UNIVERSIDADE ESTADUAL PAULISTA - UNESP

Instituto de Biociências, Letras e Ciências Exatas - Campus de São José do Rio Preto

GUILHERME ROMANHOLO BOFO

Detecção de Domínios Maliciosos em Tráfego de DNS Passivo Utilizando Fontes de Inteligência de Ameaças e Aprendizado de Máquina

São José do Rio Preto

2025

Guilherme Romanholo Bofo

Detecção de Domínios Maliciosos em Tráfego de DNS Passivo Utilizando Fontes de Inteligência de Ameaças e Aprendizado de Máquina

Trabalho de Conclusão de Curso, apresentada à Universidade Estadual Paulista (UNESP), Instituto de Biociências, Letras e Ciências Exatas, São José do Rio Preto, para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof Dr. Adriano Mauro Cansian

São José do Rio Preto
2025

B673d

Bofo, Guilherme Romanholo
Detecção de domínios maliciosos em tráfego de DNS passivo utilizando fontes de inteligência de ameaças e aprendizado de máquina / Guilherme Romanholo Bofo. -- São José do Rio Preto, 2025
65 f. : il., tabs.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (UNESP), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto
Orientador: Adriano Mauro Cansian

1. Ciência da computação. 2. Segurança cibernética. 3. Machine learning. 4. Nomes de domínio na Internet. 5. Fraude na Internet. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

GUILHERME ROMANHOLO BOFO

DETECÇÃO DE DOMÍNIOS MALICIOSOS EM TRÁFEGO DE DNS PASSIVO UTILIZANDO FONTES DE INTELIGÊNCIA DE AMEAÇAS E APRENDIZADO DE MÁQUINA

Trabalho de Conclusão de Curso apresentado(a) à Universidade Estadual Paulista (UNESP), Instituto de Biociências, Letras e Ciências Exatas, São José do Rio Preto, para obtenção do título de Bacharel em Ciência da Computação.

Data de defesa: 05/11/2025

BANCA EXAMINADORA

Prof Dr. Adriano Mauro Cansian
UNESP – Instituto de Biociências, Letras e Ciências Exatas – Campus de São José do Rio Preto

---

Prof Dr. Arnaldo Cândido Júnior
UNESP – Instituto de Biociências, Letras e Ciências Exatas – Campus de São José do Rio Preto

---

Profa Dra. Rogéria Cristiane Gratão de Souza
UNESP – Instituto de Biociências, Letras e Ciências Exatas – Campus de São José do Rio Preto

Dedico este trabalho à todas crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.

AGRADECIMENTOS

Agradeço, primeiramente, à Deus por toda a minha vida, sempre me auxiliando a vencer todos os obstáculos ao longo da realização desse trabalho e sempre me acompanhando ao trilhar o caminho descrito por Ele.

Agradeço aos meu pais Sandra Gilene Romanholo Bofo e Vanderlei Donizeti Bofo, além do meu irmão Arthur Romanholo Bofo pelo apoio que foi concedido durante toda a minha formação, por sempre estarem presentes em minha vida e acreditarem em mim.

Agradeço à minha namorada e companheira para a vida, Gabriela Sousa Louzano, por todo amor, carinho, presença, e por todo apoio nos momentos em que eu mais precisei, sempre me motivando a continuar seguindo os meus sonhos e sendo essencial para cada linha escrita neste trabalho.

Agradeço ao meu amigo e irmão, Filippo de Oliveira Barbosa, por me acompanhar nessa trajetória desde o ensino fundamental, sempre fornencendo conselhos valiosos para o meu desenvolvimento pessoal e profissional.

Agradeço ao meu orientador Prof. Dr. Adriano Mauro Cansian por todos os ensinamentos concedidos e pela grande contribuição em minha formação acadêmica e pessoal, além da oportunidade em fazer parte do laboratório ACME!, o qual foi um grande diferencial para minha graduação.

Agradeço aos membros do laboratório ACME!, em especial o "*Team DNS*" composto por Gabriele, João Rafael, Kim e Matheus, por todos conhecimentos e cooperação, os quais me permitiram concluir este trabalho.

O presente trabalho foi realizado no âmbito do Projeto de Pesquisa e Inovação Tecnológica "Aprimoramento na Qualidade de Registro de Domínios Via Detecção Precoce de Conduta Abusiva - Fase 2", realizado com o apoio do Núcleo de Informação e Coordenação do Ponto BR (NIC.br) - Processo FUNDUNESP No. 3467/2023 (1º. Termo Aditivo 2025 - 2027).

"Security is a chain; it's only as secure as the weakest link."
(Schneier, 2004)

RESUMO

O *Domain Name System* (DNS) é um protocolo essencial para o funcionamento da Internet. No entanto, devido ao seu amplo alcance e presença na vida dos usuários, também se tornou um alvo frequente de exploração por agentes maliciosos em ataques como *phishing*, *spam*, distribuição de *malwares* e *Command-and-Control* (C2). As abordagens tradicionais de detecção baseadas exclusivamente em listas de bloqueio apresentam limitações significativas na identificação precoce de novas ameaças, devido ao tempo necessário para catalogação dos domínios maliciosos. Este trabalho propõe o desenvolvimento de um sistema modular para detecção de domínios maliciosos por meio da análise de tráfego de DNS passivo em uma rede de computadores de produção. A abordagem híbrida combina fontes de inteligência de ameaças com técnicas de aprendizado de máquina, utilizando o algoritmo *Random Forest* treinado com características léxicas, de DNS ativo e *Term Frequency-Inverse Document Frequency* (TF-IDF). O sistema foi validado com dados reais do servidor recursivo da Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP), que processa aproximadamente 23 milhões de consultas diárias. O modelo proposto alcançou AUC de 0.918 e F1-score de 0.81 em dados de teste, demonstrando capacidade de detecção precoce de ameaças não catalogadas em listas de bloqueio. Durante oito meses de monitoramento em ambiente real, o sistema identificou entre 622 e 829 domínios maliciosos mensalmente, com especificidade superior a 0.94. Os resultados são disponibilizados por meio de uma *Application Programming Interface* (API) integrada a dashboards visuais no *Framework* DNS, fornecendo aos analistas de segurança informações enriquecidas com dados de geolocalização, *Autonomous Systems* (AS) e origem das ameaças, permitindo análises forenses detalhadas e tomada de decisões fundamentadas em múltiplas camadas de proteção.

Palavras-Chave: cibersegurança; DNS passivo; inteligência de ameaças; domínios maliciosos; aprendizado de máquina.

ABSTRACT

The Domain Name System (DNS) is a fundamental protocol for the functioning of the Internet. However, due to its broad reach and presence in users' daily lives, it has also become a frequent target of exploitation by malicious actors and abuse in attacks such as phishing, spam, malware distribution, and Command-and-Control (C2). Traditional detection approaches based exclusively on blocklists present significant limitations in the early identification of emerging threats, due to the time required for malicious domains to be cataloged. This work proposes the development of a modular system for detecting malicious domains through the analysis of passive DNS traffic in a production computer network. The hybrid approach combines threat intelligence sources (Hagezi TIF, ThreatFox, and DGA Bambenek Consulting) with machine learning techniques, utilizing the Random Forest algorithm trained with lexical features, active DNS features, and Term Frequency-Inverse Document Frequency (TF-IDF). The system was validated with real data from UNESP's recursive server, which processes approximately 23 million queries daily. The proposed model achieved an AUC of 0.918 and F1-score of 0.81 on test data, demonstrating the ability to detect emerging threats not cataloged in blocklists. During eight months of monitoring in a real environment, the system identified between 622 and 829 malicious domains monthly, with specificity exceeding 0.94. The results are made available through a REST API integrated with visual dashboards in the DNS Framework, providing security analysts with enriched information including geolocation data, Autonomous Systems (AS), and threat origins, enabling detailed forensic analysis and decision-making based on multiple layers of protection.

Keywords: cybersecurity; passive DNS; threat intelligence; malicious domains; machine learning.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
AS	Autonomous System
ASCII	American Standard Code for Information Interchange
ASN	Autonomous System Number
AUC	Area Under the Curve
C2	Command-and-Control
ccTLD	country code Top-Level Domain
CTI	Cyber Threat Intelligence
DGA	Domain Generation Algorithm
DNS	Domain Name System
DoH	DNS Over HTTPs
DP	Desvio Padrão
DT	Decision Tree
ENTRADA	ENhanced Top-level domain Resilience through Advanced Data Analysis
FN	Falso Negativo
FP	Falso Positivo
FQDN	Fully Qualified Domain Name
gTLD	generic Top-Level Domain
IOC	Indicator of Compromise
IP	Internet Protocol
JSON	JavaScript Object Notation
ML	Machine Learning
MVC	Model-View-Controller
NIC.br	Núcleo de Informação e Coordenação do Ponto BR

ROC	Receiver Operating Characteristic
RR	Resource Record
TF-IDF	Term Frequency-Inverse Document Frequency
TIF	Threat Intelligence Feeds
TLD	Top-Level Domain
TTL	Time to Live
UNESP	Universidade Estadual Paulista “Júlio de Mesquita Filho”
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VP	Verdadeiro Positivo
VN	Verdadeiro Negativo

SUMÁRIO

1 INTRODUÇÃO

1.1 CONSIDERAÇÕES INICIAIS

O DNS (*Domain Name System*) é um mecanismo responsável por nomear recursos de uma forma em que possam ser utilizados em diferentes dispositivos (Mockapetris, 1987). Segundo os dados observados na atualidade, o Brasil ultrapassou a marca de 5 milhões de domínios registrados e continuará em um constante crescimento (NIC.br, 2024). Entretanto, em conjunto à esse grande crescimento, também surgem brechas para a utilização dos nomes de domínios em atividades maliciosas, como *phishing*, distribuição de *malwares* e *typosquatting*. Dessa forma, tornou-se essencial a detecção de domínios utilizados para finalidades maliciosas, visando uma Internet mais segura.

Tradicionalmente a detecção de domínios maliciosos é realizada por meio de listas de bloqueio, as quais são disponibilizadas e atualizadas diariamente com domínios maliciosos identificados por fontes confiáveis. No entanto, embora seja um método muito eficiente e ainda amplamente utilizado, é possível observar uma limitação na identificação e bloqueio precoce de novas ameaças, dado o tempo considerável para inserção de novas ameaças nas listas.

Com o advento da inteligência artificial, surgiram novos métodos capazes de identificar de maneira precoce domínios maliciosos, utilizando métricas de DNS ativo (Kountouras et al., 2016) e métricas de DNS passivo (Silveira et al., 2021), resolvendo a limitação observada nas listas de bloqueio. No entanto, nessa abordagem surge um problema relacionado à garantia de acertos do modelo, visto que não é possível garantir a sua totalidade de acertos.

1.2 JUSTIFICATIVA

Com base no que foi apresentado, identificou-se a necessidade do estudo sobre uma abordagem híbrida entre os métodos tradicionais e de ML, realizando uma compensação entre seus pontos fortes e fracos, a fim de corroborar para um método mais eficiente na detecção de domínios maliciosos.

Assim, por meio das técnicas de ML, é possível observar uma detecção precoce de domínios maliciosos, sem que haja a necessidade do tempo de espera referente à adição de domínios em listas de bloqueio. Por outro lado, a utilização dos métodos tradicionais garantem um *ground truth* sólido e resiliente para as detecções realizadas pelos modelos de ML.

Diante disso, a proposta desse trabalho é realizar a análise do tráfego de DNS passivo (Weimer, 2005) em uma rede de computadores, utilizando a abordagem híbrida descrita anteriormente, a fim de avaliar o desempenho da combinação entre esses métodos apresentados.

1.3 OBJETIVOS

Este trabalho tem como por objetivo geral o desenvolvimento de um sistema capaz de analisar tráfego de DNS passivo em uma rede de computadores, realizando a combinação de métodos de ML e listas de bloqueio para a detecção de domínios maliciosos, fornecendo as informações de maneira organizada e apresentável para administradores de rede e analistas de segurança. Esse objetivo principal foi divido da seguinte forma:

1. Desenvolvimento de uma API (*Application Programming Interface*) responsável por monitorar o tráfego de DNS passivo, fazendo a coleta dos dados de maneira programada durante o dia;

2. Detecção de domínios maliciosos observados no DNS passivo por meio da verificação em listas de bloqueio;

3. Adição de métodos de ML combinando métricas textuais e ativas de DNS para a identificação precoce de possíveis ameaças indicadas pelo modelo;

4. Enriquecimento das ameaças observadas na rede, coletando informações importantes para CTI (*Cyber Threat Intelligence*) como: IP de origem, IP de destino, ASN (*Autonomous System Number*) e outras informações relevantes;

5. Desenvolvimento de uma interface gráfica para a visualização dos dados referentes as ameaças identificadas pelo sistema;

6. Validação do sistema realizando uma análise em uma rede de computadores em mundo real, fora do ambiente controlado de laboratório.

1.4 ORGANIZAÇÃO DA MONOGRAFIA

Esta monografia foi dividida em cinco capítulos distintos, contanto com o atual. O Capítulo 2 é referente a fundamentação teórica, explicando as tecnologias e os conceitos teóricos utilizados no desenvolvimento deste trabalho. No Capítulo 3 será abordada a metologia do projeto, passando desde a etapa da coleta dos dados de DNS passivo até a representação dos resultados. No Capítulo 4 serão discutidos os testes e resultados obtidos do sistema proposto. Finalmente, no Capítulo 5 serão apresentadas as conclusões, desafios encontrados e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 PROTOCOLO DNS

Embora programas possam referenciar máquinas, serviços de *e-mail* e outros recursos por meio de endereços de IP, a memorização para os usuários acaba se tornando pouco intuitiva, devido a grande quantidade de endereços existentes na Internet. Além disso, é muito comum que endereços sejam constantemente trocados devido à migrações de serviços, alterações na infraestrutura ou decisões políticas, necessitando que os usuários fossem constantemente atualizados sobre qualquer mínima atualização.

Dessa forma, foram introduzidos nomes ASCII com o objetivo de desacoplar os endereços das máquinas (Tanenbaum, 2002). Nos primórdios da Internet os nomes eram armazenados em um arquivo denominado *hosts.txt*, responsável por realizar o mapeamento local para endereços de IP. Entretanto, com o crescimento da Internet, surgiu a necessidade da criação de um sistema distribuído e escalável, capaz suportar um grande volume de dados e evitar conflitos na atualização dos mapeamentos, surgindo assim o DNS.

2.1.1 Hierarquia DNS

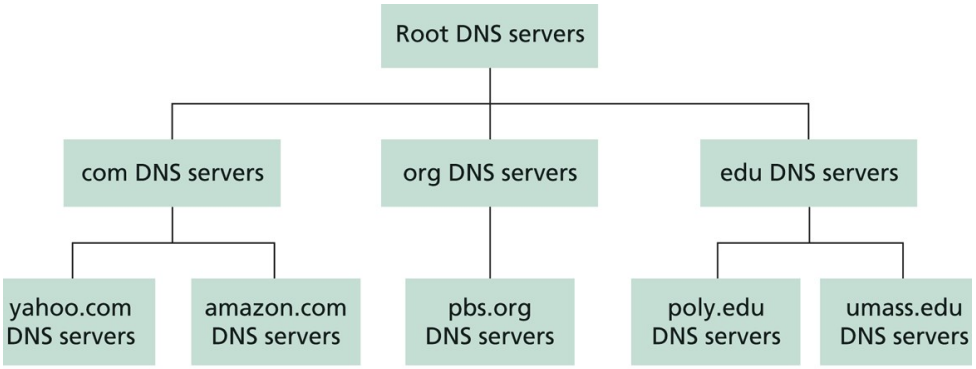
Para garantir a escalabilidade, o DNS utiliza um grande número de servidores distribuídos de forma hierárquica ao redor do mundo, garantindo que nenhum servidor possua todos os mapeamentos de nomes de domínio para endereços IP. Dessa forma, como pode ser observado na Figura 1, os servidores de DNS são divididos da seguinte forma: servidores raiz, servidores TLD (*Top-Level Domain*) e servidores autoritativos.

Os servidores de DNS raiz (*root servers*) são encontrados no topo da hierarquia. Na Internet existem 13 desses servidores que são nomeados de A a M, os quais possuem replicas para fins de segurança e confiabilidade. Seguindo a hierarquia, esses servidores serão os primeiros a serem consultados, tendo a função de retornar uma resposta sobre quais são os servidores TLD que respondem pelo domínio consultado.

