UNIVERSIDADE ESTADUAL PAULISTA - UNESP

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

GABRIELE ZANCHETA SCAVAZINI

AMAZONAS – Módulo de scoring e monitoramento de domínios recém registrados

São José do Rio Preto

2026

Gabriele Zancheta Scavazini

AMAZONAS – Módulo de scoring e monitoramento de domínios recém registrados

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

Área de Concentração: Computação Aplicada

Orientador: Adriano Mauro Cansian

São José do Rio Preto
2026

S288a

Scavazini, Gabriele Zancheta
AMAZONAS - Módulo de scoring e monitoramento de domínios recém registrados / Gabriele Zancheta Scavazini. -- São José do Rio Preto, 2026
97 p.

Dissertação (mestrado) - 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).

IMPACTO POTENCIAL DESTA PESQUISA

Esta pesquisa contribui para a segurança cibernética ao apoiar a detecção e o monitoramento de domínios maliciosos com IA. Seu impacto inclui reduzir fraudes digitais, apoiar a análise humana, proteger ativos digitais e aplicar a solução em universidades, órgãos públicos e empresas.

POTENTIAL IMPACT OF THIS RESEARCH

This research contributes to cybersecurity by supporting the detection and monitoring of malicious domains with AI. Its impact includes reducing digital fraud, supporting human analysis, protecting digital assets, and applying the solution in universities, public agencies, and companies.

GABRIELE ZANCHETA SCAVAZINI

AMAZONAS – MÓDULO DE SCORING E MONITORAMENTO DE DOMÍNIOS RECÉM REGISTRADOS :

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

Área de Concentração: Computação Aplicada

Data de defesa: 24/04/2026

BANCA EXAMINADORA

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

---

Profž Dr. Arnaldo Cândido Junior
Instituto de Biociências, Letras e Ciências Exatas

Profž Dr. Vinicius Fulber Garcia
Universidade Federal do Paraná

Dedico este trabalho à minha querida mãe
Meire Luci Zancheta Scavazini

AGRADECIMENTOS

Agradeço ao meu orientador Adriano Mauro Cansian pela sabedoria e conselhos repassados durante os anos de mestrado. Agradeço também toda a equipe do laboratório ACME! pelos conselhos, risadas e momentos de tensão. Por fim, agradeço a todos que de alguma forma influenciaram na produção deste texto.

O presente trabalho foi realizado com apoio da Fundunesp proc. 3467/2023 e NIC.br.

“The physical body exists at a less evolved plane only to verify ones existence in the universe“
(Ueda; Konaka, 1998, ‘Layer 06: KIDS‘, Serial Experiments Lain)

RESUMO

O Sistema de Nomes de Domínio (DNS), embora fundamental para a usabilidade da Internet, é frequentemente explorado como um vetor para atividades maliciosas, como *phishing*, *typosquatting* e a operação de Algoritmos de Geração de Domínio (DGA). A crescente sofisticação dessas ameaças demanda novas soluções para a identificação de domínios perigosos. Este trabalho propõe e detalha o desenvolvimento do AMAZONAS, um módulo de segurança projetado para classificar e monitorar domínios potencialmente maliciosos. A metodologia é estruturada em duas fases distintas: a Fase I utiliza um conjunto de classificadores modulares baseados em Inteligência Artificial para realizar uma análise inicial e atribuir um *score* de risco aos domínios, focando em características léxicas, de typosquatting e DGA. Domínios considerados suspeitos são encaminhados para a Fase II, que implementa um sistema de monitoramento contínuo através de coletores de dados *web*, DNS ativo e DNS passivo, a fim de acumular evidências para uma análise humana final. O presente estudo resultou na validação de três classificadores baseados em Deep Learning, dois focados em domínios maliciosos específicos e outro de caráter geral. Nos três modelos, a acurácia ficou acima de 80%, reforçando o potencial de aplicação em cenários reais de monitoramento. Além disso, os coletores demonstraram capacidade de agregar dados acionáveis de domínio e, de forma complementar, a geração automática de *prompts* para LLM mostrou-se útil como apoio à análise humana na Fase II, sem substituir a decisão final do analista. Testes finais realizados com cerca de 20 mil domínios concluíram a capacidade do AMAZONAS em detectar e separar domínios em 3 categorias diferentes. Por fim, com a ajuda de LLMs, foi possível certificar que domínios benignos e maliciosos em suas devidas listas.

Palavras-Chave: sistema de nome de dominios; segurança de domínios; machine learning; deep learning; typosquatting.

ABSTRACT

The Domain Name System (DNS), while fundamental to the usability of the Internet, is often exploited as a vector for malicious activities, such as phishing, typosquatting, and the operation of Domain Generation Algorithms (DGA). The growing sophistication of these threats demands new solutions for identifying dangerous domains. This work proposes and details the development of AMAZONAS, a security module designed to classify and monitor potentially malicious domains. The methodology is structured in two distinct phases: Phase I uses a set of modular classifiers based on Artificial Intelligence to perform an initial analysis and assign a risk score to domains, focusing on lexical, typosquatting, and DGA characteristics. Domains deemed suspicious are forwarded to Phase II, which implements a continuous monitoring system through web, active DNS, and passive DNS data collectors, in order to gather evidence for a final human analysis. This study resulted in the validation of three Deep Learning-based classifiers, two focused on specific malicious domain patterns and one with a broader scope. Across all three models, accuracy remained above 80%, reinforcing the potential for real-world deployment. Furthermore, the collectors demonstrated the ability to acquire actionable domain intelligence and, as a complementary step, automatic LLM *prompt* generation proved useful to support human analysis in Phase II while preserving analyst final decision-making. Final tests conducted with approximately 20,000 domains confirmed AMAZONAS ability to detect and separate domains into three different categories. Finally, with the support of LLMs, it was possible to corroborate that benign and malicious domains were assigned to their appropriate lists.

Keywords: domain name system; domain security; machine learning; deep learning; typosquatting.

