posteriormente, é realizada a segmentação dos domínios utilizando o *SentencePiece*, uma ferramenta desenvolvida pelo Google para a geração de texto em redes neurais, que permite definir o vocabulário antes do treinamento. Dentre as técnicas disponibilizadas, optou-se pelo uso do *Unigram Language Model*, devido à sua capacidade de lidar melhor com palavras desconhecidas e variações morfológicas — características comuns em nomes de domínios. A segmentação permite dividir domínios em unidades menores e mais coerentes, como no exemplo *"steamstat.us"*, que pode ser separado em *"steam"* e *"stat"*, aumentando a diversidade e a naturalidade dos domínios gerados. Em seguida, os dados *tokenizados* são convertidos em sequências de índices numéricos, adicionando-se *tokens* especiais de "início de sequência" (sos) e "fim de sequência" (eos).

3.3. Descrição da Arquitetura da GAN + *autoencoder*

A arquitetura da GAN investigada neste trabalho é composta por um *autoencoder* baseado em *transformer*, demonstrado na Figura 2, com duas camadas de *self-attention* e quatro cabeças de atenção, seguido por uma camada *feedforward* com 512 unidades e uma dimensão latente comprimida de tamanho 56. Essa dimensão comprimida atua como um espaço de representação condensado da sequência de entrada, capturando suas características mais relevantes em um vetor fixo de tamanho reduzido. O objetivo é que o modelo aprenda a codificar informações essenciais dos domínios nesse espaço latente, servindo de ponte entre o codificador e o decodificador.

O gerador e o discriminador são implementados como redes *Multi Layer Perceptron* (MLP) de uma camada com 128 neurônios, o que reduz a complexidade sem comprometer a capacidade de aprendizado. A ativação *LeakyReLU* é utilizada para garantir estabilidade no treinamento, e a função de perda baseada na divergência de Kullback-Leibler foi escolhida por favorecer a geração de amostras diversificadas e distribuídas de forma semelhante às observações reais. A divergência de Kullback-Leibler é utilizada neste trabalho como parte da função de perda adversária no treinamento da TITAN GAN, atuando diretamente sobre a distribuição das representações latentes geradas. O gerador latente transforma amostras de uma distribuição simples $\varepsilon \sim \mathcal{N}(0, I)$ por meio da rede MLP

$g_\beta(\varepsilon)$, resultando em uma distribuição implícita $p_\beta(z)$. O objetivo é que essa distribuição se aproxime da distribuição agregada do codificador $q_\phi(z) = \mathbb{E}_{x\sim p_r(x)}[q_\phi(z|x)]$, o que é alcançado minimizando a divergência KL:

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

Esse termo é incorporado à função de perda total com um coeficiente de ponderação, incentivando que o gerador latente cubra toda a diversidade presente nas distribuições codificadas. É importante destacar que, diferentemente de modelos variacionais (VAEs), o *autoencoder* da TITAN GAN não utiliza a divergência KL para regularizar a aproximação entre $q(z|x)$ e um prior explícito $p(z)$. Em vez disso, a divergência KL atua no nível da distribuição marginal, reforçando a diversidade e compatibilidade entre os espaços latentes do gerador e do codificador.

Além disso, uma segunda divergência KL é introduzida com o objetivo de aprimorar o decodificador. Ela mede a discrepância entre a distribuição do codificador aplicada sobre amostras reconstruídas $\tilde{z} = E(G(g(\varepsilon)))$ e a distribuição original $q_\phi(z)$:

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

Essa penalização adicional assegura que o decodificador aprenda a lidar com representações provenientes do gerador, promovendo maior fidelidade na geração textual, mesmo quando confrontado com vetores latentes nunca vistos durante o treinamento.