Descendo na hierarquia são encontrados os servidores TLD, os quais podem ser divididos em gTLD (*generic Top-Level Domain*) e ccTLD (*country code Top-Level Domain*). Eles são responsáveis por domínios de alto nível, como *com, org, edu, net e gov* representando domínios genéricos (gTLD) e *br, fr, jp e ru* que representam domínios de países (ccTLD).

Por fim, na base da hierarquia estão localizados os servidores de DNS autoritativos. Esses servidores são pertencentes a qualquer organização ou empresa que possua dispositivos que precisem ser conhecidos na Internet, mapeando os nomes desses dispositivos para endereços IP. Por exemplo, um servidor autoritativo que responde pelo domínio *unesp.br* deverá conhecer a resolução dos endereços dentro de sua zona de autoridade, por exemplo *www.unesp.br* ou *ibilce.unesp.br*.

Figura 1 – Hierarquia dos servidores de DNS

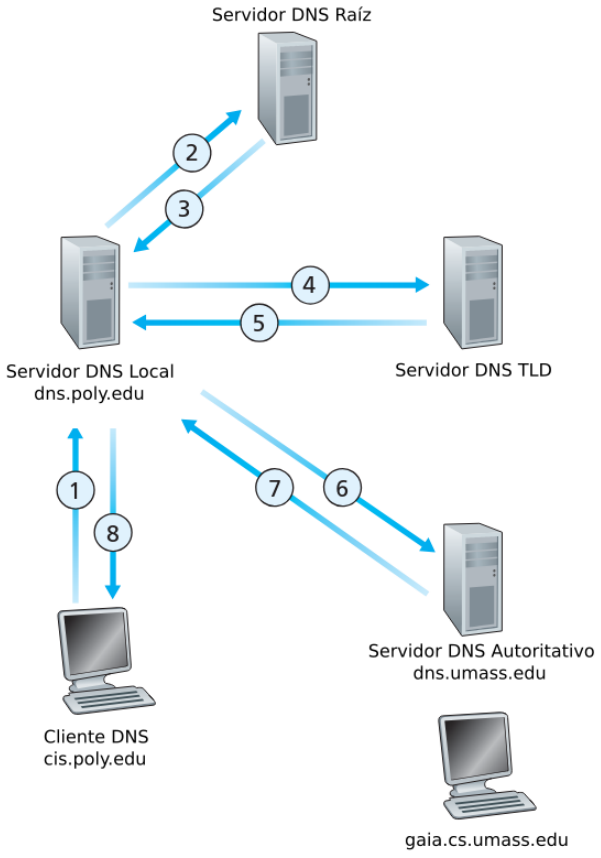


Fonte: Kurose & Ross (2012).

2.1.2 Resolução DNS

Com base no que foi comentado, o processo de resolução de domínios consiste em um passo a passo, seguindo a hierarquia dos servidores, para conversão de um nome de domínio para um endereço IP. É possível observar esse processo de resolução conforme a Figura 2.

Figura 2 – Resolução DNS



Fonte: Kurose & Ross (2012).

1. O *host* cliente (*cis.poly.edu*) deseja realizar a resolução do domínio *gaia.cs.umass.edu*. Para isso, ele realiza uma consulta para o seu servidor DNS local;

2. O servidor DNS local realiza uma requisição para o topo da hierarquia, o servidor DNS raiz, afim de obter os servidores TLD responsáveis por *edu*;

3. O servidor raiz envia a resposta ao recursivo com uma lista dos servidores TLD responsáveis por *edu*;

4. O servidor DNS local realiza a consulta para o próximo nível da hierarquia, o servidor TLD que responde por *edu*, para obter o servidores autoritativos responsáveis por *umass.edu*;

5. O servidor TLD envia de volta ao recursivo uma resposta com o endereço do servidor autoritativo responsável por *umass.edu*;

6. O servidor DNS local pergunta para o autoritativo, situado no último nível da hierarquia, qual o endereço de *gaia.cs.umass.edu*;

7. O servidor autoritativo responde para o recursivo local o endereço de *gaia.cs.umass.edu*;

8. O servidor recursivo local responde para o cliente o endereço referente ao domínio consultado no início do processo (*gaia.cs.umass.edu*) e armazena a resposta em cache para resoluções futuras.

2.1.3 *Resource Records*

Embora a principal funcionalidade do DNS seja determinar os endereços IP que correspondem à um determinado nome, o protocolo também pode ser utilizado para o propósito oposto, além de oferecer suporte a diversas outras funcionalidades (Stevens, 1993).

Nesse contexto, os RRs (*Resource Records*) são a unidade básica de informação na base de dados do DNS. Cada RR possui um tipo específico, identificado no cabeçalho do pacote, indicando o conteúda da resposta. Os principais tipos de RRs podem ser observados na Tabela 1.

Tabela 1 – Tipos de *Resource Records*

Tipo	Valor	Descrição e Propósito
A	1	Endereço IPv4
NS	2	Nome do servidor de DNS da zona
CNAME	5	Mapeia um nome para o outro (apelido)
SOA	6	Informações da zona do domínio
PTR	12	Mapeamento reverso de IP para nome
MX	15	Nome do servidor de *e-mail* associado ao domínio
AAAA	28	Endereço IPv6

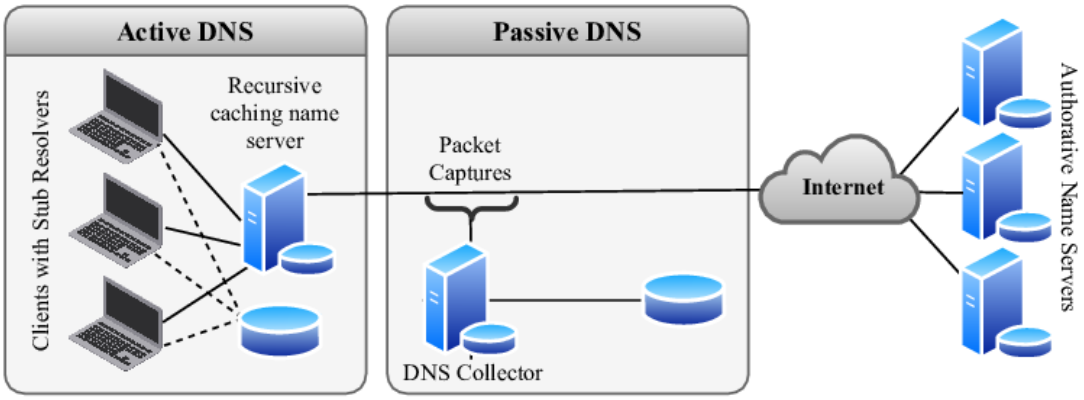
Fonte: Adaptado de Stevens (1993).

2.1.4 DNS Passivo

O conceito de DNS passivo, proposto pela primeira vez em Weimer (2005), consiste em um sistema de monitoramento de consultas DNS por meio da coleta e armazenamento das consultas realizadas por um servidor recursivo. A principal motivação proposta pelo autor consiste na necessidade de mitigar ataques de *malwares* que incorporam domínios em seu código-fonte, possibilitando a identificação e bloqueio do ataque em uma rede de computadores.

Dessa forma, a principal diferença entre o conceito de DNS passivo e ativo está no dispositivo que realiza a consulta. Como pode ser observado na Figura 3, o DNS ativo está associado aos dispositivos que realizaram a consulta para o servidor recursivo, enquanto o conceito de DNS passivo está no registro das consultas, realizadas por outros *hosts* da rede, por meio de um coletor que efetivamente não realiza as consultas.

Figura 3 – DNS Passivo x Ativo



Fonte: Pour et al. (2023).

O DNS passivo possui algumas vantagens no processo de identificação de ameaças. Entre as principais vantagens destaca-se o armazenamento do histórico de resoluções, permitindo observar como determinados domínios foram resolvidos ao longo do tempo e quais seus endereços IP associados. Além disso, o armazenamento das consultas possibilita a realização de investigações forenses mais detalhadas, viabilizando o mapeamento entre IPs de origem e os domínios acessados, o que auxilia na análise e na compreensão de incidentes de segurança.

2.2 FONTES DE INTELIGÊNCIA DE AMEAÇAS

Inteligência de Ameaças, ou *Threat Intelligence*, pode ser definida como o processo de coleta, análise e exploração de dados para compreensão de ataques cibernéticos, auxiliando analistas de segurança na identificação de vetores de ataque, táticas e atacantes.

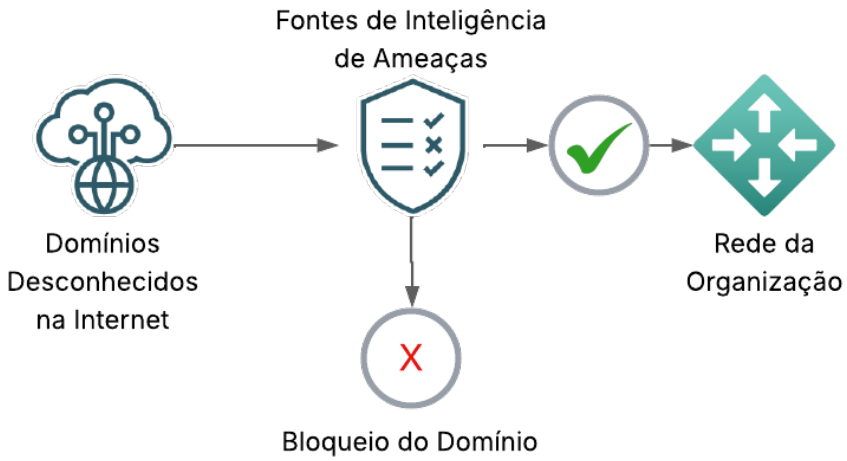
Dentre os principais métodos de inteligência de ameaças está a utilização de fontes de inteligência de ameaças, as quais podem ser descritas como um conjunto de IOCs (*Indicators of Compromise*) como IPs, domínios, URLs entre outros dados utilizados para identificar potenciais atividades maliciosas em uma rede de computadores.

Com o decorrer do tempo, um maior número de analistas de segurança tem percebido a importância no monitoramento do tráfego DNS, tornando a utilização de fontes de inteligência de ameaças de domínios mais comuns no mercado (Liska; Stowe, 2016).

2.2.1 Proteção por Fontes de Inteligência

Como pode ser observado na Figura 4, o processo de proteção utilizando fontes de inteligência, com nomes de domínios como IOC, ocorre por meio da filtragem de domínios desconhecidos que são acessados pelos *hosts* na rede da organização. Caso o domínio seja identificado em uma das fontes utilizadas ele sofre o bloqueio imediado, impedindo que a rede seja comprometida por qualquer atividade maliciosa. Vale ressaltar a existência de listas de permissão (*allowlists*), as quais realizam o processo inverso, permitindo apenas que os domínios na lista sejam acessados dentro da organização.

Figura 4 – Proteção por fontes de inteligência de ameaças



Fonte: Elaborado pelo autor.

2.2.2 Exemplos de Fontes Relevantes

Atualmente, as fontes de inteligência mais relevantes no mercado podem ser divididas em duas principais categorias: comerciais e de código-aberto. As fontes comerciais são fornecidas por empresas de segurança aos seus clientes, oferecendo em conjunto suporte técnico e garantindo a validação de todos os IOCs. Por outro lado, as fontes de código-aberto normalmente são mantidas pela comunidade ou instituições de pesquisa, garantindo uma atualização contínua em seus indicadores, por mais que a qualidade possa variar entre as fontes.

Para o desenvolvimento deste trabalho, optou-se por utilizar fontes de código-aberto, garantindo sua reprodutibilidade. Dentre as listas mais relevantes dessa categoria é possível observar:

- *Hagezi* TIF: É um conjunto de várias fontes de inteligência de ameaças de código-aberto, disponibilizada e atualizada em um repositório público no GitHub (Hagezi, 2025). A TIF utiliza domínios como IOC, sendo amplamente utilizada para bloquear ameaças conhecidas como *malwares*, *phishing*, *spam* e C2 (*Command-and-Control*) que utilizam nomes de domínios fraudulentos. Suas atualizações são feitas diariamente pela comunidade e mantenedores do repositório, tendo seu tamanho constantemente modificado com inserções e remoções;

- *ThreatFox*: É um projeto código-aberto no qual analistas de seguranças podem compartilhar IOCs identificados com a comunidade de maneira gratuita. Atualmente alguns dos IOCs disponíveis são domínios, IPS, endereços de *e-mail*, C2 e *malwares* (abuse.ch; Spamhaus, 2021). Um dos diferenciais dessa fonte está na associação de IOCs com famílias de *malwares*, facilitando a identificação de ataques;

- DGA *Bambenek Consulting*: É uma fonte de código-aberto mantida por uma empresa privada. Seu principal IOC são nomes de domínios, especificamente DGAs (*Domain Generation Algorithms*) relacionados com ataques de *malwares* e suas famílias (Consulting, 2025). A lista conta com aproximadamente 53 famílias distintas de *malwares*, com dados provenientes da análise de redes maliciosas.

2.3 APRENDIZADO DE MÁQUINA

Segundo Russell & Norvig (2021), um agente está em processo de aprendizado quando ele melhora sua performance em tarefas futuras após realizar observações sobre o mundo. Assim, com a avanço da tecnologia surgiu uma área denominada Machine Learning, responsável por estudar modelos computacionais com a capacidade de observar e identificar padrões em dados do mundo real.

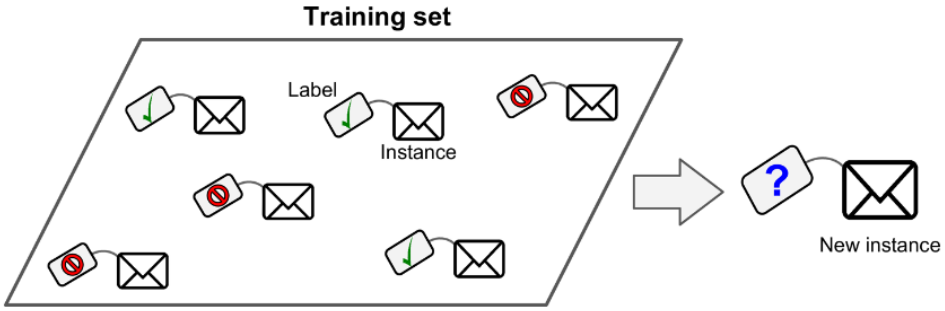
Em linhas gerais, esse aprendizado pode ser realizado de diferentes maneiras, as quais são divididas em quatro principais categorias: aprendizado supervisionado, aprendizado não supervisionado, aprendizado semi-supervisionado e aprendizado por reforço (Aurélien, 2019).

Dentre essas categorias, a utilizada nesse trabalho foi o aprendizado supervisionado, dado sua alta relevância no contexto de classificação de domínios maliciosos conforme foi abordado nos trabalhos correlatos. Além disso, é possível obter domínios rotulados de maneira conveniente, graças as listas de bloqueio apresentadas. Nessa categoria, o modelo observa os pares de características (entrada) e rótulo (saída) durante o processo de aprendizado, associando características que não foram observadas anteriormente à um rótulo por meio da correlação estabelecida no processo de treinamento.

Um exemplo pode ser observado na Figura 5, no qual o modelo é treinado para classificar mensagens de *e-mail* como *spam* ou não *spam*, por meio do treinamento com o texto dos *e-mails* e seus respectivos rótulos. Por fim, após o treinamento, quando forem entregues novas

características de um novo *e-mail* para o modelo, ele possuirá a capacidade de predizer um rótulo apropriado com base no que foi observado durante a etapa de treinamento.

Figura 5 – Aprendizado supervisionado na classificação de *spam*



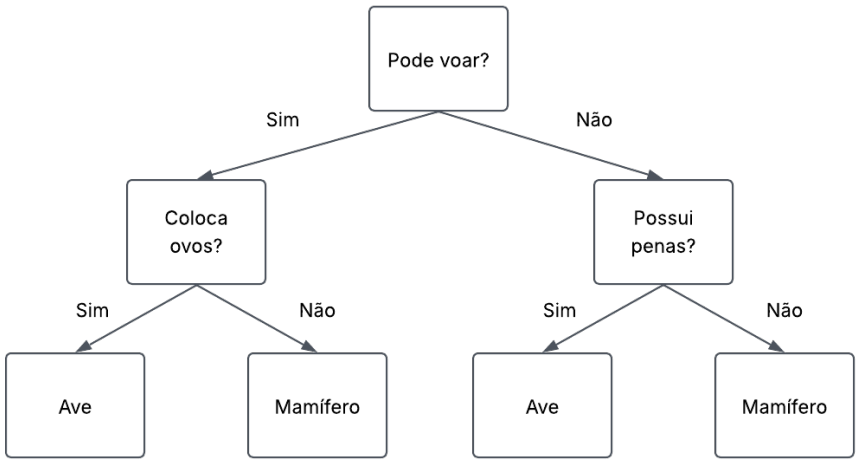
Fonte: Aurélien (2019).

