Detecção de Intrusões em Tráfego DNS utilizando Redes Neurais e Explicabilidade

Giovanna Luisa Hirata dos Anjos[1], **Daniel Macêdo Batista**[1]

[1]Instituto de Matemática, Estatística e Ciência da Computação
Universidade de São Paulo (USP) – São Paulo, SP – Brasil

giovannaluisahirata@usp.br, batista@ime.usp.br

Abstract. *The proliferation of cyberattacks, such as phishing and malware, relies on malicious domain names to store artifacts. Blocking access to these domains can be the first line of defense against the spread of malicious artifacts on the Internet. However, the speed at which they are created makes traditional approaches, based on blocklists, insufficient for effective protection. This paper presents a proactive intrusion detection system (IDS) that analyzes DNS traffic using a neural network model. Experimental results show that the explainability step is crucial, and that the model can detect benign traffic with an average F1-Score of 0.95 and malicious traffic with an average F1-Score of 0.82.*

Resumo. *A proliferação de ciberataques como phishing e malware depende de nomes de domínio maliciosos para a hospedagem de artefatos. Impedir o acesso a esses domínios pode ser a primeira barreira de proteção contra a disseminação dos artefatos maliciosos na Internet. Porém, a velocidade com que eles são criados torna as abordagens tradicionais, baseadas em listas de bloqueio, insuficientes para uma proteção eficaz. Este artigo apresenta um sistema de detecção de intrusão (IDS) proativo que analisa o tráfego DNS utilizando um modelo de redes neurais. Experimentos mostram que a etapa de explicabilidade é fundamental e que o modelo é capaz de detectar tráfego benigno com F1-Score médio de 0,95 e malicioso com F1-Score médio de 0,82.*

1. Introdução

Sistemas de detecção de intrusão (IDSs – *Intrusion Detection Systems*) utilizam informações de diversas camadas da arquitetura Internet para alertar sobre potenciais ataques em uma rede de computadores. Embora hajam ressalvas relacionadas à privacidade quando da utilização de informações das camadas superiores, conforme discutido em [Elias et al. 2022], o protocolo DNS (*Domain Name System*) ainda é transportado sem criptografia em algumas comunicações pelas redes, podendo auxiliar na detecção de ataques. Por exemplo, a detecção imediata de dispositivos de uma rede acessando nomes de domínio criados exclusivamente para hospedar cópias de sites oficias com o objetivo de obter, de forma ilegal, credenciais de acesso aos sites originais (*phishing*) pode ser usada como gatilho para a criação de uma regra em um *firewall* ou em um servidor DNS para impedir que o tráfego suspeito seja transportado. Entretanto, a criação de nomes de domínio com intenção maliciosa ocorre a todo momento e detectar essas criações de forma antecipada é algo difícil de ser realizado sem algum mecanismo automatizado. Mesmo que essa detecção ocorra, o processo burocrático para "derrubar" os domínios toma tempo, justificando mecanismos de mitigação que atuem diretamente nos dispositivos de interconexão.

Um procedimento para mitigar esse problema consiste em 1-) capturar o tráfego DNS; 2-) processar o mesmo por meio de algum mecanismo de aprendizado de máquina que identifique nomes de domínio maliciosos; e 3-) criar regras em servidores DNS ou *firewalls* para impedir que o tráfego para esses domínios tenha sucesso. Em vista desse processo, este artigo apresenta um mecanismo baseado em redes neurais para atender ao item 2-). Como objetivos secundários também foram realizados: aplicação de *feature engineering* por meio do estudo de explicabilidade e comparação do desempenho do mecanismo com outros modelos para classificação de tráfego.