3.4. Treinamento do modelo

Durante o treinamento do modelo, foi adotado o otimizador Adam para todos os componentes da arquitetura, incluindo o gerador, o discriminador e o *autoencoder*. Essa escolha se deve à sua eficácia em estabilizar e acelerar a convergência de modelos baseados em aprendizado profundo, especialmente em arquiteturas adversárias, como as GANs. Além disso, uma taxa de *dropout* de 30% foi aplicada em diferentes partes do *autoencoder*, como após as cabeças de atenção e após a camada *feedforward*. Isso é realizado para regularizar a atenção, mas também a transformação não linear aplicada dos *embeddings*.

O treinamento foi realizado com um tamanho de lote (*batch size*) de 256 amostras, valor que oferece um bom equilíbrio entre estabilidade da atualização dos gradientes e utilização eficiente da memória. Em relação às taxas de aprendizado (*learning rates*), diferentes valores foram empregados para cada componente da arquitetura: o gerador foi treinado com uma taxa de $0{,}0004$, o discriminador com $0{,}0002$, e o *autoencoder* com uma taxa relativamente mais alta de $0{,}06$. A escolha desses hiperparâmetros se deu após uma extensa análise empírica, na qual diversos experimentos foram conduzidos para avaliar o impacto de diferentes combinações de taxas de aprendizado e tamanhos de lote sobre o desempenho do modelo. Durante essa fase exploratória, observou-se que valores mais altos de *learning rate* para o discriminador e o *autoencoder* levavam ao sobreajuste (*overfitting*), fazendo com que esses componentes memorizassem os padrões de treinamento e perdessem a capacidade de generalização. Por outro lado, uma taxa de aprendizado muito baixa para o *autoencoder* resultava em subajuste (*underfitting*), impedindo que a rede aprendesse representações úteis dos dados de entrada. Assim, os valores finais adotados

Tabela 1. Exemplos de nomes de domínios gerados por modelos de DGAs *adversariais* diferentes (com o TLD removido).

TITAN DGA	CDGA	MaskDGA	CharBot	Deception DGA	DeepDGA
tradeworkreporter	uqlzfdfd	semapisimtda	banzhew	ntrvenker	thellehm
bike10life	brotherlicwo	uoruiddvwdwuihkul	sancopatvgonia	motuseroure	shtrunoa
brainmagazine	thethethmmili	sfqjkbu63r1hyfa8543	puttpandroid	wolan290451	vietips
saubrasil	theworkliimoarview	sooackgeicesmsgi	matctendgrect	olyuchedu	statpottxy

refletem um equilíbrio alcançado empiricamente entre estabilidade no treinamento, boa capacidade de generalização e eficiência computacional. A diferenciação entre os valores de *learning rate* busca atender às distintas sensibilidades e dinâmicas de otimização de cada módulo, sendo comum em GANs atribuir ao discriminador um *learning rate* menor que o do gerador para evitar o sobreajuste precoce e manter a competição equilibrada entre os dois.

A dimensão latente de entrada do gerador foi definida como um vetor de 100 dimensões, amostrado a partir de uma distribuição normal padrão $\mathcal{N}(0, I)$. Essa escolha visa fornecer um espaço latente suficientemente expressivo para capturar a diversidade das distribuições dos dados reais, sem comprometer a estabilidade do treinamento.

Por fim, foi adicionado ruído gaussiano controlado tanto no processo de treinamento quanto na inferência do *autoencoder*. Essa técnica visa aumentar a robustez do modelo, encorajando o aprendizado de representações latentes que sejam mais generalizáveis, mesmo sob pequenas perturbações nos dados de entrada.

3.5. Pós-processamento

A etapa do pós-processamento apresentada neste trabalho é dividida em duas tarefas menores: a filtragem de domínios válidos e a adição de TLD aos domínios gerados. A filtragem dos domínios ocorre em conformidade com os RFCs 1034 e 1035, que definem características básicas de como um domínio deve ser formado. Sendo assim, domínios fora do intervalo de tamanho válido são descartados, assim como domínios que terminam em hífen; além disso, a probabilidade de geração de domínios semelhantes é reduzida. Na Tabela 1 podemos ver alguns exemplos dos domínios gerados pelo TITAN DGA após o pós-processamento, porém sem o TLD, apenas para fins de comparação entre eles.