2.3.1 Árvore de Decisão

Árvores de Decisão, do inglês *Decision Trees* (DTs), são modelos de aprendizado supervisionado não paramétricos, ou seja, se adaptam dinamicamente aos dados sem assumir um número limitado de parâmetros (Pedregosa et al., 2011).

Na Figura 6 é possível observar um exemplo de DT para a classificação entre as classes *Ave* e *Mamífero*. Por meio da figura é possível observar perguntas com respostas "sim" e "não", que fazem o modelo percorrer a raiz da árvore até um nó folha, onde será determinado o rótulo para a nova instância de dados.

Figura 6 – Exemplo de Árvore de Decisão



Fonte: Elaborado pelo autor.

Assim, para realizar a predição, as DTs utilizam um processo de inferência por indução (Quinlan, 1996), onde cada pergunta do processo indutivo representa um nó da árvore, descendo

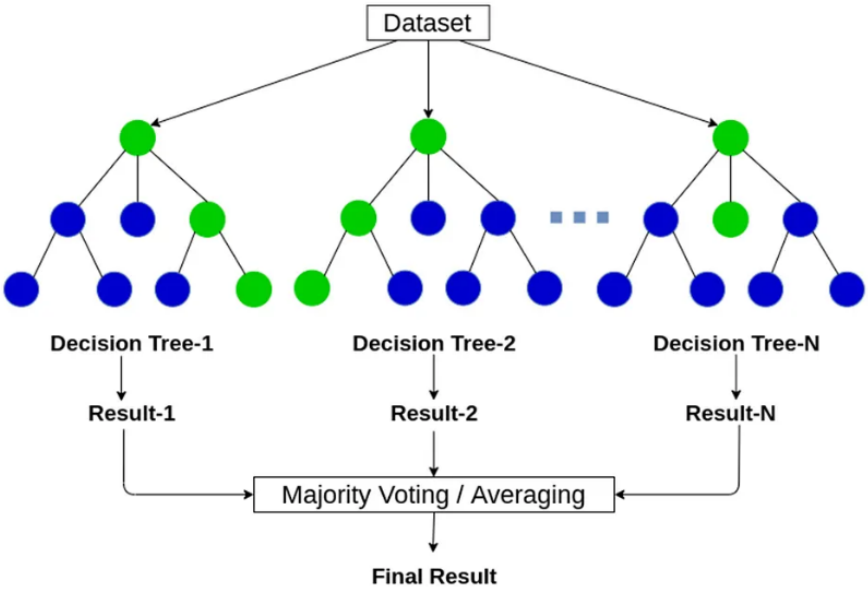
para os nós filhos de maneira hierárquica até chegar ao rótulo inferido, representado por um nó folha. Vale ressaltar que a principal estratégia adotada pelo algoritmo para chegar em um resultado consiste na divisão do espaço definido pelos atributos em "subespaços" cada vez menores.

2.3.2 Floresta Aleatória

O algoritmo de Floresta Aleatória é considerado um método *ensemble*, ou seja, é resultado da combinação entre vários outros modelos para obter um melhor desempenho na classificação. A ideia desse algoritmo, como o próprio nome diz, consiste na criação de uma "floresta" a partir da combinação entre várias árvores de decisão.

Como pode ser observado na Figura 7, um *dataset* é subdivido para as N árvores utilizadas pelo modelo, no qual cada árvore retornará um resultado para cada instância de dados, que serão combinados por meio de uma votação ou média como resultado final do modelo.

Figura 7 – Exemplo de uma Floresta Aleatória



Fonte: Omarzai (2024).

Um ponto crucial que garante o melhor desempenho do modelo está relacionado ao processo denominado *Bagging*, também conhecido como *Bootstrap-Aggregation*, que ocorre na etapa de divisão do *dataset*. A principal ideia deste conceito está na seleção aleatória dos *subsets* de treinamento, os quais serão selecionados com partes diferentes e duplicadas, gerando diversidade entre o conjunto de árvores. Dessa forma, são criados N modelos que serão treinados usando essas amostras de maneira combinada, afim de gerar a média ou a votação das previsões para a como resultado final (Salman; Kalakech; Steiti, 2024).

2.3.3 Métricas de Avaliação

Após o treinamento dos modelos de ML, é essencial que seja efetuada uma avaliação do seu desempenho em dados teste, a fim de compreender se o modelo está realizando a classificação de maneira correta. Para isso, são utilizadas as métricas de avaliação, como a acurácia, precisão, sensibilidade, AUC (*Area Under the Curve*), curva ROC (*Receiver Operating Characteristic*), matriz de confusão, entre outras medidas representativas para a quantidade de acertos e erros de um modelo sobre os dados de teste.

Na classificação binária existem dois tipos de classe, por outro lado em classificação multi-classe existem mais de duas possibilidades. Dessa forma, todas as métricas avaliativas podem ser adaptadas para a utilização em problemas multi-classe (Zheng, 2015), no entanto, não serão abordados detalhes sobre problemas multi-classe, visto que o foco deste trabalho é na utilização de classificadores binários.

A Figura 8 representa uma matriz de confusão, onde é possível observar de maneira detalhada a distribuição das classificações corretas e incorretas.

Figura 8 – Exemplo de matriz de confusão

		Detectada	
		Sim	**Não**
Real	**Sim**	Verdadeiro Positivo (VP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (VN)

Fonte: Soni (2024).

Observando na primeira linha, os VPs (Verdadeiro Positivos) indicam os dados da classe positiva que também foram classificados da classe positiva, os FNs (Falso Negativos) indicam os dados da classe positiva que foram erroneamente classificados como pertencentes a classe negativa. Já na segunda linha da matriz, os FPs (Falso Positivos) são os dados da classe negativa que foram classificados como dados da classe positiva de maneira equivocada, e por fim, os VNs (Verdadeiro Negativos), indicam os elementos da classe negativa que foram corretamente classificados. Além disso, é importante notar que a diagonal principal mostra os acertos do modelo, enquanto na diagonal secundária encontram-se os seus erros.

Com base nessa nomenclatura descrita pela matriz de confusão, é possível calcular outras métricas de avaliação do modelo. A acurácia, como consta na Equação 1, avalia qual a frequência em que o modelo realiza predições corretas em relação ao total de predições.

$$Acuracia = \frac{VP + VN}{VP + VN + FP + FN} \tag{1}$$

A Equação 2 descreve o cálculo da precisão, responsável por indicar quantas classes positivas foram corretamente preditas dentro de todas as predições positivas. Em contrapartida, na Equação 3 é possível observar o cálculo da sensibilidade, a qual indica a quantidade de predições da classe positiva dentre todos os casos positivos reais. Por fim, na Equação 4 é possível observar a especificidade, representando capacidade do modelo em identificar corretamente os casos negativos, ou seja, quando a predição corresponde a exemplos que realmente não pertencem à classe positiva.

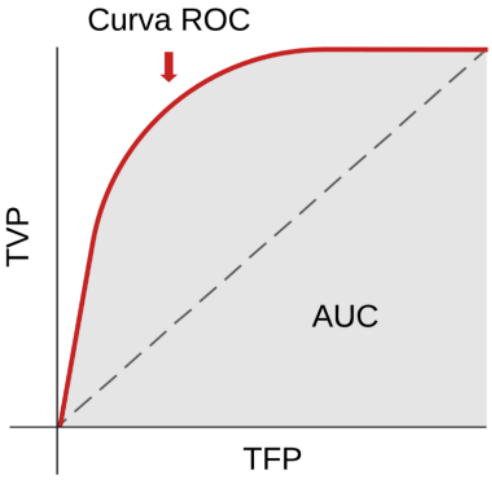
$$Precisao = \frac{VP}{VP + FP} \tag{2}$$

$$Sensibilidade = \frac{VP}{VP + FN} \tag{3}$$

$$Especificidade = \frac{VN}{VN + FP} \tag{4}$$

Outra métrica de avaliação importante é a curva ROC, a qual mostra quão bem um modelo é capaz de distinguir entre duas classes, composta pela taxa de falso positivo (TFP) no eixo x, e a taxa de verdadeiro positivo (TVP) no eixo y. Em conjunto com a curva ROC pode ser observada AUC, medida que varia entre 0.0 e 1.0 e é obtida por meio da cálculo da área abaixo da curva, sendo utilizada na medição do desempenho do modelo.

Figura 9 – Curva ROC e AUC



Fonte: Silva (2022).

Na Figura 9 é possível observar o esboço de uma curva ROC, tal como os valores observados nos seus eixos e o cálculo da AUC por meio da sua área. Também vale ressaltar a linha pontilhada, indicando o limiar de um modelo aleatório, representado pela AUC com o valor de 0.5.

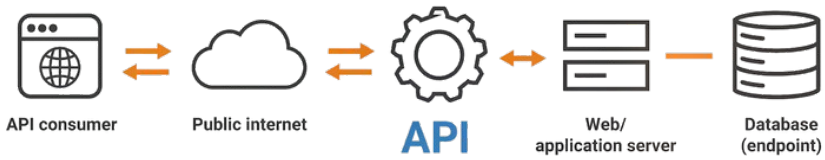
2.4 *APPLICATION PROGRAMMING INTERFACE* (API)

Uma API pode ser definida por um conjunto de regras e protocolos que permitem diferentes aplicações se comunicarem umas com as outras, facilitando a troca de dados, compartilhamento de funcionalidades e integração entre sistemas (Akamai, 2025). Dessa forma, uma API que utiliza o protocolo HTTP/HTTPs para realizar a comunicação pela rede pode ser chamada de *Web Service*.

Conforme pode ser observado na Figura 10, o processo de consulta à uma API funciona da seguinte maneira:

1. O cliente (API *Consumer*) realiza uma requisição para o *endpoint* que está rodando a API;
2. A requisição tráfega pela rede pública até chegar na API consultada (camada de aplicação);
3. Após receber a conexão, a API realiza o processamento da consulta e envia a requisição para o servidor da aplicação local;
4. O servidor da aplicação local realiza o que foi solicitado, por exemplo, uma escrita ou leitura em um banco de dados;
5. Por fim, as informações solicitadas seguem o caminho oposto, chegando com a resposta até o cliente que realizou a solicitação.

Figura 10 – Funcionamento de uma API



Fonte: Akamai (2025).

A principal padronização no formato de envio e recebimento dos dados por uma API é o JSON (*JavaSript Object Notation*). O JSON segue um princípio de pares "chave-valor", baseado na ideia de mapas/dicionários (estrutura de dados), facilitando a identificação do significado dos dados recebidos, sejam *strings*, números ou até mesmo valores *booleanos*.

Além disso, outro conceito muito importante são as RESTful API's. O termo REST é um acrônimo para *REpresentational State Transfer*, considerado um tipo de arquitetura utilizado para sistemas distribuídos de hipermídia, descrito pela primeira vez em Fielding (2000). Esse estilo de arquitetura de *software* possui seis grandes princípios que garantem escalabilidade e simplicidade, os quais são:

1. Interface Uniforme: Esse princípio refere-se a padronização da interface da API, utilizando por exemplo os métodos padrões do protocolo HTTP (GET, POST, etc) e URIs (*Uniform Resource Identifiers*) para identificar os recursos;

2. Cliente-Servidor: Esse princípio prevê uma separação de conceitos dos componentes do cliente (interface) dos componentes do servidor (banco de dados) aumentando a escalabilidade e modularização do sistema;

3. *Stateless*: Esse princípio prevê que toda requisição realizada pelo cliente deve conter todas as informações necessárias para o sistema, não guardando estado de conexão;

4. *Cacheable*: Esse princípio garante que as respostas das requisições devem indicar se a resposta pode ser armazenada em cachê para simplificar requisições futuras;

5. Sistema de Camadas: Esse princípio prevê que o sistema deve ser composto por camadas hierárquicas, permitindo a comunicação apenas por meio de camadas intermediárias, e evitando acoplamento desnecessário, como a utilização do padrão MVC (*Model-View-Controller*);

6. Código sob Demanda (Opcional): Esse princípio mostra que o servidor pode ter a capacidade de fornecer trechos de código executáveis para o cliente, por exemplo para a distribuição de novas *features*.

2.5 TRABALHOS CORRELATOS

O estudo em torno da detecção de domínios maliciosos é um tema com alto número de pesquisas, dada a sua significância para o cenário global de segurança de computadores. Foi utilizado como base para esse projeto o conceito de *Passive* DNS (Weimer, 2005) como principal fonte de dados de domínios legítimos e maliciosos. Dessa forma, é possível listar alguns trabalhos que utilizam métodos similares aos quais serão adotados neste trabalho.

No trabalho proposto por Ghafir & Prenosil (2015) é realizado o processamento de tráfego DNS na rede do campus de sua universidade, onde os domínios acessados são extraídos dos *pcaps*[1] coletados e verificados em uma lista de bloqueio periodicamente atualizada. Assim, quando ocorre um *match* entre o domínio coletado e a lista de bloqueio, é disparado um *e-mail* para os administradores da rede relatando o incidente.

Em contrapartida, no trabalho proposto por Marques, Malta & Magalhães (2021) é construída uma ideia de *firewall* baseado em técnicas aprendizado de máquina, responsável por filtrar consultas à domínios maliciosos, diferindo-se da abordagem convencional de utilização de listas de bloqueio. Dessa forma, foi realizado um estudo comparativo entre algoritmos de aprendizado de máquina para descobrir qual possui o melhor desempenho para a construção do *firewall* proposto, chegando como conclusão ao algoritmo CART com 95.6% de acurácia.

[1] Formato de arquivo que armazena capturas de tráfego de rede de maneira padronizada.

Nas pesquisas realizadas em Gomes (2018) e Pompeu (2018), também foram utilizadas técnicas de *machine learning* baseadas, respectivamente, em características textuais e de coleta ativa para a detecção de domínios maliciosos. Em ambos os trabalhos foram realizados estudos comparativos entre diversos algoritmos da literatura, tendo destaque nos resultados obtidos pelos algoritmos *Multilayer Perceptron* e *Random Forest*, com AUCs, respectivamente, de 85.93% e 92.6%.

No trabalho proposto por Gregório et al. (2024), é possível observar a utilização de técnicas mais sofisticadas, como redes neurais convolucionais, para a detecção de domínios maliciosos. Nesse caso, o foco da pesquisa foi na identificação de DGAs (*Domain Generation Algorithm*) utilizados por *botnets*. Dessa forma, a pesquisa obteve resultados muito expressivos, obtendo acurácias de 99.12% combinando técnicas de *Embedding*, ReLU e *MaxPooling* em suas redes neurais.

Em Akiba et al. (2019) foi desenvolvido um *framework* utilizado para a otimização de hiperparâmetros em modelos de aprendizado de máquina de forma automática, com base em conceitos de MLOps. O sistema foi desenvolvido de forma modular, permitindo a adição de novos modelos e reduzindo a necessidade de intervenção humana, de forma que todo o sistema fosse gerido de forma automática.

No trabalho proposto por Wullink et al. (2016) foi desenvolvido um sistema denominado ENTRADA (*ENhanced Top-level domain Resilience through Advanced Data Analysis*), utilizado para o processamento de um grande volume de dados de DNS. O processamento é realizado em cima de *pcaps* e dividido em etapas de limpeza e enriquecimentos dos dados brutos, transformando-os em um formato denominado *parquet*. Assim, os dados processados são inseridos em um banco de dados de alto desempenho a fim de permitir consultas complexas e rápidas em grandes volumes de dados.

Por fim, em Batista (2020) foi desenvolvido o *Framework* DNS, um sistema capaz de gerenciar modelos de *machine learning* de forma automática, além de integrar um banco de dados distribuído com dados de tráfego de DNS. Esse trabalho mostrou-se eficiente na utilização de classificadores, criando um *pipeline* capaz de re-treinar os modelos e checar sua acurácia sem a alteração de seu código-fonte. Além disso, o trabalho também obteve uma ótima eficiência na escalabilidade, disponibilidade e compactação dos dados de DNS, reduzindo em até 95% o tamanho dos arquivos coletados.