Resultados obtidos com a utilização do mecanismo na detecção de domínios maliciosos presentes no dataset BCCC-CIC-Bell-DNS-2024 ([Behaviour-Centric Cybersecurity Center (BCCC) 2024] [Shafi et al. 2024]), um dataset criado para modelar o perfil comportamental de diferentes atividades de DNS, mostraram que a etapa de explicabilidade foi fundamental para identificar *features* irrelevantes, levando o modelo a detectar tráfego benigno com F1-Score médio de 0,95 e tráfego malicioso com F1-Score médio de 0,82 (Para garantir a reprodutibilidade do trabalho, todo o código desenvolvido está disponibilizado como software livre).

O restante deste artigo está organizado da seguinte forma: a Seção 2 fornece um panorama atual do estado da arte neste campo de pesquisa. A Seção 3 descreve o procedimento realizado para o desenvolvimento e análise de desempenho do modelo de classificação de tráfego DNS proposto baseado no algoritmo MLP (*Multilayer Perceptron*). A Seção 4 apresenta os resultados obtidos com a análise de desempenho do modelo e, por fim, a Seção 5 apresenta conclusões e recomendações para trabalhos futuros.

2. Trabalhos relacionados

Em [Ahmad et al. 2021] é apresentada uma revisão sistemática da literatura sobre o uso de aprendizado de máquina e aprendizado profundo no desenvolvimento de IDSs. Observou-se que a implementação de um IDS em geral segue três fases fundamentais: pré-processamento de dados, treinamento e teste. Além disso, os autores concluem que há uma tendência na utilização de técnicas de aprendizado profundo baseadas em redes neurais e que melhorias podem ser alcançadas com a seleção inteligente de *features*. Seguindo as recomendações de [Ahmad et al. 2021], nosso trabalho utiliza redes neurais e seleção inteligente de *features*.

Em [Mahdavifar et al. 2021] é proposta a classificação multiclasse de domínios (*malware*, spam e *phishing*) utilizando o dataset CIC-Bell-DNS2021, que simula um ambiente real com 99% de tráfego benigno. Entre os classificadores avaliados no estudo, que incluem SVM, KNN, MLP, Naive Bayes e Regressão Logística, o algoritmo KNN superou os modelos clássicos de aprendizado de máquina. Os algoritmos que apresentaram melhores desempenhos foram considerados em nosso trabalho.

Em [Shafi et al. 2024] é realizado um amplo estudo sobre a detecção de perfis no tráfego DNS visando identificar tráfego malicioso. Os autores avaliam as *features* mais importantes no processo de classificação do tráfego, propõem um modelo de detecção baseado em redes neurais e criam um dataset para que outros pesquisadores possam realizar trabalhos similares. Assim como os autores de [Shafi et al. 2024], a nossa proposta é voltada para detecção de tráfego DNS malicioso e utiliza redes neurais. Além disso, nós utilizamos o dataset criado. A principal diferença está no fato de que toda a implementação

realizada no nosso trabalho está disponível como software livre.

Em [Santos and Nobre 2025] é apresentado um mecanismo para mitigação de *phishing* baseado na tecnologia eBPF (*extended Berkeley Packet Filter*) e no framework XDP (*eXpress Data Path*). O mecanismo age como um *firewall* de DNS bloqueando resoluções de nomes direcionadas a domínios maliciosos. Diferente da nossa proposta, o trabalho em [Santos and Nobre 2025] baseia-se em uma lista de domínios maliciosos pré criada. A detecção de novos domínios não é realizada.

3. Metodologia

Para o desenvolvimento do modelo de redes neurais visando a detecção de domínios maliciosos, a seguinte metodologia foi seguida: (i) Escolha do dataset; (ii) Pré-processamento dos dados (tratamento de *features* categóricas, normalização de *features* numéricas e remoção de *features* com baixa variância); (iii) Decisão do modelo/arquitetura; (iv) Criação de modelos clássicos para servirem de referência na análise de desempenho; (v) Treinamento dos modelos; (vi) Teste dos modelos; e (vii) Análise de explicabilidade com a técnica SHAP proposta em [Lundberg and Lee 2017] e implementada em [Lundberg 2023].