LISTA DE FIGURAS

Figura 1 Hierarquia DNS 15
Figura 2 Visão abstrata de medida de DNS passivo vs ativo 17
Figura 3 Exemplo de classificação utilizando *Random Forest*. 23
Figura 4 Visão conceitual da arquitetura BERT para classificação. 30
Figura 5 Funcionamento geral do projeto AMAZONAS. 33
Figura 6 Representação do banco de dados do projeto AMAZONAS Fase II. 35
Figura 7 Fase I do módulo AMAZONAS detalhada. 37
Figura 8 Fase II do módulo AMAZONAS detalhada. 41
Figura 9 Distribuição dos domínios por classe de classificação. 45
Figura 10 Distribuição da pontuação final de classificação no intervalo $[0, 1]$. 46
Figura 11 Distribuição do classificador acionado na decisão final. 47
Figura 12 Proporção de domínios encaminhados e não encaminhados para a Fase II. 47
Figura 13 Conteúdo do zip gerado pelo coletor web. 51
Figura 14 Visão do módulo Nevermore com a consulta contendo "google.com". . . . 58
Figura 15 Visão do módulo de classificadores, contendo os classificadores Léxico e Ativo. 58
Figura 16 Visão do módulo de *dashboard* exibindo algumas das informações acionáveis disponíveis. 59

LISTA DE TABELAS

Tabela 1 – Técnicas de DGA e malwares utilizados 20
Tabela 2 – Taxonomia de Técnicas de Typosquatting com Exemplos 21
Tabela 3 – Campos do conjunto de dados 44
Tabela 4 – Métricas de avaliação do modelo Typosquat 48
Tabela 5 – Métricas de avaliação dos modelos BERT e DistilBERT 49
Tabela 6 – Comparação das métricas de avaliação entre os classificadores 50
Tabela 7 – Hiperparâmetros Utilizados no Treinamento do Modelo BERT 60
Tabela 8 – Hiperparâmetros Utilizados no Treinamento do Modelo DistilBERT 61
Tabela 9 – Hiperparâmetros Utilizados no Treinamento do Modelo Canine-c 62

SUMÁRIO

1 INTRODUÇÃO 13

2 FUNDAMENTAÇÃO TEÓRICA 15
2.1 DNS 15
2.1.1 Tipos de registro DNS 16
2.1.2 DNS passivo 17
2.2 ARQUITETURA DE IMPLEMENTAÇÃO 17
2.2.1 Abusos em nomes de domínio 18
2.3 MACHINE LEARNING 20
2.4 REDES NEURAIS PROFUNDAS 22
2.4.1 Estrutura de uma rede neural 23
2.4.2 Funções de ativação 24
2.4.3 Funções de perda 25
2.4.4 Backpropagation 26
2.4.5 Otimização 27
2.4.6 Modelos Transformer 29
2.5 TRABALHOS DE BASE 29
2.6 ESTUDOS RELACIONADOS 31
2.6.1 ENTRADA 2 31
2.6.2 DNSCheck 31

3 METODOLOGIA 33
3.1 BANCO DE DADOS 34
3.2 DATASETS 34
3.2.1 DGAs 34
3.2.2 Typosquat 35
3.2.3 Léxico 36
3.3 FASE I 36
3.3.1 Classificadores 36
3.4 FASE II 40
3.4.1 Extração Web 42
3.4.2 Extração de DNS ativo 42
3.4.3 Extração de DNS Passivo 43

4 RESULTADOS 45
4.1 VISÃO GERAL DOS RESULTADOS OPERACIONAIS 45

4.1.1 Distribuição de classificação e listas . . . 45
4.1.2 Distribuição de score e acionamento de classificadores . . . 46
4.1.3 Encaminhamento para a Fase II . . . 46
4.2 RESULTADOS DA FASE I: CLASSIFICADORES . . . 46
4.2.1 Classificador DGA . . . 48
4.2.2 Classificador Typosquat . . . 48
4.2.3 Classificadores Léxicos . . . 48
4.2.4 Resultado geral dos classificadores . . . 49
4.3 RESULTADOS DA FASE II: COLETORES . . . 50
4.3.1 Coletor Web . . . 50
4.3.2 Coletor de DNS Ativo . . . 51
4.3.3 Coletor de DNS Passivo . . . 51
4.3.4 Apoio à decisão com LLM . . . 51

5 CONCLUSÃO . . . 53
5.1 TRABALHOS FUTUROS . . . 53

REFERÊNCIAS . . . 55

Appendices . . . 57

A DNSCHECK . . . 58

B HIPERPARÂMETROS DE TREINAMENTO . . . 60

C CÓDIGO DOS CLASSIFICADORES . . . 63

D SCRIPT DE GERAÇÃO AUTOMÁTICA DE PROMPT PARA LLM . . . 67

E EXEMPLO DE *PROMPT* PARA DOMÍNIO POTENCIALMENTE MALICIOSO . . . 76

F EXEMPLO DE *PROMPT* PARA DOMÍNIO SEGURO . . . 80

G RESPOSTA DA IA PARA DOMÍNIO POTENCIALMENTE MALICIOSO . . . 84

H RESPOSTA DA IA PARA DOMÍNIO BENIGNO . . . 88

I OUTPUT DOS COLETORES . . . 91

DADOS CURRICULARES . . . 95

1 INTRODUÇÃO

No contexto de um mundo globalizado, a utilização da Internet tornou-se imprescindível. Consequentemente, surgem mecanismos para otimizar sua usabilidade, como o DNS (*Domain Name System*), que traduz nomes facilmente memorizáveis, a exemplo de “example.com", para seus respectivos endereços IP (Kurose; Ross, 2017). Seu surgimento em 1983, ainda na era da ARPANET[1], permitiu que os computadores abandonassem o uso de um único arquivo de texto, chamado `Hosts.txt`, a favor desse sistema altamente distribuído.

Mas assim como qualquer outra coisa na área de tecnologia, novas descobertas trazem novos riscos e não foi diferente com o DNS. Agentes mal-intencionados passaram a explorar essa tecnologia para fins ilícitos, como DNS *Hijacking* (Sequestro de DNS), DNS *Spoofing* (Falsificação de DNS) e ataques de negação de serviço distribuído (DDoS) amplificados (Kurose; Ross, 2017; Mockapetris, 1987). O foco principal deste trabalho, no entanto, recai sobre outra ameaça: a utilização de domínios com aparência legítima como vetores para atividades maliciosas, como *phishing* e a operação de *Domain Generation Algorithms* (DGA) (ManageEngine, 2025; Plohmann et al., 2016).

Desta forma, surge então a necessidade de identificação de domínios legítimos (que não possuem a intenção de lesar ou gerar algum mal para um bem ou pessoa) e domínios utilizados por agentes maliciosos. Para abordar essa problemática, propõe-se a criação do AMAZONAS, um módulo projetado para fácil integração a *frameworks* existentes. Sua função é classificar e, subsequentemente, monitorar domínios potencialmente maliciosos em um processo de duas fases. Para tal, serão utilizadas tecnologias de IA (Inteligência Artificial) e gerenciamento de APIs (como Django e FastAPI). Na implementação apresentada nesta dissertação, o sistema utiliza 3 módulos de IA classificadora para verificação inicial de um determinado domínio.