3 METODOLOGIA

3.1 ARQUITETURA DO SISTEMA

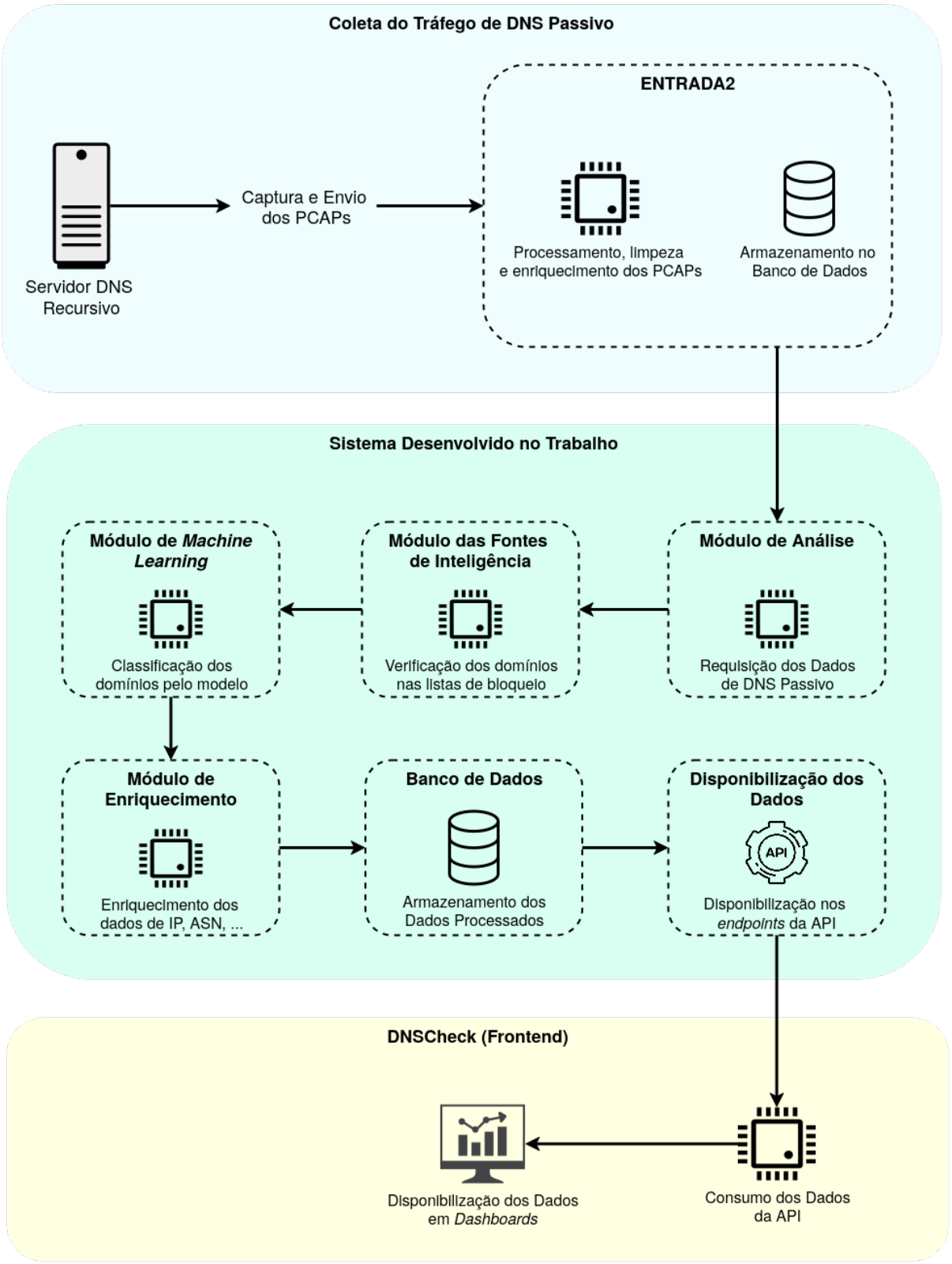
O objetivo principal deste trabalho é o desenvolvimento de um sistema modular responsável por coletar, analisar, processar e disponibilizar dados referentes a atividade maliciosa em DNS passivo. Em linhas gerais, os dados serão coletados a partir de uma fonte de DNS passivo, processados por modelos de aprendizado de máquina e listas de bloqueio, enriquecidos com informações importantes de CTI e disponibilizados para analistas em *dashboards* e APIs. Cada uma dessas etapas será abordada com um maior número de detalhes no decorrer deste capítulo.

Na Figura 11 pode ser observada a arquitetura do sistema descrito e as suas partes: o sistema de coleta, processamento e visualização dos dados. Também é possível observar o fluxo da informação, desde a coleta em um servidor recursivo de DNS até a visualização final em um *dashboard*. Ainda com base na arquitetura apresentada, o sistema foi subdivido nos seguintes módulos:

- Módulo de Análise: É responsável por realizar a requisição automática e periódica no banco de dados distribuído com dados de DNS do ENTRADA2 [1];
- Módulo das Fontes de Inteligência: É responsável por gerenciar as fontes de inteligência de ameaças para fazer a validação dos domínios;
- Módulo de *Machine Learning*: É responsável por gerenciar os modelos que realizam a classificação dos domínios;
- Módulo de Enriquecimento: É responsável por enriquecer os dados das ameaças identificadas, obtendo informações como IPs de origem, IPs de destino, ASN, entre outros.

[1] Versão atualizada do ENTRADA (Wullink et al., 2016), capaz de utilizar mecanismos modernos como *buckets* S3 para o armazenamento distribuído dos dados.

Figura 11 – Arquitetura do sistema



Fonte: Elaborado pelo autor.

3.2 COLETA DOS DADOS DE DNS PASSIVO

A coleta dos dados que serão enviados para a análise no sistema é o ponto inicial da arquitetura. Para isso, foram utilizados dados de DNS passivo provenientes do servidor recursivo localizado na Reitoria da UNESP (Universidade Estadual Paulista "Júlio de Mesquita Filho").

A captura dos pacotes das consultas de DNS que são resolvidos pelo recursivo é realizada no próprio servidor por meio da ferramenta *dnscap*. Após a captura, são armazenadas no formato *pcap* e enviadas pela rede para o servidor que hospeda o ENTRADA2, o qual será responsável pelo processamento e enriquecimento dos dados brutos, além do armazenamento no formato *Parquet*[2].

Como o servidor recursivo atende todos os campi da universidade, os dados são enviados em um intervalo de tempo menor (em torno de 10 minutos), dado o seu grande volume de consultas diárias. Para compreender a proporção dos dados, é possível observar na Tabela 2 a quantidade média diária de consultas de DNS, domínios distintos e FQDNs (*Fully Qualified Domain Name*) distintos.

Tabela 2 – Quantidade média diária do tráfego capturado pelo servidor recursivo

Dado	Quantidade
Domínio	30.000
FQDN	250.000
Consultas	23.100.000

Fonte: Elaborado pelo autor.

Vale ressaltar que o *dnscap* é uma ferramenta desenvolvida pela DNS-OARC (*Domain Name System Operations Analysis and Research Center*) para captura de pacotes em múltiplas interfaces de rede, projetada especificamente para dados de DNS. Essa ferramenta possui uma gama de utilitários e otimizações para o tráfego DNS, permitindo ajustar e customizar a coleta conforme a necessidade do projeto.

3.3 GERENCIAMENTO DAS FONTES DE INTELIGÊNCIA

Como foi discutido na fundamentação teórica do trabalho, fontes de inteligência são essenciais para garantir a segurança de sistemas. Para a aplicação no trabalho, as fontes de inteligência escolhidas foram as listas de bloqueio, as quais possuem como IOC nomes de domínios maliciosos.

[2] *Parquet* é um formato para armazenamento de dados otimizado para as ferramentas do ecossistema *Apache*.

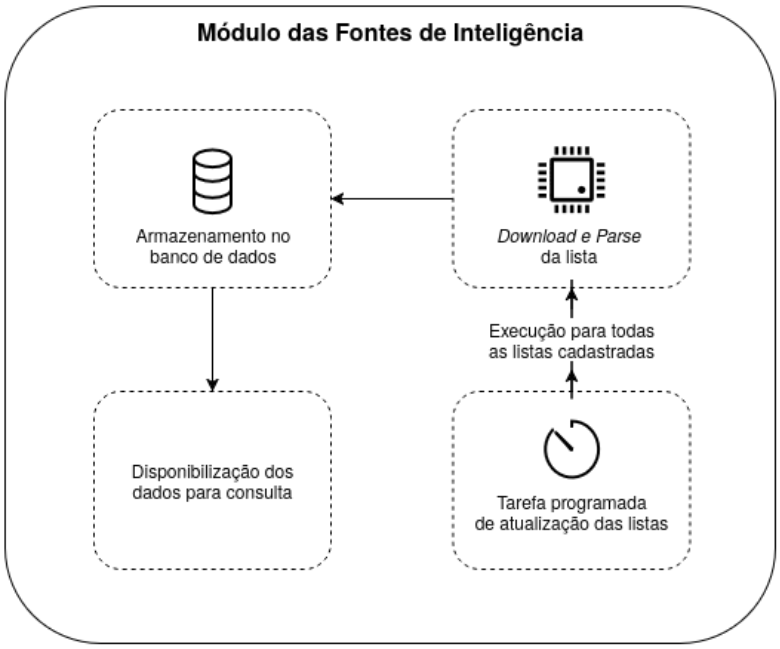
As listas utilizadas nesse trabalho foram: *Hagezi* TIF, *ThreatFox* e DGA *Bambenek Consulting*. Vale ressaltar que no período de desenvolvimento deste trabalho todas as listas são de código-aberto, ou seja, estão disponíveis à qualquer usuário para *download* e utilização.

3.3.1 Arquitetura do Módulo de Fontes de Inteligência

O gerenciamento das listas de bloqueio no sistema são realizados pelo Módulo das Fontes de Inteligência, localizado dentro da arquitetura do sistema desenvolvido, conforme a Figura 11. As informações mais detalhadas acerca da arquitetura do módulo podem ser observadas na Figura 12.

Em linhas gerais, esse módulo é responsável por realizar a atualização programada e diária das listas de bloqueio com uma tarefa agendada para às 00:30 horas, garantindo que os dados estejam constantemente atualizados. Após isso, os dados são armazenados no banco e disponibilizados para a consulta, sendo utilizados como filtro para os domínios provenientes do DNS passivo.

Figura 12 – Arquitetura do Módulo de Fontes de Inteligência



Fonte: Elaborado pelo autor.

3.3.2 Metadados das Fontes de Inteligência

As informações descritas na Tabela 3 referem-se aos metadados utilizados pelo sistema para manipular as listas de bloqueio. Essas informações são atualizadas quando a lista sofre a atualização diária dos seus dados ou quando o administrador do sistema deseja realizar alguma alteração em um campo específico.

Tabela 3 – Metadados de uma fonte de inteligência

Dado	Descrição
Parser	Método utilizado para fazer o *parsing* dos domínios da lista
Fonte	URL (*Uniform Resource Locator*) da lista
Data de criação	Data de criação da lista no sistema
Data de atualização	Data da última atualização da lista pelo sistema

Fonte: Elaborado pelo autor.

3.4 GERENCIAMENTO DO CLASSIFICADOR

O gerenciamento do classificador utilizado pelo sistema é realizado pelo Módulo de *Machine Learning*. Seu principal objetivo é automatizar o *pipeline* de treinamento do classificador, mantendo-o sempre disponível para a classificação dos domínios provenientes do módulo de coleta.

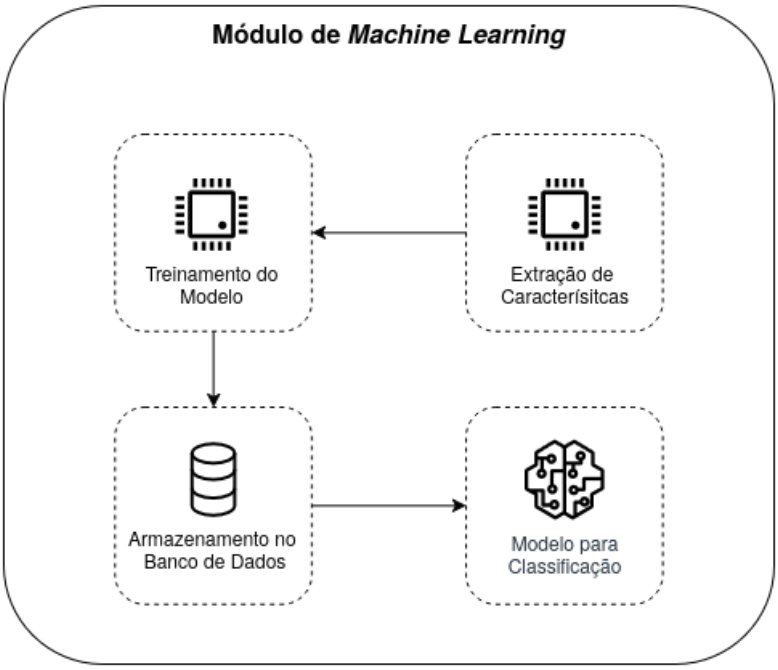
O modelo escolhido foi o *Random Forest*, devido aos seus resultados promissores observados nos trabalhos propostos em Pompeu (2018), Gomes (2018) e Marques, Malta & Magalhães (2021), já discutidos na Seção 2.5. Além disso, a abordagem escolhida está na utilização de características léxicas e de DNS ativo, de maneira combinada, para o treinamento do modelo proposto.

3.4.1 Arquitetura do Módulo de *Machine Learning*

A ideia do módulo de ML é garantir o gerenciamento automático do modelo, passando pelas etapas de extração de características, treinamento do modelo, armazenamento do modelo treinado e disponibilização para classificação, conforme o *pipeline* descrito na Figura 13.

A única etapa manual será a submissão das listas por parte do administrador do sistema. Após esse processo, o modelo cumprirá o *pipeline* proposto de maneira automatizada ficará disponível para a classificação dos domínios. Também é possível manter múltiplos modelos treinados e disponíveis para a utilização, além da escolha de um modelo padrão para analisar os dados proveniente do tráfego de DNS.

Figura 13 – Arquitetura do Módulo de *Machine Learning*



Fonte: Elaborado pelo autor.

3.4.2 Obtenção dos Dados de Treinamento

Os domínios utilizados para o treinamento do modelo proposto foram coletados de duas fontes de dados, uma utilizada para a classe de domínios maliciosos e outra para a classe benigna.

Para os domínios benignos foi utilizada a *Majestic Million*, uma lista pública e gratuita com um milhão de domínios mais acessados do mundo, considerando como métrica principal de avaliação o número de *subnets* que fazem referência a cada domínio. Por outro lado, para a classe maliciosa foi utilizada a lista *Hagezi* TIF[3] discutida previamente na Seção 2.2.2.

Para prosseguir para a extração de características, foi coletada uma amostra aleatória de 100.000 domínios de cada lista, devido ao alto processamento necessário para extrair as características e treinar o modelo com a totalidade dos dados das listas.

3.4.3 Extração de Características

Durante a etapa de extração de características foram utilizadas métricas léxicas e de DNS ativo. O processo de extração de características léxicas foi realizado por meio da linguagem Python, enquanto a extração das características de DNS ativo foi realizada por meio de requisições ao DoH (*DNS Over HTTPs*) da *Cloudflare*[4]. O processo de extração pode ser observado na

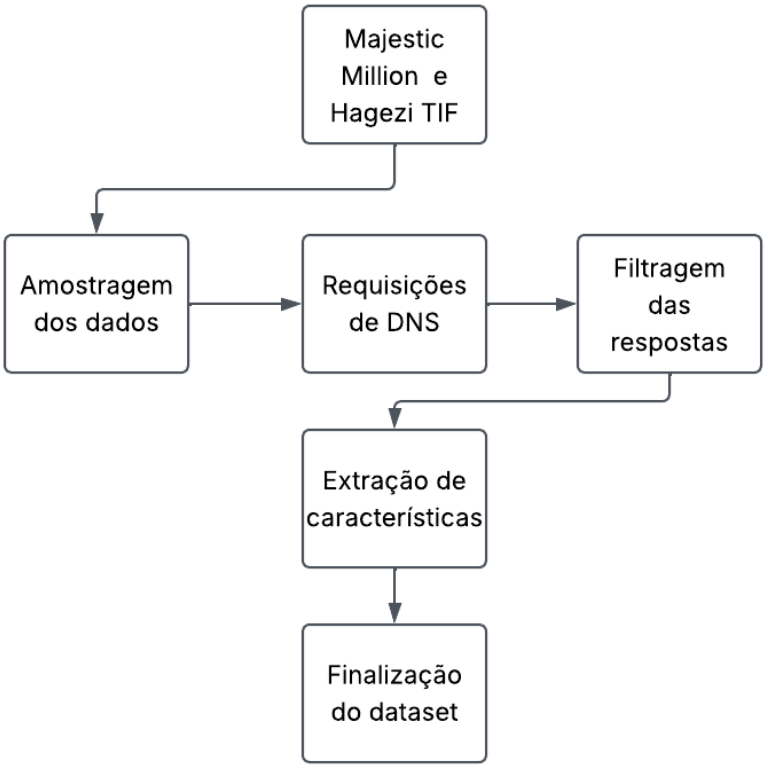
[3] A lista *Hagezi* TIF possui um tamanho variável devido as suas constantes inserções e remoções de domínios.
[4] https://cloudflare-dns.com

Figura 14 e os detalhes sobre cada característica utilizada pode ser observado na Tabela 4.