3.1. Dataset

Após investigações do estado da arte em *datasets* que contivessem tráfego DNS, optou-se pela utilização do BCCC-CIC-Bell-DNS-2024, devido à sua modernidade e inovação quanto às *features* inéditas que melhor caracterizam o tráfego DNS, incluindo *features* lexicais, estatísticas e de fluxo. O *dataset* conta com um total de 3.595.920 amostras de tráfego DNS. Dessas amostras, 95,68% é tráfego benigno, 2,27% é tráfego malicioso utilizado para propagação de *malware*, 1,21% é tráfego malicioso voltado para *phishing* e 0,84% é tráfego malicioso utilizado para *spam*. A proporção de mais de 95% de amostras benignas pode representar um cenário real em que há mais tráfego legítimo do que malicioso, mas pode produzir resultados pouco acurados para as classes menos favorecidas proporcionalmente, como a classe *spam* por exemplo (Como a parcela de tráfego de *spam* é baixa, classificá-la como benigno não afetaria significativamente algumas métricas de qualidade do modelo). Esse aspecto foi levado em conta na interpretação dos resultados obtidos com os experimentos.

As 122 *features* presentes no *dataset* dizem respeito ao tráfego DNS combinando informações das camadas de rede e de transporte (IPs, portas, ...), informações da camada de aplicação (TTL, quantidade de registros DNS, ...), estatísticas sobre instantes de tempo, tamanhos dos pacotes e análises lexicais dos nomes de domínio. Dois exemplos de *features* deste último conjunto são a distribuição de frequência de caracteres e a entropia de Shannon de um nome de domínio (nomes de domínios criados com propósitos maliciosos tendem a possuir entropia alta).

3.2. Modelo proposto

Após uma Análise Exploratória de Dados e a leitura de trabalhos relacionados ([Ahmad et al. 2021] [Shafi et al. 2024]), optou-se pelo desenvolvimento de um modelo de aprendizado profundo baseado em MLP (*Multilayer Perceptron*), um algoritmo de rede neural artificial. Além disso, decidiu-se realizar procedimentos de seleção inteligente de

features, inicialmente com o objetivo de reduzir o tempo de execução do modelo, mas que também mostrou-se útil para mitigar vieses nas classificações.

A Figura 1 ilustra a arquitetura do modelo MLP. Ele foi implementado para classificação binária (tráfego benigno ou malicioso) e para classificação multiclasse (diferenciando entre os três tipos de ataque) com a biblioteca `TensorFlow` [TensorFlow 2024].

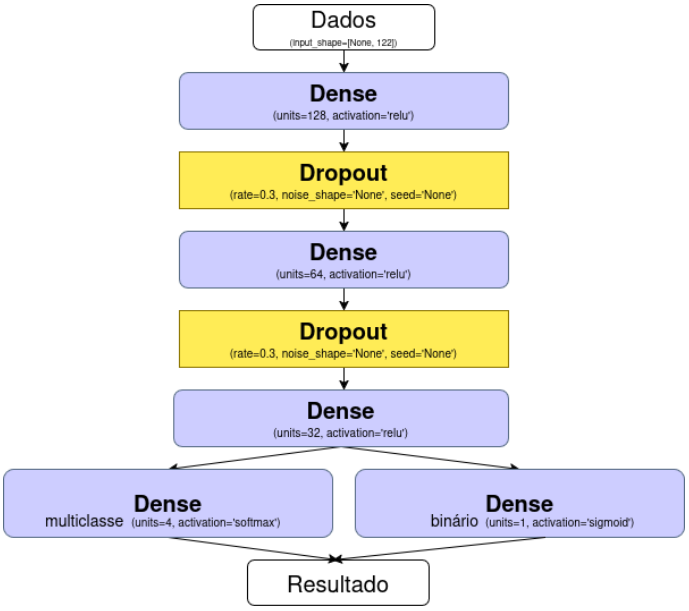

Figura 1. Arquitetura do modelo MLP.