OBJETIVO

O objetivo do módulo AMAZONAS é utilizar duas etapas distintas para a detecção, *scoring* e monitoramento de domínios recém registrados. Para isso, são utilizados, em sua primeira etapa, os seguintes classificadores de IA:

- Classificador Léxico: utiliza informações léxicas e contextuais do domínio de uma maneira geral, portanto sendo o classificador mais abrangente.
- Classificador de *Typosquatting*: A detecção de *typosquat* analisa a similaridade entre palavras e atribui um *threshold* para indicar o quão similar um domínio pode ser em relação ao alvo sem caracterizar uma técnica de *phishing*.

[1] *Advanced Research Projects Agency Network* rede de computadores construída em 1969 para que ocorresse transmissão de dados sigilosos e de pesquisa nos EUA. Descontinuada para a implementação da Internet.

- Classificador de DGA: Detecta domínios gerados por algoritmos, alguns deles possuem pouca similaridade aos olhos humanos, enquanto outros são baseados em dicionários. São normalmente utilizados por famílias de *malwares* e de *botnets* para que seja possível a sua comunicação entre o *host* (infectado) e seu controlador (usuário malicioso).

Já na segunda etapa, são coletadas informações que permitem o monitoramento dos domínios classificados como maliciosos, sendo elas:

- Coletor Web: verifica e coleta o conteúdo da página *web* do domínio monitorado.
- Coletor DNS Ativo: coleta dados de DNS de forma ativa (realizando *queries* de DNS) a fim de monitorar mudanças de endereços IP, configurações de entrega de e-mail (como registros MX e TXT/SPF) e outras informações obtidas em consultas DNS.
- Coletor DNS Passivo: utilizando infraestrutura de DNS passivo, os dados são filtrados para consultar acessos ao domínio monitorado. Com isso, é possível analisar eventuais endereços de IP conhecidos por estarem em *blocklists* ou em famílias de *botnets*.

Portanto, o objetivo central deste trabalho é desenvolver e validar um protótipo funcional do módulo AMAZONAS, demonstrando sua eficácia com a abordagem modular e multifásica na identificação de domínios maliciosos.

Este documento se encontra estruturado da seguinte forma: o Capítulo 2 apresenta a fundamentação teórica sobre DNS e as técnicas de aprendizado de máquina utilizadas. O Capítulo 3 detalha a metodologia de construção do AMAZONAS, incluindo a visão sobre os *datasets* e a arquitetura dos classificadores e coletores. O Capítulo 4 apresenta e discute os resultados obtidos na avaliação dos modelos. Finalmente, o Capítulo 5 conclui o trabalho, sumarizando as contribuições e apontando direções para pesquisas futuras.

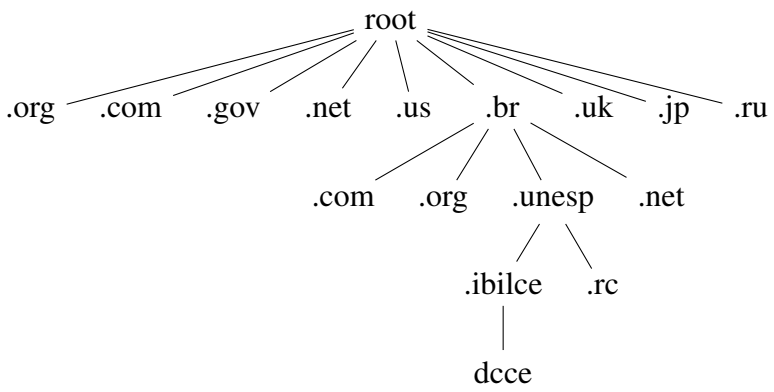
2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão apresentados fundamentos essenciais para a criação do módulo AMAZONAS, assim como estudos que de alguma forma estão relacionados e/ou incorporados dentro do módulo.

2.1 DNS

Criado em 1983 como RFC 1035 para solucionar a constante expansão do número de computadores interconectados e substituir a obtenção manual do `HOSTS.txt`, que caracterizava um único ponto de falha. Com isso, o sistema que antes era centralizado agora se torna distribuído e de forma hierárquica (Kurose; Ross, 2017).

Figura 1 – Hierarquia DNS



Fonte: Autora.

O formato hierárquico do DNS pode ser exemplificado e separado em alguns componentes:

- DNS recursivos: DNS recursivo representa sua ponta final, ou aquele que está conectado mais próximo do cliente. Além disso, tem a função de armazenar em seu cache consultas realizadas por usuários e realizar a conexão com servidores autoritativos.
- DNS autoritativos: Representam a estrutura central do DNS e contêm dados de autoridade sobre um domínio. Também podem delegar subdomínios a outros servidores autoritativos, que passam a ser responsáveis por essas zonas. Respondem aos servidores recursivos com informações sobre os domínios para os quais têm autoridade.
- *Root Servers*: Servidores raiz ou ".", são aqueles que possuem a autoridade máxima no nível hierárquico do DNS. Respondem aos servidores recursivos a delegação de TLD (*Top-Level Domains*).

A hierarquia do sistema de nomes de domínio pode ser simplificada na Figura Figura 1, que representa apenas a estrutura em níveis do espaço de nomes, partindo da raiz até domínios e subdomínios. O fluxo de resolução, por sua vez, ocorre quando o cliente consulta um servidor recursivo; esse servidor percorre a hierarquia consultando servidores autoritativos em níveis superiores até obter a resposta ou determinar a não existência do domínio. Por fim, o servidor recursivo envia a resposta ao usuário.

2.1.1 Tipos de registro DNS

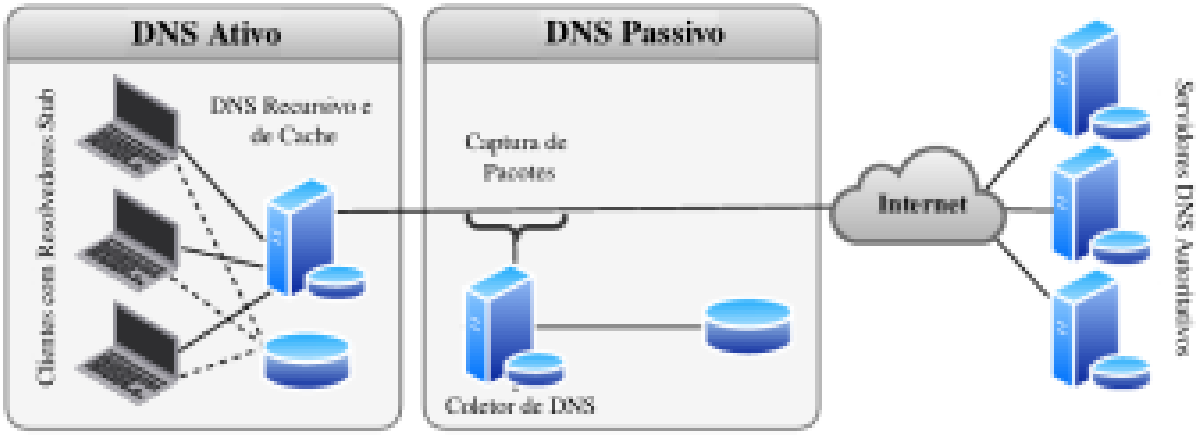
A variedade de tipos de registro ou *Resource Records* (RR) demonstram a flexibilidade do DNS, evoluindo de um simples sistema para mapeamento de nomes para endereços de IP em um banco de dados distribuído com proposito geral (Mockapetris, 1987). Os tipos de RR mais comuns são:

- A (*Address Record*): Mapeia um nome de domínio diretamente para um endereço IPv4.
- AAAA (IPv6 *Address Record*): Semelhante ao registro A, o AAAA mapeia um nome de domínio para um endereço IPv6.
- CNAME (*Canonical Name*): Cria um *alias*, ou apelido, apontando um nome de domínio para outro nome de domínio. Quando um *resolver* encontra um CNAME, ele reinicia a consulta usando o nome canônico.
- MX (*Mail Exchange*): Especifica os servidores de e-mail responsáveis por receber mensagens para um domínio.
- NS (*Name Server*): Delega uma zona de DNS para ser gerenciada por um conjunto específico de servidores de nomes autoritativos. Cruciais para o funcionamento da hierarquia distribuída do DNS.
- TXT (*Text Record*): Permite que um administrador de domínio insira texto arbitrário em um registro. Tem uso em segurança e verificação, como as políticas SPF (*Sender Policy Framework*) e DKIM (*DomainKeys Identified Mail*) para autenticação de e-mails.
- SOA (*Start of Authority*): Cada zona de DNS deve ter um único registro SOA, que contém informações administrativas cruciais. Isso inclui o servidor de nomes primário da zona, o e-mail do administrador e o número de série da zona.
- PTR (*Pointer Record*): Usado principalmente para a resolução reversa de DNS, que é o processo de mapear um endereço IP de volta para um nome de domínio.

2.1.2 DNS passivo

Diferentemente de como foi especificada uma consulta de DNS ativo anteriormente, em que um cliente realiza a requisição direta a um servidor de nomes de domínio para obter uma resposta em tempo real, o DNS Passivo (pDNS) é uma técnica de reconhecimento que se baseia em coletar e armazenar dados de resolução DNS ao longo do tempo. Ou seja, diferente da consulta direta, o pDNS registra respostas que foram enviadas para outros clientes as capturando normalmente em servidores de DNS recursivos (Edmonds, 2012). A diferença entre o DNS ativo e passivo pode ser vista na Figura Figura 2, em que, enquanto no ativo computadores realizam diretamente a consulta para um servidor recursivo, o passivo escuta essa linha de comunicação entre computadores realizando a consulta e recebendo a resposta.

Figura 2 – Visão abstrata de medida de DNS passivo vs ativo



Fonte: Adaptado de Pour et al. (2023)

2.2 ARQUITETURA DE IMPLEMENTAÇÃO

O módulo AMAZONAS foi implementado como uma aplicação Django composta por múltiplos aplicativos internos, com compartilhamento de ORM, banco de dados e configuração central. Essa organização foi adotada para reduzir acoplamentos operacionais entre componentes e facilitar a integração entre classificação (Fase I) e monitoramento (Fase II).

Sob essa arquitetura, a persistência de dados foi estruturada para manter rastreabilidade ponta a ponta: desde a entrada do domínio, passando pelos resultados individuais dos classificadores, até os artefatos coletados durante o monitoramento contínuo.

Os sistemas de DNS passivo seguem as seguintes etapas conceituais:

- Estágio inicial de captura de pacotes em servidores de DNS recursivo de alto tráfego.
- Registros são enviados para um banco de dados central (nesta dissertação, o ENTRADA 2), onde serão indexados juntamente com seus metadados, como a data/hora de consulta e localização com base em GeoIP.

- Registros são processados, reduzidos e disponibilizados por intermédio de interface SQL (Wullink et al., 2016).
- Analistas podem consultar os dados e seu histórico para investigar diversas informações sobre um *host* ou domínio sem nunca ter que enviar um único pacote para seus servidores recursivos.

Enquanto a consulta ativa de DNS mostra apenas o estado atual de um domínio, o PDNS surge como vantagem permitindo criar um perfil histórico e contextual de *host*/domínio.

2.2.1 Abusos em nomes de domínio

A ICANN define abuso de DNS em cinco categorias (ICANN, 2024):

- *Botnets*: Conjunto de dispositivos infectados por *malwares* que realizam C2 (*Command and Control*) com objetivo de extrair informações ou realizar ações maliciosas. Exemplos: *Mirai* e *Mozi* (Edmonds, 2012; Jeremiah et al., 2025).
- *Malware*: *Software* que realiza ações maliciosas em um sistema sem o consentimento do usuário. Exemplos: *Spyware*, *Adware* e *Ransomware*.
- *Pharming*: Redirecionamento não intencional de site legítimo para ilegítimo por meio de manipulação ou envenenamento de DNS com objetivo de obter as credenciais de acesso da vítima.
- *Phishing*: Estratégia de engenharia social que se baseia na ideia de induzir a vítima a realizar uma ação de forma desatenta ou não intencional. Utiliza-se da identidade de uma pessoa, empresa ou serviço para “fisgar" o usuário, realizando assim, o roubo de seus dados ou provocando o *download* de algum *malware*.
- *Spam*: Conjunto de e-mails enviados em massa sem a autorização do remetente.

DGA

Ao comprometer um sistema, um agente malicioso precisa manter a comunicação com suas vítimas por meio de um servidor de C2 (*Command and Control* – Comando e Controle). Para tal, a utilização de nomes de domínio se torna viável, pois permite a alteração constante de endereços IP, evitando detecção simples por bloqueios estáticos. Contudo, à medida que equipes de segurança identificam e bloqueiam esses domínios maliciosos, os atacantes necessitam de um método para contornar tal defesa. Para resolver esse problema, foi adotada a estratégia de Algoritmos de Geração de Domínio (*Domain Generation Algorithms* – DGA) (Plohmann et al., 2016; Jeremiah et al., 2025).

Fundamentalmente o DGA cria um grande número de nome de domínios de forma algorítmica e pseudoaleatória, onde apenas alguns dos domínios gerados serão realmente registrados pelo atacante em um determinado momento. A princípio, tanto a máquina infectada pelo *malware* quanto o servidor do atacante executam o mesmo algoritmo, muitas vezes utilizando uma semente em comum (tal como a data e hora), garantindo que ambos saibam qual domínio estará ativo sem a necessidade de uma comunicação prévia, criando um alvo móvel, e tornando listas de bloqueio estáticas muitas vezes ineficientes, pois, quando um domínio for bloqueado, o *malware* provavelmente estará tentando se comunicar através de dezenas de outros nomes de domínio desconhecidos (Plohmann et al., 2016).