Dessa forma, para finalizar a criação dos novos domínios, a escolha de um TLD é realizada a partir de uma lista de dezenas de TLDs. A escolha é feita de maneira aleatória, a fim de proporcionar uma maior diversidade de domínios que podem ser gerados.

4. Experimentos e Resultados

O objetivo dos experimentos realizados é avaliar o desempenho e a adaptabilidade dos dados gerados pelo TITAN DGA, com base em classificadores utilizados na literatura. Além disso, buscou-se investigar métricas que capturam a diversidade das amostras geradas, bem como a incidência de repetições nos domínios sintetizados. Essa análise permite compreender tanto a eficácia da geração quanto o seu potencial para enganar detectores de DGA e enriquecer conjuntos de dados sintéticos.

Inicialmente, foi construído um conjunto de dados representativo, com instâncias suficientes das principais famílias de DGA, a fim de garantir o treinamento eficaz dos classificadores. Para conduzir os experimentos, as seguintes etapas foram realizadas:

1. **Treinamento dos modelos classificadores:** foram utilizados três classificadores distintos, baseados em abordagens da literatura [Schüppen et al. 2018, Tran et al. 2018, Highnam et al. 2020]. Esses classificadores foram treinados individualmente usando 80% dos dados disponíveis no *dataset* DB1. O classificador FANCI [Schüppen et al. 2018] conta originalmente com 21 características (*features*) para a classificação. No nosso trabalho, os TLDs são selecionados de uma lista fixa. Logo, somente 16 características foram utilizadas nos experimentos. Cada modelo foi configurado conforme especificações originalmente propostas, respeitando as arquiteturas e hiperparâmetros sugeridos nos trabalhos de referência.
2. **Inferência e avaliação de classificadores:** após o treinamento, os classificadores foram submetidos ao processo de inferência, utilizando os dados gerados pelo TITAN DGA. O objetivo foi verificar a capacidade de evasão dos domínios sintetizados, observando o percentual de domínios que não foram corretamente classificados ou que foram erroneamente atribuídos a outra classe.
3. **Comparação com outros DGAs da literatura:** os DGAs gerados pelos modelos propostos neste trabalho foram comparados a outros cinco modelos disponíveis da literatura: CDGA [Zhai et al. 2022], CharBot [Peck et al. 2019], DeceptionDGA [Spooren et al. 2019], DeepDGA [Anderson et al. 2016] e MaskDGA [Sidi et al. 2019].
4. **Cálculo das métricas complementares:** métricas complementares foram exploradas, incluindo a diversidade lexical dos domínios gerados (*Self-BLEU*) e o tamanho médio dos domínios gerados. Essas métricas são essenciais para avaliar a utilidade do modelo gerador na construção de amostras realistas, porém difíceis de detectar.

A validação dos experimentos foi realizada de acordo com diferentes métricas. Primeiramente, foi executada a verificação do treinamento dos classificadores com base em duas métricas de validação de classificadores binários: a área sob a curva (AUC) e a curva ROC (*Receiver Operating Characteristic*). Essas duas métricas são quantificadas entre 0 e 1, onde o resultado mais próximo de 1 é melhor. Essa métrica foi empregada para assegurar que os classificadores alcançaram um bom desempenho nos dados reais. Além disso, para a avaliação do desempenho dos detectores nos dados sintéticos, foram utilizadas outras 4 métricas: a precisão, a acurácia, o recall e o F1-Score. Note que, para a habilidade de anti-detecção, quanto menores os valores dessas métricas, melhores os resultados. Como métrica final, este trabalho explora o Self-BLEU [Papineni et al. 2002] na comparação qualitativa dos dados gerados. Importante destacar que quanto mais próximas de 0 são as relações entre os *n-grams*, mais diverso é o texto. Essa métrica foi empregada devido à sua ampla utilização em avaliações de modelos mais gerais voltados para a geração de texto, pois explora a verificação de repetição de *n-grams* nos textos gerados.