A dimensão da camada de entrada da rede neural é o total de *features* da amostra de treinamento. As camadas densas ocultas possuem, respectivamente, as seguintes quantidades de neurônios e funções de ativação: 128, activation=‘relu’; 64, activation=‘relu’; e 32, activation=‘relu’. Nas camadas densas, cada neurônio é conectado a todos os neurônios da camada anterior para aprender padrões e ajustar os pesos e o *bias*. Entre estas camadas, foram implementadas camadas *dropout* usadas para a regularização, desativando aleatoriamente 30% dos neurônios durante o treinamento para evitar *overfitting*. A camada de saída tem dimensão 1 no caso da classificação binária e função de ativação ‘sigmoid’. Na compilação do modelo, foram escolhidas as seguintes configurações: optimizer=‘adam’ e loss=‘binary_crossentropy’. Também foi utilizada regularização com callbacks: EarlyStopping para interromper o treinamento quando não há melhorias no conjunto de dados, monitorando o valor ‘val_loss’ e ReduceLROnPlateau para reduzir o *learning rate* quando uma métrica para de melhorar, neste caso, a métrica usada é ‘val_loss’. O modelo foi treinado em 30 épocas.

A diferença para a construção do modelo multiclasse está nos seguintes aspectos: alteração do alvo para rótulo multiclasse; função de ativação da camada de saída alterada para ‘softmax’ ao invés de ‘sigmoid’; função de perda (loss) alterada para ’sparse_categorical_crossentropy’; camada de saída com dimensão 4 para cada categoria considerada.

A fim de comparar o desempenho do modelo proposto, foram implementados

dois algoritmos clássicos de aprendizado de máquina: SVM (*Support Vector Machine*) e KNN (*K-Nearest Neighbors*). A escolha desses algoritmos justifica-se pelos desempenhos superiores para este cenário de classificação em relação aos demais como se confirma nos estudos feitos em [Mahdavifar et al. 2021]. Diferentes parâmetros foram testados para cada algoritmo buscando-se analisar o comportamento e potenciais cenários de otimização classificatória, estas configurações estão descritas na Tabela 1 juntamente com os resultados alcançados em termos de acurácia, precisão, *recall* e *F1-Score*.

3.3. Explicabilidade

Muito além de resultados relacionados à acurácia, é vital entender o modelo e o que contribui para suas classificações. Por isso, a explicabilidade de cada modelo treinado foi realizada utilizando a técnica SHAP. Basicamente, o que ela faz é indicar as *features* que influenciam o modelo a tomar as decisões. Isso é feito usando teoria dos jogos para atribuir um "valor SHAP" para cada *feature*. Uma *feature* que contribui muito para a predição de um modelo tem um valor SHAP maior do que uma que contribui pouco.

A implementação da técnica SHAP em Python disponível em [Lundberg 2023] foi extensivamente utilizada durante a fase pós-experimental de análise dos resultados para entender por que o modelo prediz uma classe dentre todas as outras possíveis e observar a contribuição de cada *feature* para o modelo final decidir sobre o resultado mais provável. O processo envolveu: treinar o modelo, usar a biblioteca SHAP para computar os valores SHAP, plotar esses valores e interpretar os resultados.

3.4. Experimentos

Todas as execuções foram realizadas em um computador equipado com um processador AMD Ryzen 9 7950X 16-Core e 128GB de RAM rodando o sistema operacional Debian Forky e o interpretador Python 3.13.11. Todos os códigos desenvolvidos para realização dos experimentos foram escritos em Python e estão disponíveis como software livre em: `https://github.com/giovannahirata/IDS-DNS.git`.

Em cada um dos experimentos, foram mensurados os tempos de treinamento e teste, além das métricas tradicionais de avaliação de cada modelo: acurácia, precisão, *recall* e *F1-Score*. Por fim, o estudo de explicabilidade usando SHAP foi realizado. O treinamento de cada modelo foi realizado com 70% do dataset total.