As técnicas de DGA evoluíram para se tornarem evasivas, podendo ser primordialmente categorizadas de acordo com a natureza dos domínios gerados:

- DGAs baseados em caracteres aleatórios: criam sequências de caracteres de alta entropia, que não se assemelham a palavras reais, como `kdjf8923hsd.com`. Facilmente identificáveis por humanos, mas eficazes para evadir filtros de reputação.

- DGAs baseados em dicionário: sua geração de domínio é baseada na concatenação de palavras de um dicionário (ex: `greenstreetmorning.net`). Tais domínios são mais difíceis de serem identificados por algoritmos, já que sua estrutura lexical se assemelha à de domínios legítimos.

A Tabela 1 detalha exemplos de diversas técnicas que empregam e famílias de *malware* que as utilizam, isso também ilustra a diversidade e possível sofisticação desses algoritmos.

Typosquatting

A criação de nomes de domínio fraudulentos explora uma fraqueza fundamental da percepção humana: dificuldade de identificar pequenas variações em cadeias de caracteres. A técnica de *typosquatting* explora especialmente essa fraqueza, registrando nomes de domínios que são variações intencionais de marcas e serviços conhecidos, com o intuito de subverter e enganar o usuário final (Kaspersky, 2025; University of Texas at San Antonio, TechSolutions, 2024; ManageEngine, 2025).

Contudo, tais técnicas transcendem apenas erros simples de digitação e podem utilizar diversas técnicas muito mais profundas. Demonstram um grande entendimento sobre a psicologia do usuário, *layouts* de teclados[1] e ambiguidades linguísticas. O objetivo é interceptar tráfego de usuários que digitam um endereço incorretamente ou que clicam em links fraudulentos enganados, acreditando ser um serviço legítimo. A Tabela 2 apresenta uma taxonomia detalhada das técnicas mais comuns, desde simples omissões de caracteres até ataques de homógrafos.

[1] Seja pela proximidade de teclas como “u" e “i" ou alterações baseadas em linguística, como caracteres cirílicos

Tabela 1 – Técnicas de DGA e malwares utilizados

Técnica	**Descrição**	**Família de *Malware***
Geração aleatória	Gera cadeias com aparência aleatória a partir de um gerador pseudoaleatório, sem depender necessariamente de um valor externo interpretável pelo analista.	BumbleBee
Geração baseada em semente	Usa uma entrada compartilhada, como data, chave ou identificador de campanha, para que cliente infectado e servidor atacante reproduzam a mesma lista de domínios.	GameoverZeus, Conficker
Geração por algoritmo	Cria-se um algoritmo específico utilizando regras para produzir domínios a partir de um estado inicial.	Dridex
Geração baseada em dicionário	Percorre um dicionário ou lista de palavras para construir domínios.	Mirai
Geração baseada em tempo	É um caso específico de geração baseada em semente no qual a data ou horário define janelas de validade, produzindo listas diferentes a cada dia, hora ou período.	Storm Worm, Orchard V3
Geração baseada em *hash* criptográfico	Utiliza funções criptográficas para gerar domínios aparentemente aleatórios.	Locky
Geração ciente de contexto	Baseia-se no sistema ao qual está inserido para geração de domínios.	Spargo

Fonte: Jeremiah et al. (2025)

2.3 MACHINE LEARNING

Ao considerar que 2.5 quintilhões de bytes de dados são gerados todos os dias (MIT Technology Review Brasil, 2022), devemos também pensar em uma forma de processar dados de maneira eficiente. Nesse contexto, *Machine Learning* (ML) pode ser definido como a área que estuda algoritmos capazes de melhorar seu desempenho em uma tarefa a partir da experiência (dados), sem programação explícita para cada caso (Mitchell, 1997). Assim, o objetivo principal de um modelo de ML é, após treinamento em um determinado conjunto de dados, realizar previsões, classificações ou análise de padrões (Paixão et al., 2022).

Com isso, a vasta gama de algoritmos de *Machine Learning* pode ser organizada em seus paradigmas fundamentais, que são:

- Aprendizagem supervisionada: Paradigma mais comum (Ludermir, 2021). O algoritmo aprende a partir de um conjunto de dados rotulado e seu objetivo é aprender uma função de mapeamento geral que associa novas entradas (que não foram vistas durante o treinamento) a saídas corretas. Como exemplo, um classificador de domínios maliciosos

Tabela 2 – Taxonomia de Técnicas de Typosquatting com Exemplos

Técnica	**Descrição**	**Legítimo**	**Typosquatting**
Substituição de Caracteres	Troca de caracteres por outros visualmente semelhantes (letras por números, etc.).	google.com	g00gle.com
Omissão	Remoção de um ou mais caracteres do nome de domínio.	facebook.com	facebok.com
Adição	Adição de um ou mais caracteres ao nome de domínio.	twitter.com	twittter.com
Transposição	Troca da ordem de dois caracteres adjacentes.	example.com	exmaple.com
TLD Incorreto	Utilização de um Domínio de Nível Superior (TLD) diferente, mas semelhante.	amazon.com	amazon.co
Hifenização	Adição ou remoção de hífens no nome de domínio.	meubanco.com	meu-banco.com
Combosquatting	Anexação de palavras-chave (como login, seguro) a uma marca.	paypal.com	paypal-seguranca.com
Ataque de Homógrafos	Utilização de caracteres Unicode de outros alfabetos que são visualmente idênticos.	apple.com	apple.com (com a cirílico)

Fonte: Kaspersky (2025), University of Texas at San Antonio, TechSolutions (2024), ManageEngine (2025), Welch (2025)

receberia milhares de domínios rotulados como “maliciosos" e “benignos" e, após seu treinamento, deverá ser capaz de classificar corretamente um domínio nunca visto antes (Behera; Das, 2017). As tarefas do aprendizado supervisionado podem ser divididas em duas categorias:

  - Classificação: Têm como objetivo prever o rótulo de saída de forma discreta. De forma a exemplificar, a detecção de uma fraude pode ser categorizada como: “fraudulenta" e “não fraudulenta", sendo estes dados discretos.
  - Regressão: Tem como objetivo prever um valor contínuo, como valores de ações, de clima e do mercado imobiliário.

- Aprendizagem não supervisionada: Diferentemente do aprendizado supervisionado, o algoritmo é alimentado com dados não rotulados. O propósito é explorar e encontrar estruturas e padrões por conta própria. Uma forma de encontrar padrões é a criação de agrupamentos ou *clusters*, de modo que dados dentro de um mesmo *cluster* são semelhantes entre si. Pode ser utilizado, por exemplo, para agrupamento de clientes ou detecção

de anomalias em grandes conjuntos de dados.