Com base nos resultados obtidos no dataset de validação DB1 na Figura 3, podemos observar que os classificadores Bilbo e LSTM MI apresentaram um desempenho excepcional, ambos alcançando um AUC (Área Sob a Curva) de 0.98. Esse resultado indica uma alta capacidade de discriminação entre os domínios benignos e DGAs, demonstrando elevada eficácia na tarefa de classificação. Por outro lado, o classificador FANCI obteve um AUC de 0.83. Embora este resultado ainda possa ser considerado bom, está significativamente abaixo dos outros dois modelos. Isso é explicado pelo FANCI ser um

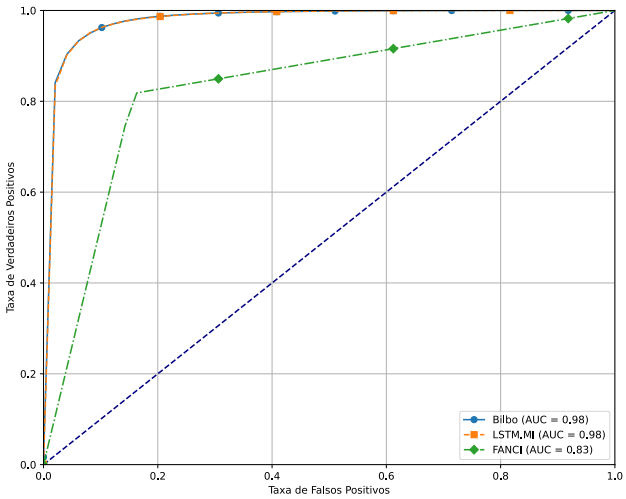

Figura 3. Curva ROC e AUC dos 3 classificadores distintos treinados e validados nos 20% de validação do *dataset* DB1.

modelo de Random Forest, que é mais simplificado do que modelos de *Deep Learning* baseados em CNNs ou LSTMs, os quais são capazes de identificar padrões mais complexos presentes em alguns DGAs.

A análise da curva ROC reforça esses achados, pois tanto Bilbo quanto LSTM.MI apresentam taxas de verdadeiros positivos (TVP) elevadas em relação às taxas de falsos positivos (TFP), o que é desejável em um classificador robusto. O FANCI demonstra uma relação menos favorável entre TVP e TFP, o que pode indicar uma maior tendência a erros de classificação em cenários com dados desbalanceados ou ruidosos.

Com relação à inferência dos dados sintéticos no classificador FANCI, os resultados apresentados na Tabela 2 indicam que o TITAN DGA apresenta o maior impacto negativo sobre o desempenho do classificador, destacando-se como o modelo com maior capacidade de evasão entre os DGAs analisados. O TITAN DGA obteve os menores valores em todas as métricas, indicando que seus domínios são os mais difíceis de serem corretamente identificados como maliciosos. Esse comportamento evidencia uma habilidade acentuada de camuflagem frente aos mecanismos de detecção do FANCI, especialmente na relação entre precisão e F1-score. Embora sua precisão (0,4411) seja aproximadamente 37,5% inferior à do MaskDGA (0,7058), a diferença no F1-score se amplia para 42,6% (0,3915 contra 0,6816). Comparando com o DeceptionDGA, que obteve um F1-score de 0,4470, o TITAN DGA ainda apresenta um desempenho de classificação pior, o que reforça sua habilidade de evadir detecção, mesmo quando comparado a um modelo projetado especificamente para burlar o FANCI. Esse comportamento compromete o trade-off entre falsos positivos e falsos negativos, tornando mais difícil distinguir domínios legítimos dos gerados pelo TITAN DGA.

Resultados semelhantes foram observados na inferência sobre o classificador BILBO (Tabela 4). O TITAN DGA novamente apresenta os menores resultados entre os modelos, com destaque para o F1-score de apenas 0,2117, frente a valores significativa-

Tabela 2. Resultados de avaliação obtidos na inferência dos *datasets* de inferência dos DGAs da literatura no classificador FANCI.