A próxima Seção relata os resultados obtidos principalmente com dois conjuntos de experimentos: Classificação multiclasse com o dataset desbalanceado e com todas as *features* e Classificação multiclasse com o dataset balanceado sem a *feature* de porta fonte. Os resultados para classificação binária não são tão discutidos por limitação de páginas.

4. Resultados

A Tabela 1 apresenta o desempenho de todos os modelos implementados com o dataset desbalanceado e com todas as *features*. Para uma mesma métrica, os melhores resultados para o cenário binário e para o cenário multiclasse estão destacados em negrito. Nota-se que o modelo proposto obtém os melhores resultados no cenário multiclasse para todas as métricas. Não foi possível comparar o desempenho do modelo desenvolvido com aquele apresentado em [Shafi et al. 2024] pelo fato do código deste último não estar disponibilizado publicamente. A tentativa de contato com os os autores para obtenção do código não teve sucesso.

Algoritmo	Configuração	Acurácia	Precisão	Recall	F1-Score
SVM (binário)	SVC(kernel='linear', C=1)	0,9917	**1,0000**	0,8072	0,8933
SVM (multiclasse)	SVC(kernel='rbf', C=10, gamma=0.01)	0,8367	0,8394	0,8367	0,8266
KNN (binário)	n_neighbors=3	0,9979	0,9652	**0,9867**	**0,9758**
KNN (multiclasse)	n_neighbors=11	0,8925	0,8916	0,8925	0,8917
MLP (binário)	activation='sigmoid',loss='binary_crossentropy'	**0,9993**	0,8541	0,9761	0,9111
MLP (multiclasse)	activation='softmax',loss='sparse_categorical_crossentropy'	**0,9990**	**0,9993**	**0,9990**	**0,9991**

Tabela 1. Parâmetros e métricas de desempenho dos modelos.

A Figura 2 mostra os tempos que o modelo MLP levou para treino e teste levando em conta os *folds* da avaliação cruzada. O modelo final teve um tempo de treino de cerca de 17 minutos e um tempo de teste desprezível de 0,79 segundo, atestando a utilidade do modelo em um cenário real, dado que a latência da detecção de menos de 1 segundo dificilmente seria percebida por um usuário. Comparando esses resultados com o modelo SVM com a melhor configuração de parâmetros, o tempo de treinamento foi em média 69,56% menor enquanto o tempo de teste foi 58,81% menor. Por outro lado, o comportamento obtido ao comparar com o modelo KNN mostra que o tempo de treinamento foi 999900,0% maior, enquanto o tempo de teste foi 45,76% menor.

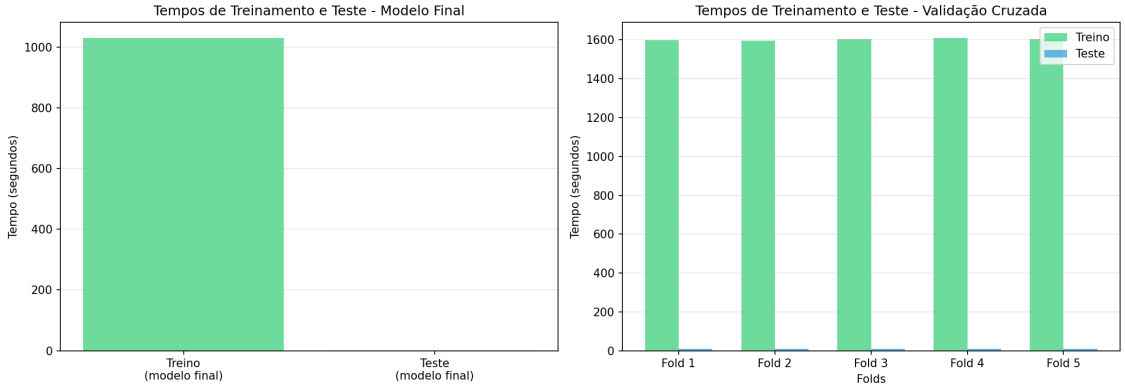

Figura 2. Tempos de treinamento e teste: MLP multiclasse.