- Aprendizagem semi-supervisionada: Paradigma intermediário entre o supervisionado e o não supervisionado. Utiliza um pequeno conjunto de dados rotulados combinado com um grande volume de dados não rotulados. A ideia é aproveitar a informação dos dados rotulados para construir uma função de mapeamento mais precisa. Esta abordagem é útil quando a aquisição de dados é fácil, mas a rotulagem é custosa.

Random Forest Classifier

O *Random Forest* é um algoritmo de aprendizado de máquina que utiliza a ideia de agregação de múltiplos modelos simples de *decision tree* para aumentar a precisão e robustez do modelo final. A técnica em questão é baseada em *Ensemble Learning*, combinando vários modelos para tomar uma decisão final (Breiman, 2001).

No contexto deste trabalho, *decision trees* são modelos que particionam o espaço de características por meio de regras sequenciais até chegar a uma decisão de classe. Já *Ensemble Learning* corresponde à combinação de múltiplos modelos para reduzir variância e aumentar robustez preditiva.

O poder do algoritmo reside na introdução da aleatoriedade em dois níveis distintos, o que permite construir um conjunto de árvores de decisão não correlacionadas:

- Amostragem aleatória de dados (*Bagging*): Cada árvore na “floresta" é treinada de forma individual sobre uma pequena amostra aleatória dos dados de treinamento, promovendo a diversidade entre os modelos do conjunto.

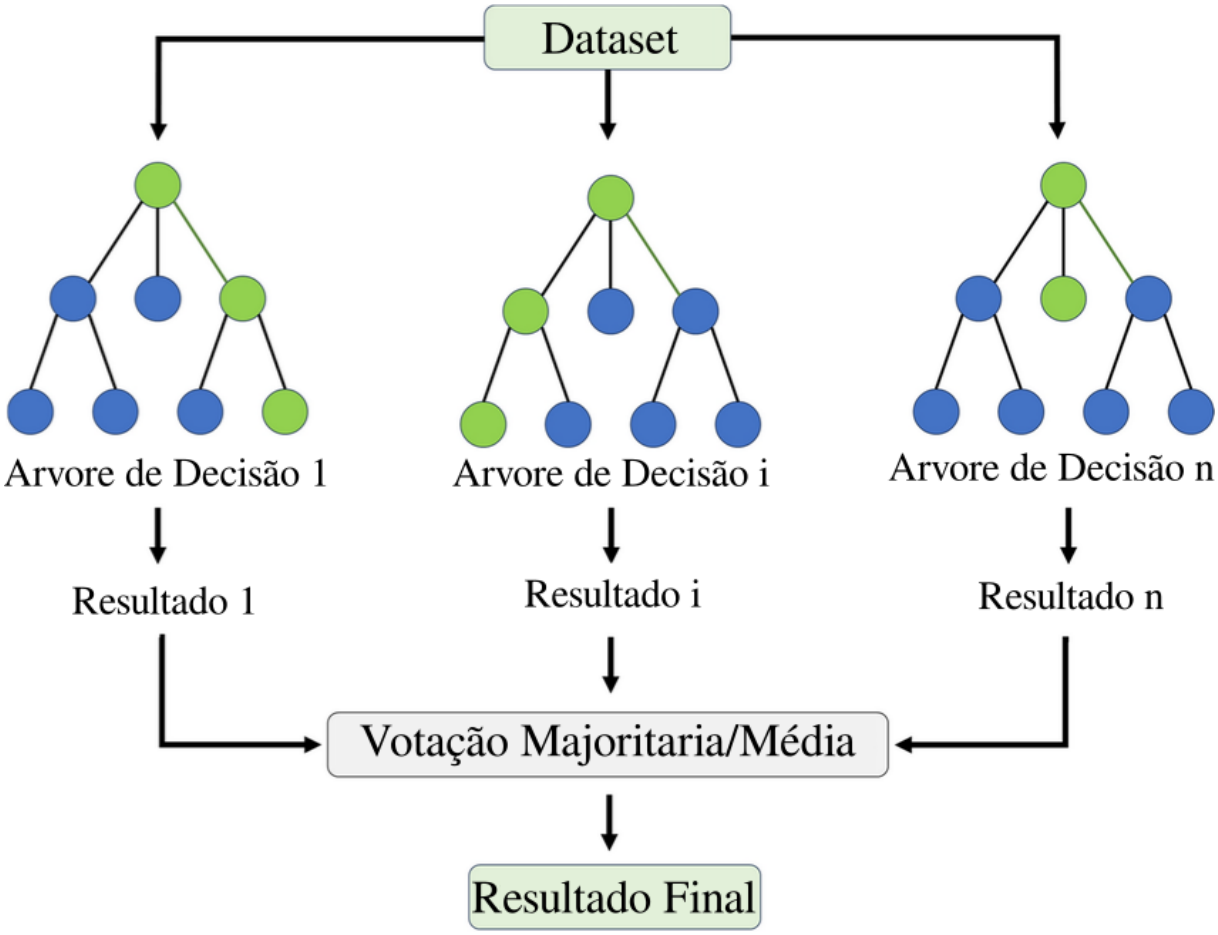
- Seleção aleatória de características: Em cada nó de árvore, o algoritmo considera um subconjunto aleatório de características ao invés de avaliar todas as características disponíveis. Tal estratégia diminui a correlação entre árvores, impedindo que características mais fortes dominem todos os modelos.

Dessa forma, cada uma das árvores dentro da floresta é treinada de forma independente utilizando uma amostra aleatória de dados de treinamento e uma seleção aleatória de características para cada nó, evitando *overfitting*. A predição final do modelo é feita a partir de uma votação majoritária, em que o resultado de cada uma das árvores é contabilizado e o vencedor é tomado como o resultado da classificação, como pode ser visto na Figura Figura 3.

2.4 REDES NEURAIS PROFUNDAS

Um outro ponto de avanço das técnicas de ML são as técnicas de *Deep Learning* (DL), que utilizam modelos computacionais com múltiplas camadas de processamento para aprender representações com mais níveis de abstração. Como exemplo, na análise de um domínio, as

Figura 3 – Exemplo de classificação utilizando *Random Forest*.



Fonte: Adaptado de Subramaniam, Jeyananthan & Sathiparan (2023)

primeiras camadas aprendem a detectar n-gramas de caracteres comuns; as camadas intermediárias aprendem a combinar esse n-grama em padrões lexicais suspeitos (como palavras de dicionário mal concatenadas) e, por fim, camadas finais reconhecem a estrutura geral de um DGA ou um domínio legítimo. Em sua base, está a capacidade de realizar o *representation learning* (Aprendizado por representações), um paradigma onde a máquina descobre de forma autônoma características para executar uma tarefa (LeCun; Bengio; Hinton, 2015).