Métricas	CharBot	DeceptionDGA	MaskDGA	CDGA	DeepDGA	TITAN DGA
Precisão	0.5980	0.5129	0.7058	0.6342	0.6467	**0.4411**
Acurácia	0.5710	0.5073	0.6884	0.6057	0.6205	**0.4724**
Recall	0.5710	0.5073	0.6884	0.6057	0.6205	**0.4724**
F1-Score	0.5393	0.4470	0.6816	0.5836	0.6028	**0.3915**

Tabela 3. Resultados de avaliação obtidos na inferência dos *datasets* de inferência dos DGAs da literatura no classificador LSTM.MI.

Métricas	CharBot	DeceptionDGA	MaskDGA	CDGA	DeepDGA	TITAN DGA
Precisão	0.8317	0.7673	0.9503	0.9045	0.9304	**0.7271**
Acurácia	0.5890	0.5500	0.8767	0.6838	0.7748	**0.5354**
Recall	0.2233	0.1434	0.7950	0.4111	0.5939	**0.1135**
F1-Score	0.3521	0.2416	0.8657	0.5653	0.7250	**0.1963**

mente mais altos dos demais DGAs, como o MaskDGA (0,8776) e o DeepDGA (0,6235). A baixa pontuação de recall (0,1234) também reforça a hipótese de que os domínios gerados pelo TITAN DGA escapam à detecção da arquitetura, mesmo em classificadores baseados em mecanismos distintos, como o BILBO, que combina aprendizado de embeddings com técnicas supervisionadas. Essa repetição do padrão de evasão sugere que a capacidade de disfarce do TITAN DGA é, de fato, transversal a diferentes abordagens de modelagem.

Considerando os três classificadores — FANCI, LSTM.MI e BILBO — o TITAN DGA exibe uma evasão consistente, apresentando os menores valores de F1-score em todos os casos (0,3915, 0,1963 e 0,2117, respectivamente). Essa consistência reforça a hipótese de que os domínios sintéticos gerados por esse modelo possuem características altamente adversariais, capazes de explorar vulnerabilidades comuns entre diferentes estratégias de detecção. Em particular, a combinação de baixos valores de recall e precisão implica não apenas uma menor taxa de detecção, mas também um desbalanceamento crítico nas classificações, comprometendo a sensibilidade dos sistemas, mesmo com arquiteturas distintas. Esses resultados destacam, sobretudo, o papel da arquitetura baseada em *transformers*, que é crucial para a geração de textos coesos e bem estruturados. A capacidade dos *transformers* em modelar sequências complexas e gerar domínios com características adversariais eficazes foi determinante para alcançar esses resultados, evidenciando a relevância dessa tecnologia na criação de DGAs difíceis de detectar.

A Figura 4 mostra a pontuação Self-BLEU ao longo das épocas de treinamento da

Tabela 4. Resultados de avaliação obtidos na inferência dos *datasets* de inferência dos DGAs da literatura no classificador BILBO.

Métricas	CharBot	DeceptionDGA	MaskDGA	CDGA	DeepDGA	TITAN DGA
Precisão	0.8564	0.7847	0.9517	0.9077	0.9190	**0.7443**
Acurácia	0.6079	0.5545	0.8864	0.6860	0.7151	**0.5405**
Recall	0.2594	0.1502	0.8142	0.4141	0.4718	**0.1234**
F1-Score	0.3982	0.2522	0.8776	0.5687	0.6235	**0.2117**