As métricas léxicas são focadas principalmente em características do nome do domínio, como a quantidade de números ou de caracteres especiais. A justificativa na utilização dessas características está relacionada à caracteres incomuns ou aleatórios, por exemplo os DGAs (*Domain Generation Algorithm*), utilizados principalmente para a criação de vários domínios por algoritmo com o objetivo de evadir detecções.

Enquanto isso, as métricas de DNS ativo são provenientes dos RR's associados ao domínio. Uma característica muito observada em domínios maliciosos é a resolução de endereços IPs com valores baixos de TTL (*Time to Live*), utilizado para dificultar a inclusão de domínios em listas de bloqueio (Bilge et al., 2014). Outra característica, por exemplo, está na quantidade de servidores de *e-mail* associados à um domínio, onde um maior número é observado em domínios associados à campanhas de *spam* e *phishing* (Sperotto; Toorn; Rijswijk-Deij, 2017).

Figura 14 – Processo de extração de características



Fonte: Elaborado pelo autor.

Tabela 4 – Características utilizadas pelo modelo

Característica	Descrição	Tipo
IP Count	Número de IPv4 e IPv6 associados	ASN
ASN Count	Quantidade de ASNs distintos associados	
A Count	Número de respostas para o *record* A	DNS
NS Count	Número de respostas para o *record* NS	
MX Count	Número de respostas para o *record* MX	
AAAA Count	Número de respostas para o *record* AAAA	
CNAME Count	Número de respostas para o *record* CNAME	
SOA Default TTL	TTL do padrão do *record* SOA	
MX Medium TTL	TTL médio entre os *records* MX	
AAAA Medium TTL	TTL médio entre os *records* AAAA	
A/CNAME Stdev TTL	DP entre o TTL dos *records* A e CNAME	
A/CNAME Medium TTL	TTL médio entre os *records* A e CNAME	
SOA Retry	Valor do campo *retry* no *record* SOA	
SOA Refresh	Valor do campo *refresh* no *record* SOA	
SOA Minimum	Valor do campo *minimum* no *record* SOA	
SOA Expire Length	Tamanho do campo *expire* no *record* SOA	
SOA Serial Length	Tamanho do campo *serial* no *record* SOA	
Lenght	Tamanho do domínio	Léxico
Entropy	Entropia de Shannon	
Dot Count	Quantidade de pontos no domínio	
Trace Count	Quantidade de traços no domínio	
Vowel Count	Quantidade de vogais no domínio	
Number Count	Quantidade de números no domínio	
Consonant Count	Quantidade de consoantes no domínio	
Vowel Consec	Quantidade de vogais consecutivas	
Number Consec	Quantidade de números consecutivos	
Consonant Consec	Quantidade de consoantes consecutivas	
Size Max Word	Tamanho da maior palavra	

Fonte: Elaborado pelo autor.

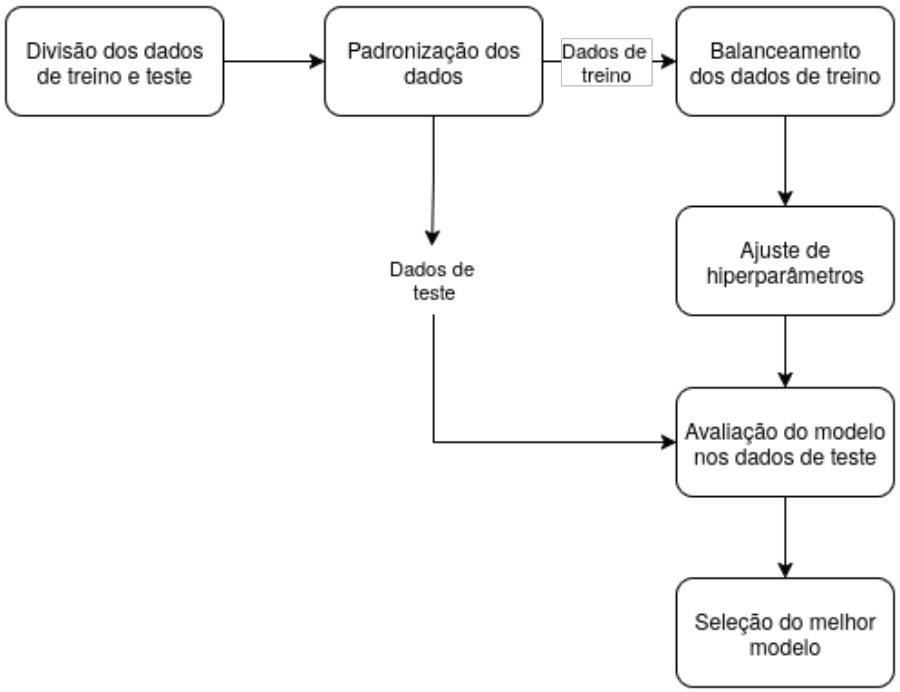
3.4.4 Treinamento e *Fine-Tunning*

Após a extração de características foi realizado o processo de treinamento e ajuste fino do modelo, o qual foi subdividido em algumas etapas conforme pode ser observado na Figura 15.

O primeiro passo do treinamento consiste na divisão da base de dados, a qual foi subdividida em treino e teste, utilizando a função *train_test_split* da biblioteca *scikit-learn* (Pedregosa et al., 2011) com a semente de aleatoriedade 42. A proporção da divisão foi 70% para dados de treinamento e 30% para dados de teste.

A seguir, foi realizada a padronização dos dados com a técnica de *Standardization*. A ideia é evitar que os dados estejam em escalas muito discrepantes, transformando-os de modo que possuam média igual à zero e desvio padrão igual à um. Vale ressaltar que o escalonador foi ajustado com os dados de treinamento e aplicado em ambos os conjuntos (treino e teste).

Figura 15 – *Pipeline* de treinamento do modelo



Fonte: Elaborado pelo autor.

Após a padronização, observou-se a necessidade de balanceamento dos dados, visto que vários domínios foram filtrados na base de dados original por não possuírem resposta de DNS para a extração das métricas ativas. Dessa forma, optou-se pela técnica de *Random Under-Sampling*, pegando uma amostra reduzida da classe majoritária para equilibrar a proporção entre classes.

Além do uso exclusivo das características numéricas, foram também realizados experimentos com a adição de novas características textuais do domínio por meio da técnica *TF-IDF (Term Frequency-Inverse Document Frequency) Vectorizer*. O objetivo foi verificar o impacto dessa técnica no desempenho do modelo, além da comparação com a abordagem baseada apenas nas características propostas na Tabela 4.

O ajuste dos hiperparâmetros do modelo foi realizado usando a técnica de *Randomized Search*, realizando uma busca aleatória no espaço de hiperparâmetros definidos na Tabela 5. Optou-se pela utilização dessa técnica devido ao grande espaço de busca, limitando para 150 iterações aleatórias. Por fim, o modelo foi avaliado na base de dados de teste e comparado com o modelo *baseline* (com os hiperparâmetros padrões).

Tabela 5 – Hiperparâmetros utilizados no ajuste fino do modelo

Hiperparâmetro	Espaço de busca
Max. Depth	$\{x \in Z \mid 10 \leq x \leq 50\}$
Max. Features	*sqrt*, *log2*, None
Min. Samples Leaf	$\{x \in Z \mid 1 \leq x \leq 5\}$
Min. Samples Split	$\{x \in Z \mid 2 \leq x \leq 10\}$
Number of Estimators	$\{x \in Z \mid 100 \leq x \leq 300\}$

Fonte: Elaborado pelo autor.

3.5 IDENTIFICAÇÃO DE AMEAÇAS

Como foi representado na Figura 11, a identificação de ameaças nos dados de DNS passivo é realizada por meio de uma abordagem combinada entre os módulos de Fontes de Inteligência e de *Machine Learning*, conforme foram descritos, respectivamente, nas seções 3.3 e 3.4.

O Módulo de Análise realiza a requisição para o banco de dados distribuído do ENTRADA2 por meio de uma tarefa agendada de hora em hora, para obter os dados referentes as novas consultas do dia. O *pipeline* para a requisição dos dados e identificação das ameaças possui a seguinte ordem:

1. Requisição dos dados de DNS passivo do ENTRADA2 no formato (*domain_name*, *count*), onde *count* é a quantidade de vezes que o domínio foi acessado no dia da consulta;

2. Classificação dos domínios observados no tráfego de DNS pelo modelo descrito na Seção 3.4. Durante esse processo as características dos domínios são extraídas e enviadas para o modelo realizar a classificação, guardando os domínios potencialmente maliciosos, com *score* de classificação maior que 0.5;

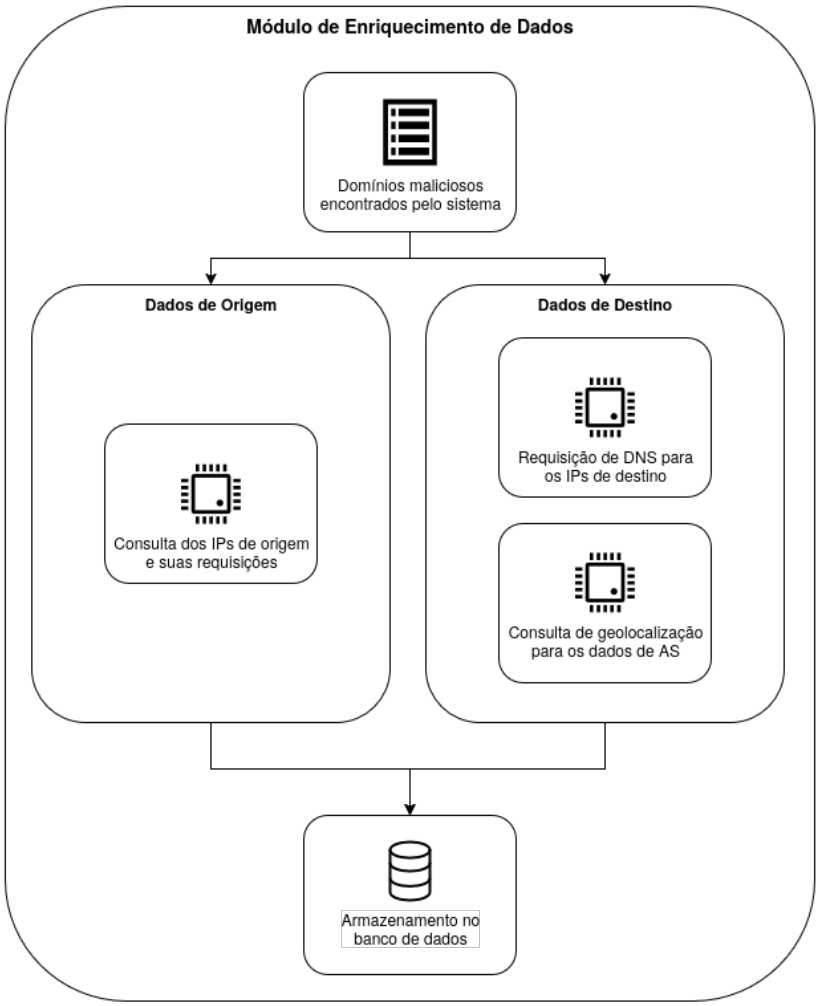
3. Os domínios são buscados nas listas de bloqueio disponibilizadas pelo módulo apresentado na Seção 3.3 e associados com seus dados de classificação. Caso existam domínios maliciosos não identificados pelo classificador, o sistema realiza a marcação como ameaça identificada apenas pelas listas de bloqueio;

4. Por fim, esses domínios identificados são enviados para o módulo de enriquecimento, onde serão coletadas informações adicionais importantes para uma análise mais detalhada.

3.6 ENRIQUECIMENTO DOS DADOS

O Módulo de Enriquecimento dos Dados é responsável por coletar informações adicionais e associá-las aos domínios maliciosos identificados, traçando um panorama geral sobre o possível ataque identificado. A arquitetura desse módulo pode ser observada na Figura 16, indicando como ocorre a coleta desses dados adicionais.

Figura 16 – Módulo de enriquecimento de dados



Fonte: Elaborado pelo autor.

Os domínios maliciosos identificados por listas de bloqueio e métodos de ML são enviados ao módulo responsável, onde passam por duas etapas de processamento para o enriquecimento dos dados: identificação da origem e do destino do ataque.

As informações relacionadas à origem do ataque são extraídas do banco de dados de DNS passivo gerenciado pelo ENTRADA2. A partir da combinação dos nomes dos domínios maliciosos com a data associada ao ataque, é possível obter os endereços IP de origem comprometidos, ou seja, os *hosts* que acessaram domínios maliciosos dentro da rede monitorada.

Ademais, as informações referentes ao destino do ataque são obtidas por meio de consultas ativas de DNS e técnicas de geolocalização, a fim de realizar um reconhecimento e levantamento de dados mais detalhado sobre os domínios maliciosos. O primeiro passo do processo é por meio da coleta dos registros do tipo A, que fornecem os endereços IP associados aos domínios. Em seguida, esses IPs são enriquecidos com dados de geolocalização, permitindo associar com informações sobre *Autonomous Systems* (AS) e o país onde os servidores estão hospedados.

3.7 DISPONIBILIZAÇÃO DOS RESULTADOS

Por fim, após o processamento dos domínios, os dados obtidos pelo sistema serão disponibilizados no formato JSON em *endpoints* de uma API REST. A API foi desenvolvida utilizando a linguagem de programação Python em conjunto com o *framework* Django, desenvolvido pela Django Software Foundation (2005) e focado para o desenvolvimento de *Web APIs*.

Além disso, esses dados ficarão disponíveis para utilização no *frontend* do DNS *Framework* (Batista, 2020), já abordado na Seção 2.5, disponibilizando representações visuais por meio de *dashboards* que facilitem a análise de profissionais.

4 TESTES E RESULTADOS

4.1 RESULTADOS DO *MACHINE LEARNING*

Conforme foi descrito na Seção 3.4 da metodologia, foi realizada a extração de características e o ajuste fino do modelo proposto, em conjunto com a combinação entre métricas léxicas e de DNS ativo. Nesta seção serão abordados os resultados do módulo de *machine learning*, como a performance durante a extração de características e o comparativo das métricas do modelo antes e depois do ajuste fino.

4.1.1 Extração de Características

Para a obtenção de características ativas foi necessário executar consultas DNS para os domínios analisados, resultando em diversas operações de entrada e saída pouco eficientes para um volume muito grande de dados.

Dessa forma, foram utilizadas amostras das listas utilizadas no treinamento (*Hagezi* TIF e *Majestic Million*) para avaliar o desempenho de extração entre três distintas situações: extração linear, paralela e paralela assíncrona. O resultados comparando amostras de 1000 e 100.000 domínios podem ser observados na Tabela 6.

Tabela 6 – Comparação de desempenho entre os métodos de extração de características

Método	Quantidade de domínios	Tempo de execução
Assíncrono	1000	15s
Paralelo	1000	1min 19s
Linear	1000	2min 50s
Assíncrono	100.000	7min 35s
Paralelo	100.000	1h 4min 45s
Linear	100.000	45h 10min 25s

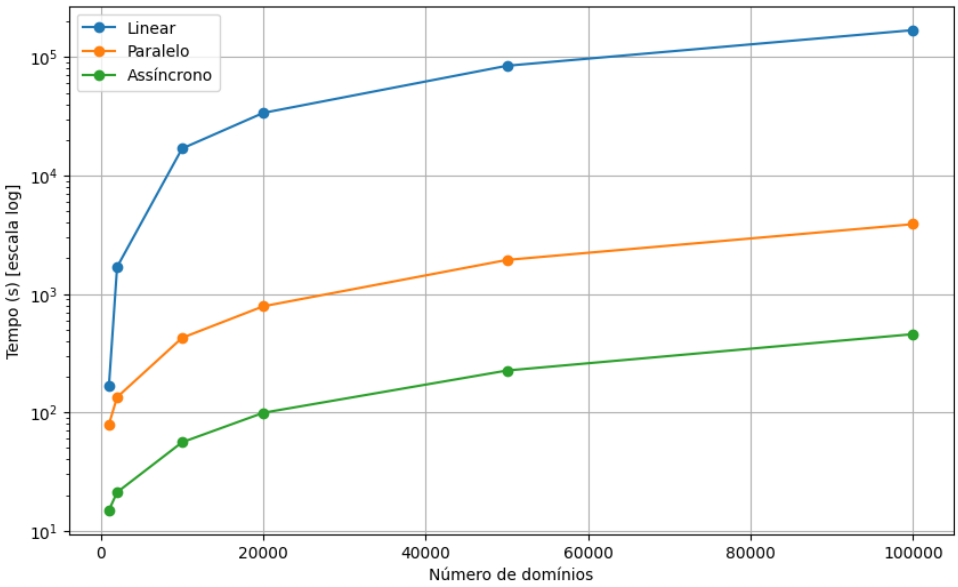
Fonte: Elaborado pelo autor.