Diferente de estratégias tradicionais de ML, que exigem o processo de *feature engineering* (engenharia de características), métodos baseados em DL automatizam essa etapa (Lecun et al., 1998). Cada camada de uma rede neural profunda recebe como entrada uma representação que foi gerada por uma camada anterior e a transforma em uma outra representação mais abstrata e complexa. (LeCun; Bengio; Hinton, 2015).

2.4.1 Estrutura de uma rede neural

Uma rede neural é composta por unidades interconectadas apelidadas de neurônios, organizadas em formato de camadas: uma camada de entrada (ou *input*), uma ou mais camadas

ocultas (ou *hidden*) e, por fim, uma camada de saída (ou de *output*). A quantidade de camadas ocultas define a profundidade de uma rede neural (LeCun; Bengio; Hinton, 2015).

Dentro de cada camada, cada um dos neurônios calcula sua saída a partir de entradas da camada anterior. Primeiro, se calcula a soma ponderada de todas as entradas e se adiciona o viés (ou *bias*), cuja formula é vista na Equação 1, na qual x_i são as entradas, w_i são os pesos e b é o viés (Rumelhart; Hinton; Williams, 1986).

$$z = \left(\sum_{i=1}^{n} w_i x_i \right) + b \tag{1}$$

Contudo, se uma rede utilizasse apenas a soma ponderada, mesmo em múltiplas camadas, seria matematicamente equivalente a um modelo linear. O que traz a diferença às redes neurais é utilizar funções não lineares e extremamente complexas para função de ativação sobre z (LeCun; Bengio; Hinton, 2015).

2.4.2 Funções de ativação

As funções de ativação adicionam a não-linearidade fundamental para a rede. Algumas das mais importantes são:

- Sigmoide logística: Comprime qualquer valor de entrada para um intervalo entre 0 e 1, como visto na Equação 2. No entanto, sofre de saturação (a derivada se aproxima de zero para valores grandes), o que contribui para o desaparecimento do gradiente (*vanishing gradients*) (Glorot; Bordes; Bengio, 2011).

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

- Tangente Hiperbôlica (Tanh): Centrada em zero, e limitada entre -1 e 1, visto na Equação 3, isto é: sua média de saída tende a ser próxima de zero. Permite que gradientes para a próxima camada sejam tanto positivos quanto negativos, o que gera uma convergência mais rápida e estável durante o treinamento. Também, assim como a sigmoide, sofre com o desaparecimento de gradiente em regiões de grande magnitude (Goodfellow; Bengio; Courville, 2016).

$$tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{3}$$

- ReLU (*Rectified Linear Unit)*: Função muito utilizada em redes profundas. Com uma fórmula simples (vista na Equação 4), reduz o problema do desaparecimento do gradiente para valores positivos, pois sua derivada é 1 nessa região, permitindo que o gradiente flua durante o *backpropagation*. Ainda assim, não resolve totalmente o problema, já que para valores negativos pode ocorrer o fenômeno de *dying ReLU*.

$$f(z) = max(0, z) \tag{4}$$

- Softmax: Não opera em valor único, mas em um vetor de K valores reais z (*logits*), visto na Equação 5. Transforma o vetor em uma distribuição de probabilidade, onde cada elemento da saída está no intervalo (0,1) e a soma de todos os elementos é igual a 1. É a escolha padrão para a camada de saída em tarefas de classificação multiclasse, pois, a saída pode ser diretamente interpretada como a probabilidade da entrada pertencer a cada uma das K classes.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{para } i = 1, \ldots, K \tag{5}$$

2.4.3 Funções de perda

Após os dados serem passados pela rede, o modelo gera uma predição. Para ser treinado, são necessários mecanismos para medir o quão distante a predição está do resultado esperado. Então, para isso, surge a função de perda (ou função de custo[2]), calculando um valor que quantifica o erro do modelo para um determinado conjunto de dados. O objetivo de todo treinamento é ajustar os parâmetros da rede para minimizar o valor dessa função (Goodfellow; Bengio; Courville, 2016).

A escolha da função de perda está atrelada à natureza do conjunto de dados e a como o modelo deve resolvê-lo (regressão ou classificação). Para isso, existem algumas funções utilizadas, como:

- Erro Quadrático Médio: *Mean Squared Error* (MSE), definida pela Equação 6, é a função de perda mais comum para tarefas de regressão. Ele calcula a média dos quadrados da diferença entre os valores preditos e os valores reais.

$$L_{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{6}$$

  Onde:

  - n é o número de amostras dos dados (*batch*).
  - y_i é o valor real da i-ésima amostra.
  - $\hat{y}_i$ é o valor predito pelo modelo para a i-ésima amostra.

  Ao elevarmos o erro ao quadrado, a função penaliza erros maiores e garante o resultado sempre positivo.

- Entropia Cruzada Binária: *Binary Cross-Entropy*, utilizada para classificações binárias, onde a saída deve ser uma de duas classes (como 0 ou 1 para “Benigno" e “Malicioso"), é a função de perda mais comum para tarefas de classificação. Tipicamente utilizada em conjunto com uma função de ativação sigmoide logística na camada de saída (pois produz uma probabilidade entre 0 e 1).

[2] na literatura encontrado muitas vezes em inglês como *loss function* ou *cost function*

A função mede a diferença entre a distribuição de probabilidade real e a distribuição predita pelo modelo. A Equação 7 a define.

$$L_{\text{BCE}} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right] \tag{7}$$

Onde:

- n é o número de amostras.
- y_i é o rótulo real (0 ou 1) da i-ésima amostra.
- $\hat{y}_i$ é a probabilidade predita pelo modelo que a amostra i pertence a classe correta.

- Entropia Cruzada Categórica: *Categorical Cross-Entropy* é a generalização da versão binária para tarefas de classificação multiclasse. Idealmente pareada com a função *Softmax* na camada de saída (pois gera um vetor de probabilidade onde a soma dos elementos é 1).

  A função compara a distribuição da probabilidade predita pelo modelo com a distribuição real, que geralmente é representada por um vetor *one-hot encoded*[3]. A formula é vista na Equação 8.

$$L_{\text{CCE}} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c}) \tag{8}$$

  Onde:

  - n é o número de amostras.
  - C é o número total de classes.
  - $y_{i,c}$ é um valor binário que indica se a classe c é a classe correta para a amostra i.
  - $\hat{y}_{i,c}$ é a probabilidade predita pelo modelo de que a amostra i pertença à classe c.