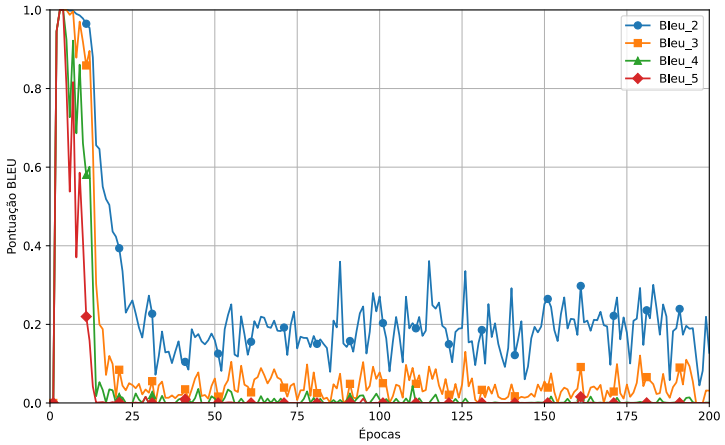

Figura 4. *Self-BLEU score* através das épocas do treinamento da GAN.

GAN, evidenciando a diversidade dos dados gerados. Cada linha desenhada no gráfico representa o número de *n-grams* considerados para o cálculo do Self-BLEU, por exemplo, a linha Bleu-2 representa a diversidade textual presente a cada época do treinamento do modelo, medindo a repetição de 2 *n-grams* em relação a todos os exemplares gerados na época. É importante ressaltar que, para o cálculo do BLEU, o TLD não é considerado. Por volta da época 20, o modelo começou a evitar a repetição de n-gramas de tamanho 3 e 4, destacando sua rápida capacidade de aprender a gerar domínios diversificados. Após a época 25, a geração de domínios não apenas se estabilizou, mas também se tornou mais variada, refletindo a habilidade da GAN em gerar domínios diferentes, não apenas no nível de nomes completos, mas também nos *n-grams* das palavras. Esse comportamento ressalta a eficácia do *autoencoder* em transformar os domínios gerados.

Tabela 5. Tamanho médio dos domínios e desvio padrão.

Tipo do Domínio	CDGA	CharBot	DeceptionDGA	DeepDGA	MaskDGA	TITAN DGA	Legítimos
Tamanho médio	10.98 ± 4.82	10.51 ± 3.24	10.21 ± 3.91	16.53 ± 5.20	15.19 ± 5.30	6.83 ± 3.07	9.63 ± 4.27

Apesar dos domínios gerados pelo TITAN DGA apresentarem, em média, um tamanho menor, em relação a outros DGAs adversários e domínios legítimos, de acordo com a Tabela 5, é importante ressaltar que a escolha dos tamanhos médios dos domínios a serem gerados pelo TITAN DGA é feita durante o treinamento. A escolha por domínios menores se deu para a demonstração da capacidade da geração de domínios diversos, pois o gráfico de Self-BLEU (Figura 4) evidencia que, mesmo com nomes mais curtos, a diversidade dos domínios gerados é mantida, com a pontuação Self-BLEU continuando baixa, indicando que o modelo está gerando continuamente domínios distintos, sem repetição significativa.

5. Conclusão

Algoritmos de Geração de Domínios (DGAs) desempenham um papel central no cenário atual de cibersegurança, dada sua importância para a manutenção da comunicação entre *bots* e servidores C&C, dificultando a detecção por sistemas tradicionais. A evolução

constante dos DGAs adversários evidencia a necessidade de soluções inovadoras para testes de robustez e para o fortalecimento de detectores.

Neste trabalho, abordamos o problema da geração de domínios maliciosos altamente realistas e evasivos, com foco na superação dos classificadores existentes. A dificuldade de gerar domínios adversários que sejam simultaneamente diversos, realistas e difíceis de detectar, especialmente em cenários sem conhecimento prévio dos detectores, motivou as contribuições propostas.

O TITAN DGA apresenta avanços relevantes para a área, ao combinar *autoencoders* baseados em *transformers* e estabilização via divergência KL, promovendo uma geração de domínios sintéticos mais coesa e diversa. Experimentalmente, o TITAN DGA demonstrou desempenho superior de evasão frente a detectores consagrados como FANCI, LSTM.MI e Bilbo, alcançando os menores valores de *F1-Score* em todas as avaliações. Além disso, métricas como *Self-BLEU* confirmaram a alta diversidade léxica dos domínios gerados, reforçando a eficácia da arquitetura proposta.