Podemos observar que em ambos os casos a extração assíncrona obteve resultados consideravelmente melhores. Isso ocorre devido a natureza do problema, que consiste em milhares de requisições externas limitadas pelo tempo de latência da rede e não pelo processamento local.

Por outro lado, é possível perceber as requisições em paralelo possuem maior desempenho, visto que são disparadas simultaneamente em *threads* ou processos separados. Entretanto, cada *thread* ainda precisa ficar "parada" enquanto aguarda o retorno da resposta pela rede.

Dessa forma, quando as requisições são realizadas de forma assíncrona todas são disparadas e as respostas são processadas na medida em que chegam no sistema, de modo que não seja necessário esperar pelo retorno de uma requisição antes de lançar a próxima, garantindo uma melhor performance.

Figura 17 – Comparação de desempenho: Linear vs Paralelo vs Assíncrono



Elaborado pelo autor.

Na Figura 17 é possível observar a comparação do tempo de extração para os três métodos discutidos em amostras com tamanhos de 1000 a 100.000 domínios, evidenciando o aumento significativo da performance em métodos assíncronos conforme o aumento do volume de dados.

4.1.2 Resultados do *Fine-Tunning*

Conforme especificado na metodologia, o ajuste fino do modelo foi realizado para as versões com e sem TF-IDF, utilizando um total de 150 iterações do método *Randomized Search*. A principal ideia é realizar uma análise comparando o ganho de performance do modelo em relação ao seu custo computacional, a partir da variação dos hiperparâmetros. A métrica avaliativa utilizada nesse processo foi a AUC.

Tabela 7 – Top 5 melhores resultados da busca por hiperparâmetros

Max. Depth	*Max. Features*	*Min. Samples Leaf*	*Min. Samples Split*	*Num. of Estimators*	Melhor *Score*	*Score* Médio	Tempo médio (s)
34	*sqrt*	3	4	269	0.90223	**0.90054**	99.81327
23	*log2*	2	2	189	0.90197	0.90053	59.66101
33	*sqrt*	3	5	203	**0.90234**	0.90052	74.20145
31	*log2*	2	6	161	0.90206	0.90037	**53.05015**
46	*sqrt*	3	3	271	0.90203	0.90036	107.3818

Fonte: Elaborado pelo autor.

Os melhores resultados provenientes do processo de ajuste fino do modelo sem a utilização de TF-IDF podem ser observados na Tabela 7. Os resultados obtidos são muito simimilares, com

uma AUC média em torno de 0.9, entretanto, é possível observar uma variação considerável no tempo médio de execução, com o maior tempo (107 segundos) sendo aproximadamente o dobro do menor (53 segundos).

Dessa maneira, é possível perceber que pequenas mudanças nos hiperparâmetros não geram grandes diferenças no desempenho do modelo, no entanto, podem afetar consideravelmente o tempo.

Tabela 8 – Top 5 melhores resultados da busca por hiperparâmetros com TF-IDF

Max. Depth	*Max. Features*	*Min. Samples Leaf*	*Min. Samples Split*	*Num. of Estimators*	Melhor *Score*	*Score* Média	Tempo médio (s)
39	*log2*	2	9	287	0.91846	**0.91680**	327.43133
46	*sqrt*	3	3	271	0.91833	0.91673	349.80827
31	*log2*	2	6	161	0.91831	0.91673	187.37551
43	*log2*	2	5	113	0.91868	0.91670	**145.31606**
33	*log2*	1	5	287	**0.91880**	0.91670	435.91658

Fonte: Elaborado pelo autor.

Já na Tabela 8 estão os melhores resultados do ajuste fino para o modelo combinado com as características do TF-IDF. Assim, é possível observar que os resultados aumentaram consideravelmente, com uma AUC média chegando a até 0.9168. No entanto, apesar do ganho de desempenho do modelo, o tempo de execução obteve um grande salto, chegando até 435 segundos para o treinamento.

Dessa forma, é possível concluir que o modelo mais balanceado é o que utiliza as *features* de TF-IDF e possui o menor tempo de treinamento (145 segundos), visto que ele manteve uma excelente pontuação.

4.1.3 Testes dos Modelos

Após o treinamento e ajuste fino dos modelos, utilizou-se um conjunto de dados de teste para a avaliação das métricas. Assim, foram a avaliados quatro modelos distintos:

- *Baseline*: Modelo com os hiperparâmetros padrões sem nenhum tipo de ajuste, utilizando apenas as características descritas na Tabela 4;
- *Fine-Tunning*: Modelo com os hiperparâmetros ajustados, utilizando apenas as características descritas na Tabela 4;
- *Baseline* + TF-IDF: Mesmas características do modelo *baseline* com o acréscimo das *features* de TF-IDF;
- *Fine-Tunning* + TF-IDF: Mesmas características do modelo *Fine-Tunning* com o acréscimo das *features* de TF-IDF.

Na Tabela 9 podem ser observados os resultados do desempenho do modelo nos dados de teste, com aproximadamente 50 mil amostras não observadas durante o treinamento. Analisando os resultados é possível perceber que o modelo '*Fine-Tunning* + TF-IDF' obteve os melhores resultados para todas as métricas, indicando balanceamento nas classificações, enquanto o *Baseline* obteve o pior desempenho. Dessa forma, é possível observar uma maior relevância no enriquecimento de expressões léxicas do nome de domínio com a técnica TF-IDF.

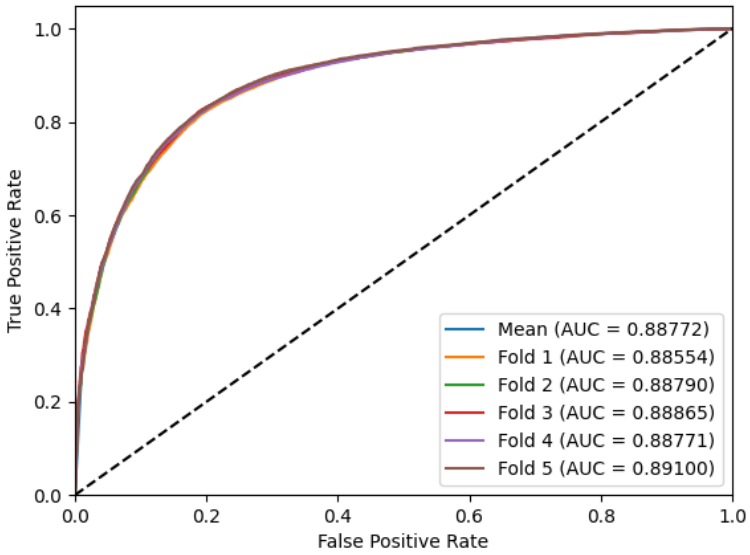
Tabela 9 – Comparação entre os modelos propostos

Modelo	AUC	Precisão	Sensibilidade	F1
Baseline	0.88772	0.75941	0.80834	0.78311
Fine-Tunning	0.90289	0.76883	0.83090	0.79866
Baseline + TF-IDF	0.90705	0.78979	0.82115	0.80516
Fine-Tunning + TF-IDF	**0.91768**	**0.79405**	**0.83183**	**0.81250**

Fonte: Elaborado pelo autor.

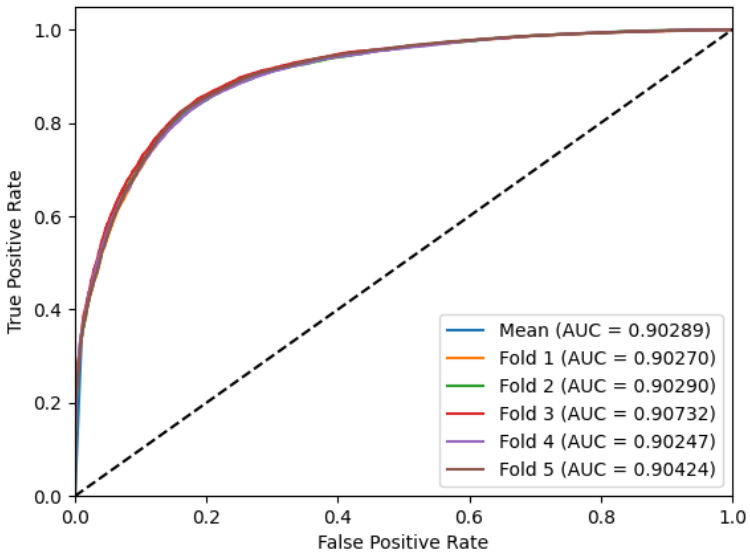
Além disso, realizou-se uma validação cruzada para os quatro modelos distintos, utilizando os mesmos conjuntos de dados separados na etapa de treinamento para a avaliação da curva ROC em 5 *folds* distintos. As curvas podem ser observadas na Figura 18, Figura 19, Figura 20 e Figura 21.

Figura 18 – Curva ROC do modelo *Baseline*



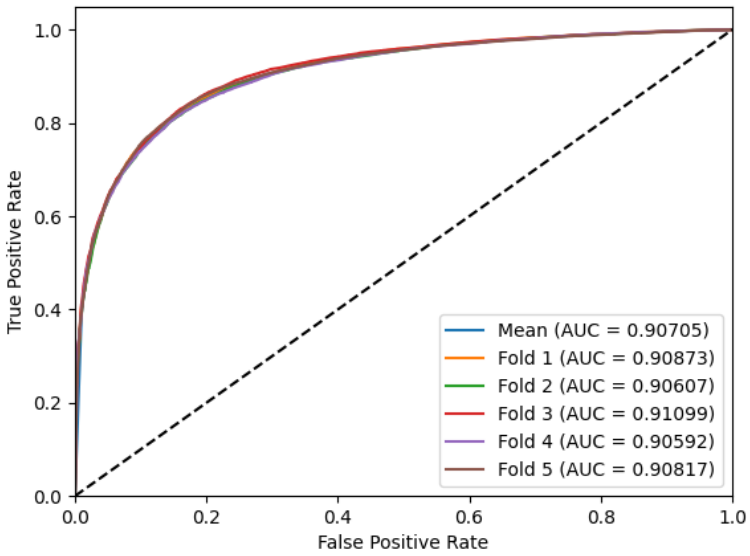
Fonte: Elaborado pelo autor.

Figura 19 – Curva ROC do modelo *Fine-Tunning*



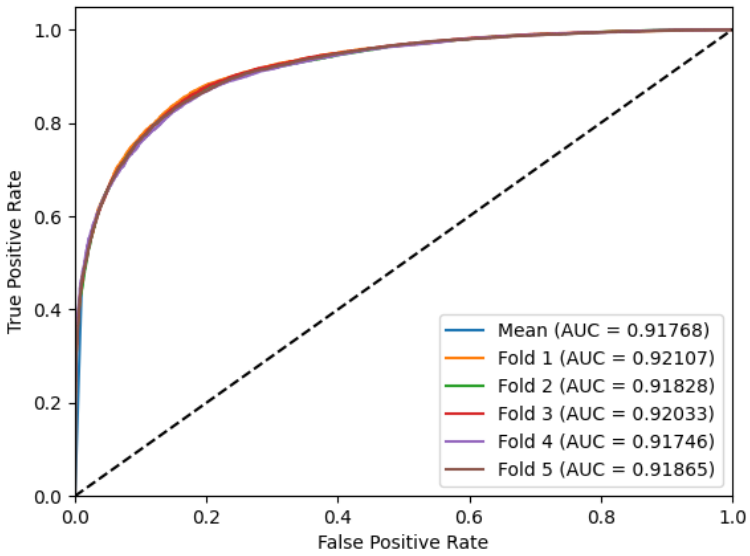
Fonte: Elaborado pelo autor.

Figura 20 – Curva ROC do modelo *Baseline* + TFIDF



Fonte: Elaborado pelo autor.

Figura 21 – Curva ROC do modelo *Fine-Tunning* + TFIDF



Fonte: Elaborado pelo autor.

Com base na análise das curvas apresentadas é possível evidenciar os resultados expressados na Tabela 9. Em linhas gerais, todos os modelos apresentaram proximidade das curvas em cada um dos *folds*, indicando uma baixa variabilidade no modelo independente da subdivisão dos dados. Além disso, é possível notar uma maior inclinação da curva na Figura 21 para o canto superior esquerdo, ainda que discreta, indicando uma maior sensibilidade do modelo com uma menor taxa de falsos positivos. Por fim, embora as curvas dos modelos indiquem resultados muito próximos, é possível observar uma melhora considerável quando são utilizadas características de TF-IDF.

4.2 RESULTADOS DA IDENTIFICAÇÃO DE AMEAÇAS

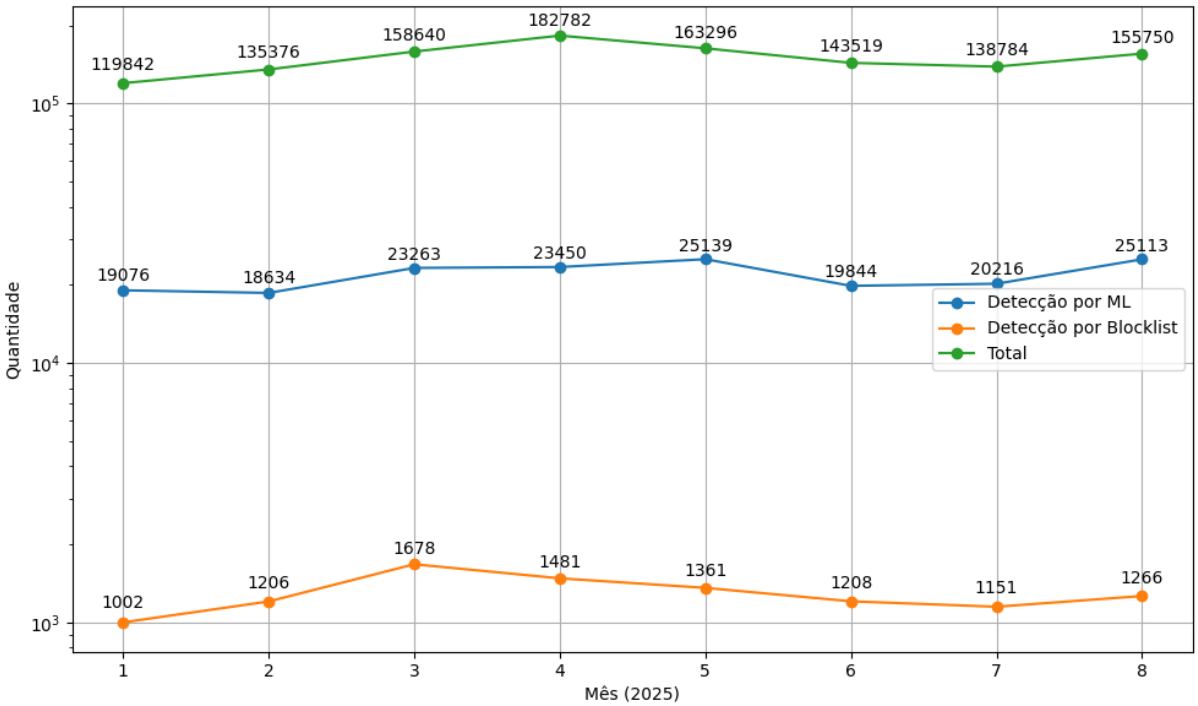
Conforme foi abordado na Seção 3.5 da metodologia, o trabalho utilizou uma versão combinada entre métodos tradicionais (listas de bloqueio) e ML para a identificação de ameaças em tráfego de DNS passivo.

A amostra de dados de DNS passivo analisada foi coletada em um período de 8 meses, tendo o início da coleta no dia 1 de janeiro de 2025 e a finalização no dia 31 de agosto de 2025. A evolução temporal de identificação dos domínios maliciosos mensalmente pode ser observada na Figura 22.