A Figura 3 mostra a importância das *features* na classificação realizada pelo MLP de acordo com os valores SHAP. É possível perceber que a *feature* `src_port` destacou-se como a de maior importância. Esse comportamento repetiu-se na explicabilidade de todos os outros modelos, o que levou a uma investigação do dataset.

Percebeu-se que havia um padrão claro para as portas fonte do tráfego capturado. O fluxo benigno sempre tinha a porta fonte igual a 58075 ou 53. Já os fluxos classificados como *malware* e *phishing* sempre tinham a porta fonte igual a 46177, enquanto o fluxo classificado como spam sempre tinha a porta fonte igual a 58075. Com isso, a *feature* `src_port` viciou o modelo. Em um cenário real em que o atacante use outra porta, a detecção falharia. Por conta disso, essa *feature* foi removida e os experimentos refeitos. Além disso, optou-se também por balancear o tráfego.

O método para o tratamento do desbalanceamento das categorias de tráfego DNS do *dataset* consistiu em: i) escolher aleatoriamente registros do *dataset* inteiro de modo que a quantidade de benignos ficasse igual à soma de todos os maliciosos (sem remover qualquer malicioso); ii) com o dataset balanceado, aplicar a proporção 70/30 de

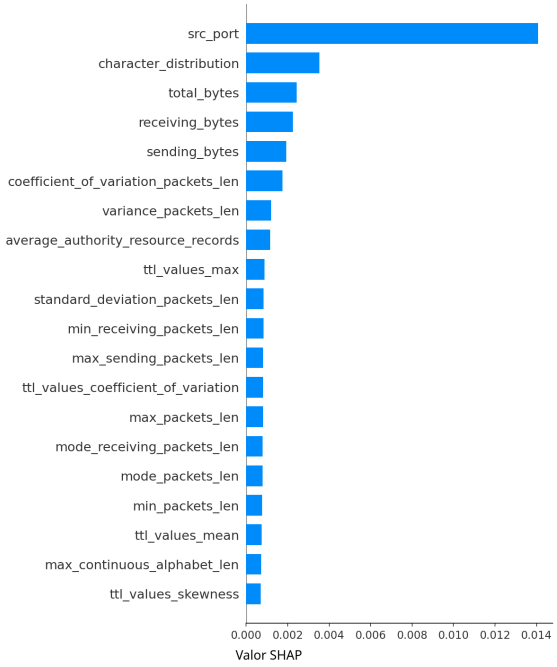

Figura 3. Avaliação do MLP multiclasse.

treino/teste; iii) repetir esse processo 10 vezes, selecionando a parte benigna de forma aleatória a cada vez para garantir 10 *subdatasets* diferentes. As Tabelas 2 e 3 apresentam o desempenho do modelo MLP, por classe, após treinamento (São apresentadas as médias das métricas e os desvios padrão considerando as 10 repetições).

Classe	Precisão média ± dp	Recall médio ± dp	F1-score médio ± dp
Benigno	0,9455 ± 0,0037	0,9535 ± 0,0049	0,9495 ± 0,0015
Malicioso	0,9532 ± 0,0046	0,9450 ± 0,0042	0,9491 ± 0,0014

Tabela 2. MLP Binário

Classe	Precisão média ± dp	Recall médio ± dp	F1-score médio ± dp
Benigno	0,9423 ± 0,0071	0,9645 ± 0,0049	0,9532 ± 0,0025
Malware	0,8199 ± 0,0140	0,8357 ± 0,0165	0,8275 ± 0,0073
Phishing	0,8637 ± 0,0227	0,7961 ± 0,0231	0,8283 ± 0,0165
Spam	0,8396 ± 0,0213	0,7864 ± 0,0241	0,8119 ± 0,0172

Tabela 3. MLP Multiclasse

O estudo de explicabilidade foi reexecutado e, observando os resultados das análises, pôde-se notar a recorrência de algumas *features* que mais contribuíram para a decisão do modelo em cada categoria de tráfego DNS. Ao destrinchar o que significam as mais relevantes, destacam-se: *character_distribution*: distribuição de frequência de cada caractere; *receiving_bytes*: *bytes* recebidos do tráfego; e *max_continuous_alphabet_len*: comprimento da maior sequência contínua de letras.