Para trabalhos futuros, pretendemos adicionar módulos de estilo e controle semântico, expandindo a variabilidade dos domínios gerados. Também planejamos expandir as avaliações apresentadas neste trabalho, além de analisar a eficácia do modelo contra detectores treinados com dados adversários.

Referências

Afifi, H., Pochaba, S., Boltres, A., Laniewski, D., Haberer, J., Paeleke, L., Poorzare, R., Stolpmann, D., Wehner, N., Redder, A., Samikwa, E., and Seufert, M. (2024). Machine learning with computer networks: Techniques, datasets, and models. *IEEE Access*, 12:54673–54720.

Alieyan, K., ALmomani, A., Manasrah, A., and Kadhum, M. M. (2017). A survey of botnet detection based on dns. *Neural Computing and Applications*, 28(7):1541–1558.

Anderson, H. S., Woodbridge, J., and Filar, B. (2016). Deepdga: Adversarially-tuned domain generation and detection. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, AISec '16, page 13–21, New York, NY, USA. Association for Computing Machinery.

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

Bianchi, J.-L. A. C., Pregardier, R. C., Silva, L. A. L., and dos Santos, C. R. P. (2025). 2Pack-GAN: Exploring transfer learning to fine-tune generative adversarial networks for network packet generation. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS)*, Honolulu, HI, USA. IEEE.

Diao, S., Shen, X., Shum, K., Song, Y., and Zhang, T. (2021). TILGAN: Transformer-based implicit latent GAN for diverse and coherent text generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4844–4858, Online. Association for Computational Linguistics.

Fraunhofer FKIE (2020). Dgarchive: Database of domain generation algorithm domains. `https://dgarchive.caad.fkie.fraunhofer.de/`.

Highnam, K., Puzio, D., Luo, S., and Jennings, N. R. (2020). Real-time detection of dictionary dga network traffic using deep learning.

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

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

Li, Y., Xiong, K., Chin, T., and Hu, C. (2019). A machine learning framework for domain generation algorithm-based malware detection. *IEEE Access*, 7:32765–32782.

Lin, Z., Khetan, A., Fanti, G., and Oh, S. (2018). Pacgan: The power of two samples in generative adversarial networks. *Advances in neural information processing systems*, 31.

Liu, W., Zhang, Z., Huang, C., and Fang, Y. (2021). Cleter: A character-level evasion technique against deep learning dga classifiers. *EAI Endorsed Transactions on Security and Safety*, 7(24).

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics.

Peck, J., Nie, C., Sivaguru, R., Grumer, C., Olumofin, F., Yu, B., Nascimento, A., and De Cock, M. (2019). Charbot: A simple and effective method for evading dga classifiers. *IEEE Access*, 7:91759–91771.

Plohmann, D., Yakdan, K., Klatt, M., Bader, J., and Gerhards-Padilla, E. (2016). A comprehensive measurement study of domain generating malware. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, page 263–278, USA. USENIX Association.

Schüppen, S., Teubert, D., Herrmann, P., and Meyer, U. (2018). Fanci: feature-based automated nxdomain classification and intelligence. In *Proceedings of the 27th USENIX Conference on Security Symposium*, SEC'18, page 1165–1181, USA. USENIX Association.

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

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

Spooren, J., Preuveneers, D., Desmet, L., Janssen, P., and Joosen, W. (2019). Detection of algorithmically generated domain names used by botnets: a dual arms race. In

Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC '19, page 1916–1923, New York, NY, USA. Association for Computing Machinery.

Tran, D., Mac, H., Tong, V., Tran, H. A., and Nguyen, L. G. (2018). A lstm based framework for handling multiclass imbalance in dga botnet detection. *Neurocomputing*, 275:2401–2413.

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

Yun, X., Huang, J., Wang, Y., Zang, T., Zhou, Y., and Zhang, Y. (2020). Khaos: An adversarial neural network dga with high anti-detection ability. *IEEE Transactions on Information Forensics and Security*, 15:2225–2240.

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

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