A Figura 22 destaca o total de domínios, a quantidade detectada pelo modelo de ML e a quantidade detectada pelas listas de bloqueio. O número total de domínios manteve-se entre 120 à 185 mil durante todo o período de coleta, com pequenas variações dependendo da atividade mensal na rede. Por outro lado, a quantidade de domínios maliciosos detectados pelo modelo

de aprendizado de máquina sofre algumas variações que não acompanham o número total de domínios, como pode ser observado entre os meses 1 e 2, onde o total de domínios cresce e os detectados pelo modelo diminuem.

Figura 22 – Detecção de domínios em 8 meses



Fonte: Elaborado pelo autor.

Além disso, os domínios detectados por listas de bloqueio são significantemente menores que os demais e possuem uma variação mínima entre os meses, refletindo a limitação dos métodos tradicionais em identificar um número muito grande de ameaças. Por fim, é possível observar que o modelo identificou um grande número de domínios que não estão presentes em listas de bloqueio, sugerindo que a combinação entre esses métodos auxilia nas detecções precoces de atividade maliciosa.

Tabela 10 – Comparação entre os domínios identificados

Mês	Precisão	Sensibilidade	Especificidade	Acurácia	F1
1	0.2620	0.5635	0.9449	0.9321	0.3577
2	0.3082	0.5249	0.9556	0.9400	0.3884
3	0.2575	0.4994	0.9394	0.9216	0.3398
4	0.2751	0.4568	**0.9600**	0.9438	0.3434
5	0.2929	0.5599	0.9513	0.9376	0.3846
6	**0.3988**	0.5539	0.9698	**0.9553**	**0.4637**
7	0.3333	**0.5908**	0.9572	0.9444	0.4262
8	0.3288	0.5871	0.9556	0.9425	0.4216

Fonte: Elaborado pelo autor.

A Tabela 10 mostra o desempenho do modelo em dados reais em constante modificação, durante os 8 meses de coleta. Diferente da métrica obtida durante a etapa de teste do treinamento, o modelo apresentou uma precisão consideravelmente baixa, de 0.2575 até 0.3988, indicando que o modelo gerou um número considerável de falsos alertas. Além disso, o modelo apresentou uma sensibilidade razoável, indicando sua capacidade de identificar domínios da classe maliciosa.

Por outro lado, o modelo apresentou acurácia e especificidade muito altas, com valores maiores de 0.9. Esses valores são observados por conta do desbalanceamento dos dados no mundo real, com a existência maior de domínios da classe benigna, como pode ser observado na Tabela 11 pela soma dos VP (verdadeiros positivos) com os FN (falsos negativos). Por fim, a F1 apresenta uma performance moderada, mas indica uma perda considerável tanto na detecção da classe benigna quanto da classe maliciosa.

Também é importante salientar que as métricas descritas nas Tabelas 10 e 11 foram avaliadas apenas em cima dos dados que estavam presentes em listas de bloqueio (classe maliciosa) ou em listas de permissão (classe benigna), ou seja, apenas nos dados que foi possível atribuir um rótulo.

Tabela 11 – Erros e acertos do modelo

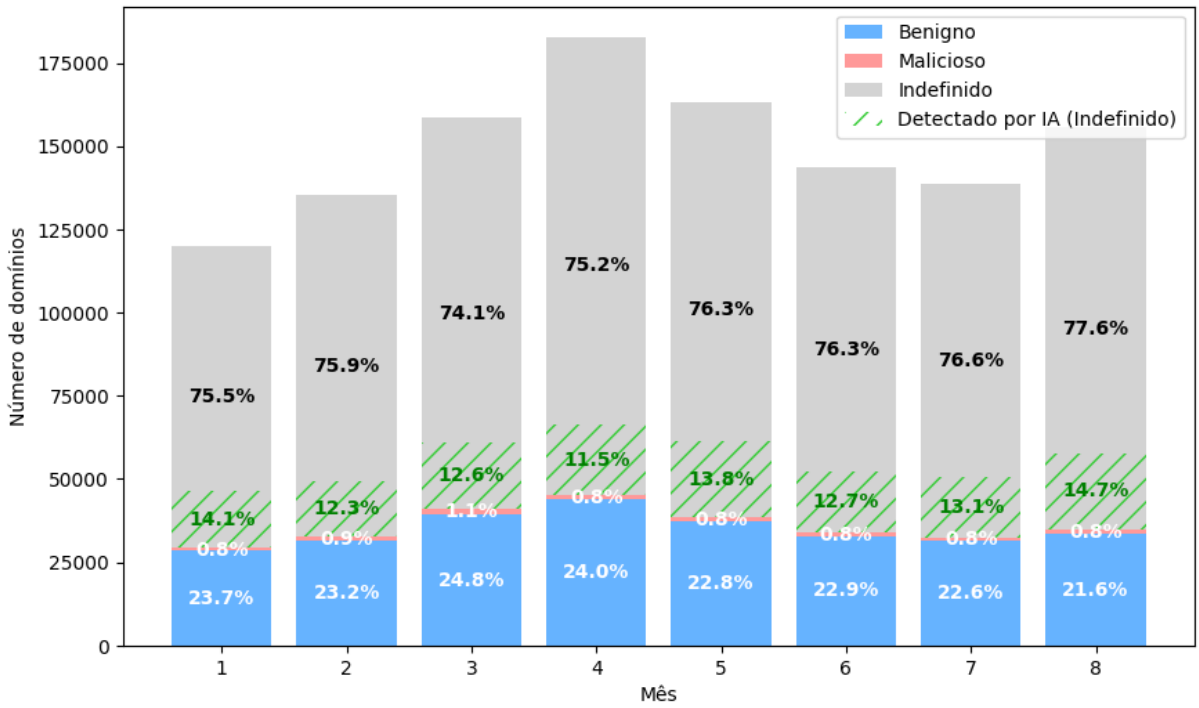
Mês	Total rotulado	VP	FP	FN	VN
1	29365	555	1563	430	26800
2	32670	622	1396	563	30068
3	41042	827	2385	829	36979
4	45344	665	1755	792	42108
5	38658	753	1818	592	35479
6	34038	658	992	530	31838
7	32473	670	1340	464	29982
8	34905	731	1492	514	32147

Fonte: Elaborado pelo autor.

Assim, analisando os resultados em conjunto com os dados da matriz de confusão presentes na Tabela 11, apesar de identificar centenas de domínios maliciosos mensalmente, o modelo apresentou um numero considerável de FP (falsos positivos). Além disso, é possível observar pelo número considerável de FN (falsos negativos) que o modelo também deixa passar domínios maliciosos.

Por fim, conforme pode ser observado na Figura 23, verifica-se a seguinte circunstância no cenário real: cerca de 74% à 77% dos domínios são indefinidos (não aparecem em listas de bloqueio ou de permissão), enquanto de 0.8% à 1.1% aparecem em listas de bloqueio e 21% à 24% aparecem em listas de permissão. Isso indica que a maior parte do tráfego não pode ser validado diretamente por métodos tradicionais. Por outro lado, dentro dos domínios indefinidos, observou-se uma variação de 11.5% à 14.7% de domínios maliciosos acusados pelo modelo, mostrando um potencial para a descoberta de ameaças que ainda não estão presentes em listas públicas.

Figura 23 – Distribuição mensal de domínios e detecções



Fonte: Elaborado pelo autor.

4.3 DISPONIBILIZAÇÃO DOS DADOS

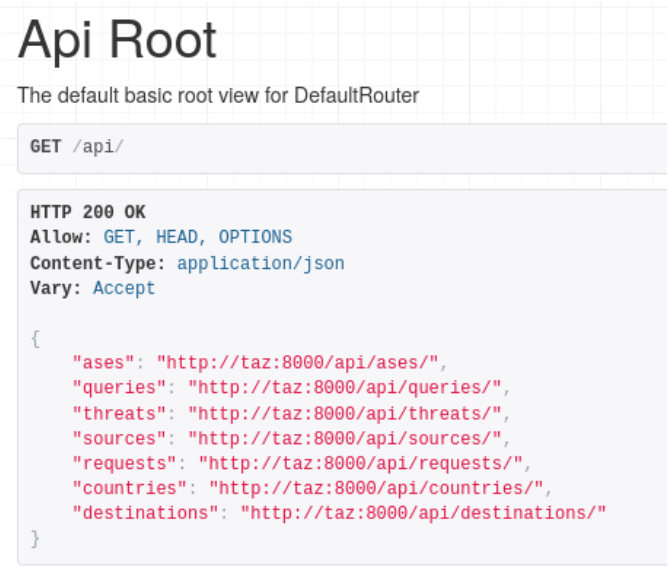
Conforme foi descrito na Seção 3.7 da metodologia, foi desenvolvida uma API REST para a disponibilização dos dados do sistema. Em conjunto, foram desenvolvidos *dashboards* que consomem esses dados para uma melhor análise visual dos dados de ameaças identificados no tráfego de DNS.

4.3.1 API do Sistema

A API desenvolvida é capaz de disponibilizar os resultados de forma estruturada e acessível. Essa API atua como camada de integração, permitindo que os dados obtidos sejam facilmente consumidos tanto por interfaces gráficas quanto por sistemas de monitoramento. Dessa forma, os resultados do modelo deixam de ser restritos a arquivos ou consultas manuais e passam a estar disponíveis em tempo real para administradores de rede, ampliando o impacto prático da solução.

Na Figura 24 é possível observar um panorama geral dos *endpoints* desenvolvidos e seus tipos de dados, entretanto, mais informações sobre a sua organização, parâmetros para a utilização e consumo dos dados da API pelo *frontend* podem ser observados no Apêndice A.

Figura 24 – *Endpoints* da API desenvolvida



Fonte: Elaborado pelo autor.

4.3.2 *Dashboards*

Para apresentar os resultados obtidos pela API, foram integrados *dashboards* no *Framework* DNS (Batista, 2020), a fim de aprimorar sua visualização e facilitar sua interpretação por analistas de segurança.

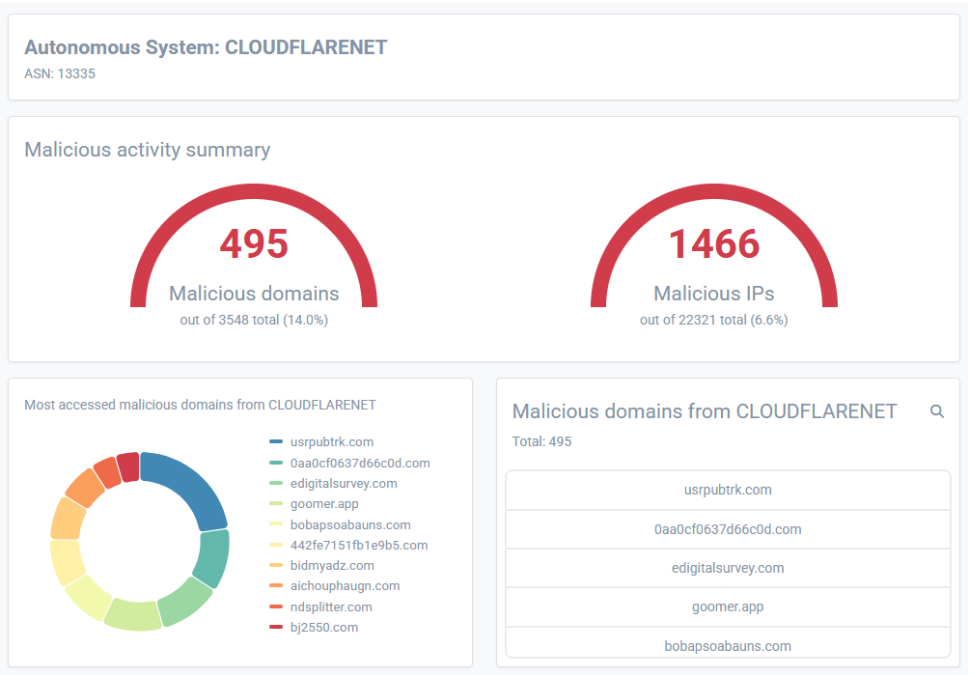
O primeiro *dashboard*, observado na Figura 25, mostra os dados de ASes relacionados com domínios maliciosos. No exemplo apresentado para o AS da *Cloudflare*, é possível visualizar a quantidade de domínios e IPs maliciosos, além dos principais domínios acessados. Esse tipo de visualização permite compreender rapidamente a concentração de ameaças em provedores específicos, auxiliando ações de mitigação.

Na Figura 26, observa-se a associação entre domínios maliciosos e países com as informações provenientes do módulo de enriquecimento de dados. O exemplo do Brasil evidencia o volume de domínios e IPs relacionados ao país. Esse painel contribui para a análise geográfica das ameaças, fornecendo informações para compreensão de ataques em nível regional.

Por fim, a Figura 27 apresenta informações detalhadas sobre domínios específicos detectados pelo sistema. Com base nos dados do DNS passivo, é possível identificar as datas de acesso ao domínio, os dispositivos infectados responsáveis pelas consultas e consultar informações enriquecidas, como os IPs e ASes associados.

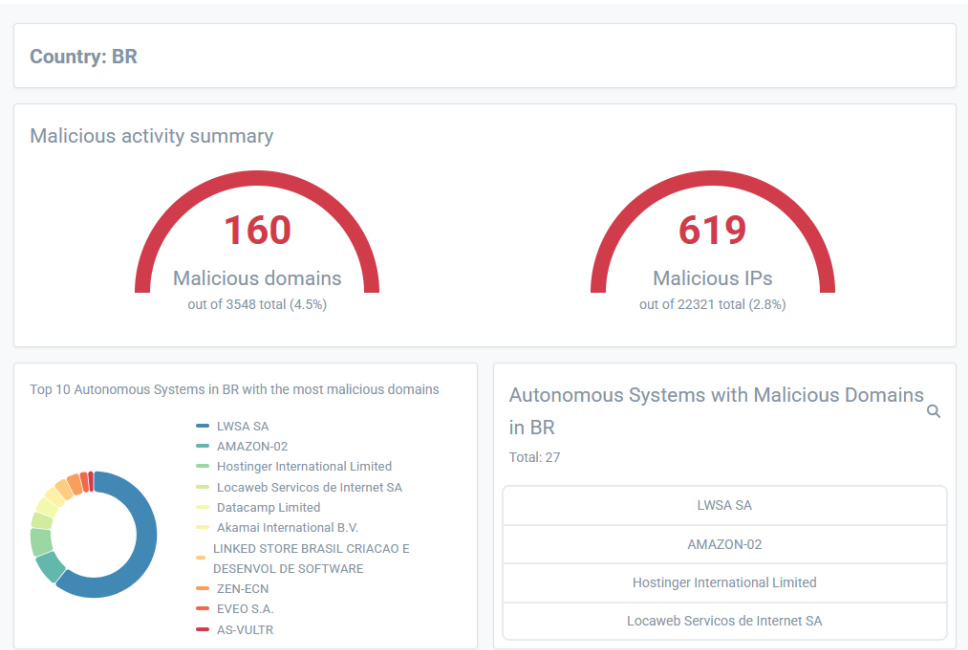
Dessa forma, os *dashboards* demonstram como a visualização dos dados complementa a análise, transformando métricas em informações acionáveis. Assim, é possível observar a importância da cibersegurança em camadas, combinando *blocklists*, aprendizado de máquina e painéis de monitoramento para tomada de decisão.

Figura 25 – Domínios maliciosos associados ao AS da Cloudflare



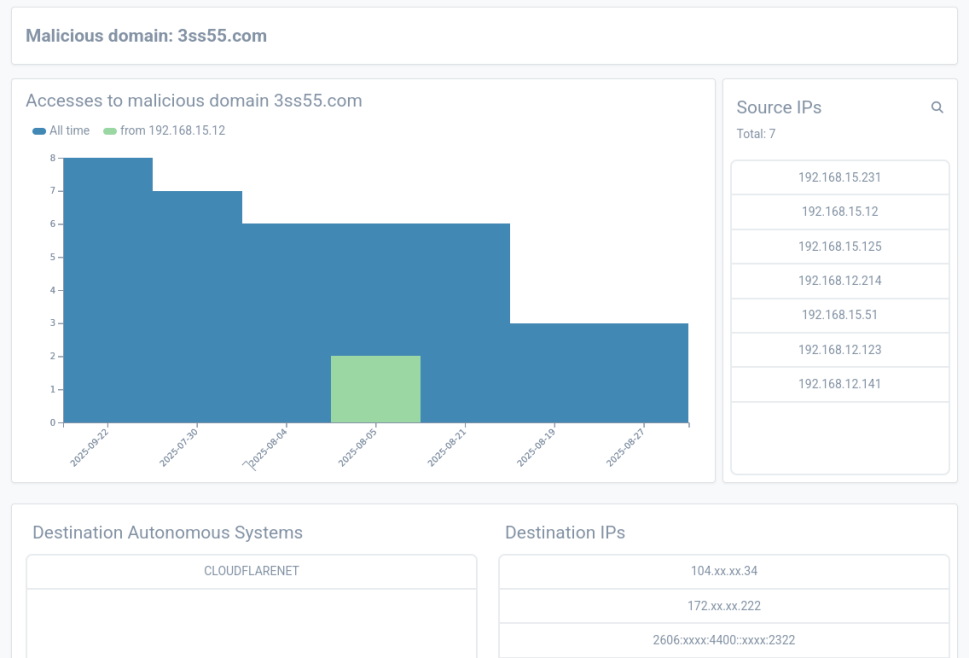
Fonte: Elaborado pelo autor.