Destas, há duas referentes à análise lexical do domínio, e uma a respeito da contagem de *bytes* recebidos da comunicação. Isso contribui para definir com maior propriedade os diferentes perfis comportamentais de cada tráfego DNS, e, consequentemente, traçar o que mais marca para determinar a natureza do domínio considerando as categorias alvo de classificação nesta pesquisa.

5. Conclusões e Trabalhos Futuros

A análise do tráfego DNS pode auxiliar na detecção de ciberataques dos mais diversos tipos, como *malware*, *phishing* e spam. Este artigo apresentou um modelo baseado no algoritmo MLP para classificar o tráfego DNS nessas três classes, diferenciando-o de

tráfego benigno. O modelo é capaz de detectar tráfego benigno com F1-Score médio de 0,95 e tráfego malicioso com F1-Score médio de 0,82. Além disso, a metodologia aplicada permitiu identificar as *features* mais importantes do tráfego. Como trabalhos futuros pretende-se implementar o modelo tanto em clientes quanto em servidores DNS, observando os pontos positivos e negativos de cada abordagem. Também pretende-se utilizar técnicas que melhorem o balanceamento entre classes.

Agradecimentos

Esta pesquisa é parte do CPE SMARTNESS (FAPESP 21/00199-8). Também é suportada pelo projeto FAPESP 25/22251-2.

Referências

Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., and Ahmad, F. (2021). Network Intrusion Detection System: A Systematic Study of Machine Learning and Deep Learning Approaches. *Trans Emerging Tel Tech.*, 32(1):e4150.

Behaviour-Centric Cybersecurity Center (BCCC) (2024). Malicious DNS and Attacks (BCCC-CIC-Bell-DNS-2024) - Behaviour-Centric Cybersecurity Center (BCCC). `https://www.yorku.ca/research/bccc/ucs-technical/cybersecurity-datasets-cds/malicious-dns-and-attacks-bccc-cic-bell-dns-2024/`. Acesso em 24 de Março de 2026.

Elias, E. M. d., Carriel, V. S., De Oliveira, G. W., Dos Santos, A. L., Nogueira, M., Junior, R. H., and Batista, D. M. (2022). A Hybrid CNN-LSTM Model for IIoT Edge Privacy-Aware Intrusion Detection. In *IEEE LATINCOM*, pages 1–6.

Lundberg, S. (2023). Welcome to the SHAP documentation – SHAP latest documentation. `https://shap-community.readthedocs.io/en/latest/`. Acesso em 24 de Março de 2026.

Lundberg, S. M. and Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems*, volume 30.

Mahdavifar, S., Maleki, N., Lashkari, A. H., Broda, M., and Razavi, A. H. (2021). Classifying Malicious Domains using DNS Traffic Analysis. In *IEEE DASC/PiCom/CBDCom/CyberSciTech*, pages 60–67.

Santos, P. and Nobre, J. (2025). Uma Abordagem para Mitigação de Phishing utilizando eBPF/XDP. In *Anais do XXV SBSeg*, pages 888–904.

Shafi, M., Lashkari, A. H., and Mohanty, H. (2024). Unveiling Malicious DNS Behavior Profiling and Generating Benchmark Dataset through Application Layer Traffic Analysis. *Computers and Electrical Engineering*, 118:109436.

TensorFlow (2024). TensorFlow. `https://www.tensorflow.org/`. Acesso em 24 de Março de 2026.