Figura 26 – Domínios maliciosos associados ao Brasil



Fonte: Elaborado pelo autor.

Figura 27 – Detalhes de um domínio malicioso



Fonte: Elaborado pelo autor.

5 CONCLUSÃO

5.1 CONCLUSÃO GERAL

Este trabalho estudou uma abordagem combinada entre métodos tradicionais e baseados em aprendizado de máquina para a detecção de domínios maliciosos em tráfego de DNS passivo. A metodologia desenvolvida incluiu a utilização de características léxicas, de DNS ativo e enriquecimento com TF-IDF, em conjunto com a comparação entre diferentes estratégias para aumentar a performance na extração de características em larga escala para o sistema.

Conforme pode ser observado nos resultados, os experimentos mostraram que a extração de características de forma assíncrona foi essencial para o desempenho do sistema, reduzindo consideravelmente o tempo de extração de características para o treinamento do modelo. Isso foi de extrema importância para garantir a escalabilidade do sistema e adaptação para um grande volume de dados.

Durante o desenvolvimento do modelo classificador, o resultados mostraram que o uso de TF-IDF contribuiu para o aumento das métricas avaliativas, com destaque para o modelo ajustado por *fine-tunning* combinado com TF-IDF, chegando aos melhores resultados (AUC = 0.918, F1 = 0.81). Isso reforça a importância de explorar padrões no nome de domínio, visto que vários padrões podem ser identificados por essa técnica.

Por outro lado, em mundo real o modelo evidenciou dificuldades para a identificação de domínios. Apesar do bom desempenho nos dados de teste, quando foi realizada a análise no tráfego de DNS passivo o modelo apresentou uma baixa precisão (0.26-0.40), sensibilidade razoável (0.45-0.59) e alta especificidade (0.94). Isso mostra que existem limitações no balanceamento entre falsos positivos e falsos negativos, o que é esperado em ambientes reais e desbalanceados.

Por fim, foi possível notar que as listas de bloqueio identificam pouquíssimas ameças, confirmando a necessidade de métodos adicionais como o aprendizado de máquina. O modelo proposto apresentou potencial para a identificação de domínios que não estão presentes em listas de bloqueio, caracterizando uma detecção precoce. Esse resultado reforça um dos pilares da cibersegurança na qual a proteção deve ser estruturada por camadas, sem a dependência de um único mecanismo. Assim, o método proposto é extremamente eficiente como um filtro inicial, enviando os resultados para demais camadas que em conjunto irão garantir a integridade de um sistema computacional.

5.2 DIFICULDADES ENCONTRADAS

A primeira dificuldade encontrada no desenvolvimento do trabalho foi a extração de características, especialmente as características de DNS ativo, dada a grande quantidade de requisições necessárias para a obtenção dos dados. Para resolver esse problema foram utilizadas consultas

de DNS assíncronas via DoH, evitando *timeouts* indesejados.

Além disso, outra dificuldade está no tempo de treinamento do modelo no processo de ajuste fino, apresentando uma grande variação com base nos hiperparâmetros utilizados. A ideia inicial era utilizar o *Grid Search* do *scikit-learn*, realizando uma busca completa pelo espaço de hiperparâmetros, mas devido ao grande custo computacional e tempo de execução, optou-se por utilizar o *Randomized Search* com um número fixo de 150 iterações.

Por fim, a principal dificuldade do trabalho foi avaliar o desempenho do modelo final nos dados coletados do mundo real, devido ao grande desbalanceamento das classes e a grande quantidade de dados indefinidos. Para isso, optou-se por rotular uma porção dos dados com base em listas de bloqueio e permissão, a fim de possuir uma base para a avaliação do modelo.

5.3 TRABALHOS FUTUROS

Com base no trabalho desenvolvido, os seguites itens podem ser estudados para a melhoria desse trabalho:

- Dado o principal desafio do trabalho que é a classificação dos dados balanceados no mundo real, surge a possibilidade da exploração de técnicas de aprendizado não supervisionado ou semi-supervisionado, a fim de aproveitar os dados indefinidos para o processo de avaliação do modelo;
- Outro ponto de melhoria para esse trabalho é a utilização de modelos mais robustos, como arquitetura de aprendizado profundo (como *auto-encoders* e *transformers*) para a captura de padrões mais complexos em nomes de domínios e consultas DNS;
- Diminuir o nível de generalização do classificador, trocando um único modelo responsável por identificar domínios maliciosos por vários outros modelos capazes de identificar domínios associados a métodos específicos, como *phishing*, *malwares* ou domínios gerados por algoritmos.

Dessa forma, os trabalhos futuros tendem a aprimorar a metodologia descrita nesse trabalho, principalmente no uso de *machine learning* para a detecção de domínios maliciosos.

REFERÊNCIAS

ABUSE.CH; SPAMHAUS. **Fighting malware and botnets**. 2021. Disponível em: https://abuse.ch/blog/introducing-threatfox/.

AKAMAI. **What Is API Automation Testing?** 2025. Disponível em: https://www.akamai.com/glossary/what-is-api-automation-testing.

AKIBA, T. et al. Optuna: A next-generation hyperparameter optimization framework. In: **Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining**. [S.l.: s.n.], 2019. p. 2623–2631.

AURÉLIEN, G. Hands-on machine learning with scikit-learn, keras, and tensorflow. **Concepts, tools, and techniques to build intelligent systems, 2nd ed**, 2019.

BATISTA, V. B. Trabalho de Conclusão de Curso (Graduação), **Framework para integração e automação de modelos de aprendizado de máquina para classificação de nomes de domínio**. São José do Rio Preto: [s.n.], 2020.

BILGE, L. et al. Exposure: A passive dns analysis service to detect and report malicious domains. **ACM Transactions on Information and System Security (TISSEC)**, ACM New York, NY, USA, v. 16, n. 4, p. 1–28, 2014.

CONSULTING, L. B. **Bambenek Consulting DGA List**. 2025. Disponível em: https://faf.bambenekconsulting.com/.

Django Software Foundation. **Django**. 2005. Disponível em: https://djangoproject.com.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. 2000. Tese (Ph.D. dissertation) — University of California, Irvine, 2000. Disponível em: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.

GHAFIR, I.; PRENOSIL, V. Dns traffic analysis for malicious domains detection. In: **2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN)**. [S.l.: s.n.], 2015. p. 613–918.

GOMES, R. L. Trabalho de Conclusão de Curso (Graduação), **Detecção de domínios maliciosos baseado em características textuais e aprendizado de máquina**. São José do Rio Preto: [s.n.], 2018.

GREGÓRIO, J. et al. Deep convolutional neural network and character level embedding for dga detection. In: **Proceedings of the 26th International Conference on Enterprise Information Systems**. [S.l.: s.n.], 2024. v. 2, p. 167–174.

HAGEZI, G. **Hagezi/DNS-blocklists: DNS-blocklists: For a better internet - keep the internet clean!** 2025. Disponível em: https://github.com/hagezi/dns-blocklists.

KOUNTOURAS, A. et al. Enabling network security through active dns datasets. In: SPRINGER. **International Symposium on Research in Attacks, Intrusions, and Defenses**. [S.l.], 2016. p. 188–208.

KUROSE, J. F.; ROSS, K. W. **Redes de Computadores e a internet: Uma abordagem top-down**. 6. ed. [S.l.]: Pearson, 2012.

LISKA, A.; STOWE, G. **DNS security: Defending the domain name system**. [S.l.]: Syngress is an imprint of Elsevier, 2016.

MARQUES, C.; MALTA, S.; MAGALHÃES, J. Dns firewall based on machine learning. **Future Internet**, MDPI, v. 13, n. 12, p. 309, 2021.

MOCKAPETRIS, P. V. **Domain names - implementation and specification**. [S.l.], 1987. Disponível em: https://rfc-editor.org/rfc/rfc1035.txt.

NIC.BR. **NIC.br celebra 35 anos do .br, um dos Domínios mais populares do Mundo**. 2024. Acessado em: 05/08/2025. Disponível em: https://cgi.br/noticia/releases/nic-br-celebra-35-anos-do-br-um-dos-dominios-mais-populares-do-mundo/.

OMARZAI, F. **Random Forest in-depth**. Medium, 2024. Disponível em: https://medium.com/@fraidoonomarzai99/random-forest-in-depth-f0556817c40b.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

POMPEU, A. Trabalho de Conclusão de Curso (Graduação), **Detecção de domínios maliciosos por meio de aprendizado de máquina utilizando coleta ativa de DNS**. São José do Rio Preto: [s.n.], 2018.

POUR, M. et al. A comprehensive survey of recent internet measurement techniques for cyber security. **Computers Security**, v. 128, p. 103123, 05 2023.

QUINLAN, J. R. Learning decision tree classifiers. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 28, n. 1, p. 71–72, 1996.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence: A modern approach**. 4. ed. [S.l.]: Pearson, 2021.

SALMAN, H. A.; KALAKECH, A.; STEITI, A. Random forest algorithm overview. **Babylonian Journal of Machine Learning**, Mesopotamian Academic Press, v. 2024, p. 69–79, jun. 2024.

SCHNEIER, B. **Secrets and lies**. Nashville, TN: John Wiley & Sons, 2004.

SILVA, L. M. d. Trabalho de Conclusão de Curso (Graduação), **Classificação de domínios recém-registrados por meio de DNS passivo aplicando técnicas de aprendizado de máquina**. São José do Rio Preto: [s.n.], 2022.

SILVEIRA, M. R. et al. Detection of newly registered malicious domains through passive dns. In: IEEE. **2021 IEEE International Conference on Big Data (Big Data)**. [S.l.], 2021. p. 3360–3369.

SONI, P. **Confusion matrix, precision, and recall**. 2024. Disponível em: https://www.blog.trainindata.com/confusion-matrix-precision-and-recall/.

SPEROTTO, A.; TOORN, O. van der; RIJSWIJK-DEIJ, R. van. Tide: Threat identification using active dns measurements. In: **Proceedings of the SIGCOMM Posters and Demos**. New York, NY, USA: Association for Computing Machinery, 2017. (SIGCOMM Posters and Demos '17), p. 65–67. ISBN 9781450350570. Disponível em: https://doi.org/10.1145/3123878.3131988.

STEVENS, W. R. **TCP/IP illustrated (vol. 1): the protocols**. USA: Addison-Wesley Longman Publishing Co., Inc., 1993. ISBN 0201633469.

TANENBAUM, A. S. **Computer Networks**. 4. ed. Upper Saddle River, NJ: Pearson, 2002.

WEIMER, F. Passive dns replication. In: **FIRST conference on computer security incident**. [S.l.: s.n.], 2005. v. 98, p. 1–14.

WULLINK, M. et al. Entrada: A high-performance network traffic data streaming warehouse. In: **NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium**. [S.l.: s.n.], 2016. p. 913–918.

ZHENG, A. **Evaluating machine learning models**. [S.l.]: O'Reilly Media, Inc, 2015.

APÊNDICE A – ENDPOINTS DA API

Este apêndice documenta os *endpoints* disponibilizados pela API desenvolvida. Todos seguem o padrão REST e retornam dados em formato JSON. Filtros podem ser aplicados por meio de *query parameters*.

A.1 /API/QUERIES/

Descrição

Retorna estatísticas de consultas realizadas por dia.

Parâmetros de Filtro

- `date`: filtra pela data exata.

Exemplo de Resposta

```
[
  {
    "date": "2025-09-28",
    "total_alerts": 1023,
    "total_threats": 34,
    "total_domains": 890
  }
]
```

A.2 /API/THREATS/

Descrição

Lista os domínios identificados como maliciosos.

Parâmetros de Filtro

- `name`: nome do domínio.
- `asn`: número do sistema autônomo.
- `country`: país associado ao IP.
- `detected_at`: data da detecção.

Exemplo de Resposta


```
[
  {
    "name": "malicious-domain.com",
    "total_count": 58,
    "detected_at": {
      "date": "2025-09-28",
      "hour": "13:42:11"
    }
  }
]
```


A.3 /API/REQUESTS/

Descrição

Retorna as ocorrências de ameaças detectadas em consultas diárias.

Parâmetros de Filtro

- `name`: domínio associado.
- `date`: data da consulta.
- `source_ip`: endereço de origem.

Exemplo de Resposta


```
[
  {
    "name": "phishing-site.net",
    "count": 12,
    "date": "2025-09-28"
  }
]
```


A.4 /API/ASES/

Descrição

Retorna estatísticas agregadas por Sistemas Autônomos (AS).

Parâmetros de Filtro

- `asn`: número do sistema autônomo.
- `address`: IP de destino.
- `as_name`: nome do sistema autônomo.
- `country`: país do IP.
- `name`: domínio associado.

Exemplo de Resposta

```
[
  {
    "asn": "13335",
    "as_name": "Cloudflare, Inc.",
    "threats_count": 15,
    "ips_count": 42
  }
]
```

A.5 /API/COUNTRIES/

Descrição

Retorna estatísticas agregadas por país.

Parâmetros de Filtro

- `asn`: número do sistema autônomo.
- `address`: IP de destino.
- `as_name`: nome do sistema autônomo.
- `country`: país do IP.
- `name`: domínio associado.

Exemplo de Resposta

```
[
  {
    "country": "US",
```

```
    "threats_count": 20,
    "ips_count": 87
  }
]
```


A.6 /API/SOURCES/

Descrição

Lista endereços IP de origem responsáveis pelas consultas.

Parâmetros de Filtro

- `address`: endereço IP.
- `date`: data associada à consulta.
- `name`: domínio associado.

Exemplo de Resposta


```
[
  {
    "address": "192.168.0.10",
    "count": 45
  }
]
```


A.7 /API/DESTINATIONS/

Descrição

Lista endereços IP de destino relacionados a ameaças.

Parâmetros de Filtro

- `asn`: número do sistema autônomo.
- `address`: IP de destino.
- `as_name`: nome do sistema autônomo.
- `country`: país do IP.
- `name`: domínio associado.

Exemplo de Resposta


```
[
  {
    "address": "8.8.8.8",
    "asn": "15169",
    "as_name": "Google LLC",
    "country": "US"
  }
]
```

DADOS CURRICULARES

IDENTIFICAÇÃO	
	Guilherme Romanholo Bofo 15/07/2003
Nacionalidade	Brasileiro
Nome em citações bibliográficas	BOFO, G. R. BOFO, Guilherme R.
Currículo Lattes	http://lattes.cnpq.br/6326792901451841
ORCID	https://orcid.org/0009-0009-0558-0394
Outro identificador	–
FORMAÇÃO ACADÊMICA	
2019/2021	Ensino Médio (2° grau). Colégio Esquema Universitário Rio Preto
2022/2025	Bacharelado em Ciência da Computação Universidade Estadual Paulista “Júlio de Mesquita Filho” - UNESP
PRODUÇÃO BIBLIOGRÁFICA	
BOFO, G. R.; CANSIAN, A. M. Automatização e Aprimoramento de Classificadores para a Detecção de Domínios Maliciosos. In: XXXVI Congresso de Iniciação Científica e Tecnológica da Unesp, 2024, São José do Rio Preto. Anais do XXXVI Congresso de Iniciação Científica da Unesp: "Ciência em tempos de crise climática e social", 2024.	
PARTICIPAÇÃO EM BANCAS E ORIENTAÇÕES	
Bancas de trabalhos de conclusão	
Orientações	
PARTICIPAÇÃO EM EVENTOS CIENTÍFICOS	
XXXVII Congresso de Iniciação Científica e Tecnológica da Unesp, 2025, São José do Rio Preto. Detecção de Domínios Maliciosos em Tráfego de DNS Passivo Utilizando Fontes de Inteligência de Ameaças e Machine Learning. 2025. (Painel). XXXVI Congresso de Iniciação Científica e Tecnológica da Unesp, 2024, São José do Rio Preto. Automatização e Aprimoramento de Classificadores para a Detecção de Domínios Maliciosos. 2024. (Painel).	