2.4.4 Backpropagation

Abreviação de *backward propagation of errors*, *Backpropagation* é o algoritmo que permite o treinamento eficiente de redes neurais profundas, criado por Rumelhart, Hinton & Williams (1986). Após a passagem de dados pela rede (*forward pass*), resultando em uma predição e no cálculo do erro através da função de custo L, o *backpropagation* realiza o próximo passo: calcula o gradiente[4] da função de custo em relação a cada um dos parâmetros da rede (∇L).

[3] Vetor que possuí zero em todas as posições exceto uma, nesse caso representando a classe correta

[4] Gradiente é um vetor que indica a direção e a magnitude do ajuste necessário em cada peso da rede para reduzir o erro do modelo. Para minimizar esse erro, o treinamento ajusta os pesos na direção oposta à do gradiente(Rumelhart; Hinton; Williams, 1986).

Matematicamente, aplica-se a regra da cadeia de forma recursiva. Iniciando pela camada de saída e calculando como o erro muda em relação a suas ativações, então propaga o erro para trás, camada por camada. Em cada uma das camadas, calcula como o erro se modifica em relação aos seus pesos e vieses (*biases*) reutilizando os gradientes calculados da camada seguinte. Esse processo permite determinar a contribuição de cada parâmetro no erro final (Goodfellow; Bengio; Courville, 2016).

2.4.5 Otimização

Uma vez que o *backpropagation* realizou o cálculo do gradiente ($\nabla_W L$), o algoritmo de otimização utilizará essa informação para atualizar os pesos, tentando encontrar o custo mínimo (Goodfellow; Bengio; Courville, 2016).

$$W_{\text{novo}} = W_{\text{antigo}} - \eta \cdot \nabla_W L \tag{9}$$

A Equação 9 denota uma regra de atualização, onde:

- W representa os pesos da rede; em uma formulação completa, os vieses também compõem o conjunto de parâmetros treináveis e podem ser incluídos em θ.
- η (eta) é a taxa de aprendizado (*learning rate*), que determina o tamanho do passo dado na direção oposta ao gradiente.
- $\nabla_W L$ é o gradiente da função de custo em relação aos pesos, calculado pelo *backpropagation*.

Tendo em vista que a Equação 9 denota a regra mais básica de atualização (SGD – *Stochastic Gradient Descent*), outras metodologias foram desenvolvidas para otimizar e superar desafios encontrados.

A notação padrão para denotar as funções de atualização será:

- θ_t: Vetor de parâmetros treináveis (pesos e vieses) no passo de tempo t.
- $g_t = \nabla_\theta L(\theta_t 1)$: Gradiente da função de custo L em relação aos parâmetros θ no passo t.
- η: Taxa de aprendizado.
- ϵ: Pequena constante para evitar divisão por zero.
- SGD com Momento: de forma clássica o SGD pode ter uma convergência lenta em superfícies de erro com “ravinas" (vales longos e estreitos). Adicionando momento, é possível acelerar o SGD na direção relevante e amortecer oscilações provenientes de ravinas (Qian, 1999). Para tal, adiciona-se à atualização atual uma fração da atualização anterior. O hiperparâmetro γ controla essa fração, funcionando como coeficiente de momento. A

Equação 10 descreve o cálculo do momento v_t. A atualização dos parâmetros, conforme a Equação 11, utiliza essa velocidade em vez do gradiente imediato.

$$v_t = \gamma v_{t-1} + \eta g_t \tag{10}$$

$$\theta_t = \theta_{t-1} - v_t \tag{11}$$

- Adagrad: *Adaptative Gradiente Algorithm* adapta o aprendizado para cada parâmetro de forma individual, realizando atualizações maiores para parâmetros associados a características infrequentes e menores para frequentes (Duchi; Hazan; Singer, 2011). Acumula o quadrado dos gradientes passados para cada parâmetro da matriz G_t, visto na Equação 12, em que $\odot$ denota o produto de Hadamard (multiplicação elemento por elemento). Já na regra de atualização (Equação 13), a taxa de aprendizado η é dividida pela soma acumulada, diminuindo a taxa de aprendizado para parâmetros que recebem gradientes grandes com frequência.

$$G_t = G_{t-1} + g_t \odot g_t \tag{12}$$

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t \tag{13}$$

- RMSprop: *Root Mean Square Propagation* resolve um dos problemas do Adagrad onde sua taxa de aprendizado decai rapidamente. Este modifica o Adagrad utilizando uma média móvel exponencial dos quadrados dos gradiente (ao invés de acumulá-los). Como visto na Equação 14, o acúmulo $E[g^2]_t$ é uma média ponderada entre o acúmulo anterior e o quadrado do gradiente atual, com o hiperparâmetro β controlando o decaimento. A regra de atualização (Equação 15) é parecida com a do Adagrad, utilizando a média móvel, impedindo que a taxa de aprendizado caia para zero rapidamente.

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1 - \beta) g_t^2 \tag{14}$$

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{15}$$

- Adam: *Adaptive Moment Estimation* é considerado o otimizador padrão para muitas tarefas de *Deep Learning*, combinando ideias do SGD com momento e RMSprop (Goodfellow; Bengio; Courville, 2016). Calcula uma média móvel exponencial tanto dos gradientes (primeiro momento, m_t) quanto dos quadrados dos gradientes (segundo momento, v_t), como mostram as Equações 16 e 17. Os hiperparâmetros β_1 e β_2 controlam, respectivamente, o decaimento exponencial do primeiro e do segundo momento.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{16}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{17}$$

Como m_t e v_t são inicializados em zero, serão enviesados em direção ao zero, especialmente nos primeiros passos do treinamento. Adam corrige essa característica calculando $\hat{m}_t$ e $\hat{v}_t$ nas Equações 18 e 19

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{18}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{19}$$

Finalmente, a regra de atualização do Adam utiliza esses momentos corrigidos, como visto na Equação 20.

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{20}$$

2.4.6 Modelos Transformer

A evolução de modelos de *Deep Learning* para análise de sequências, como textos ou URLs, foi marcada por RNNs (*Recurrent Neural Network*) e direcionada para modelos baseados em atenção. A arquitetura Transformer introduziu mecanismos de auto-atenção (*self-attention)*, permitindo ao modelo analisar uma sequência de uma única vez (Vaswani et al., 2017).

O BERT (*Bidirectional Encoder Representations from Transformers*) (Devlin et al., 2018) capitalizou sobre o poder dos *encoders* do Transformer para criar representações contextuais profundas. Utilizando de *Masked Language Model* (MLM) durante seu pré-treino, o modelo aprendeu a entender o significado de um *token* com base em seu contexto.

Contudo, com o grande tamanho de BERT e seu grande custo computacional, outras variantes surgiram de forma mais eficiente e especializadas. Em especial duas são mais relevantes a este trabalho:

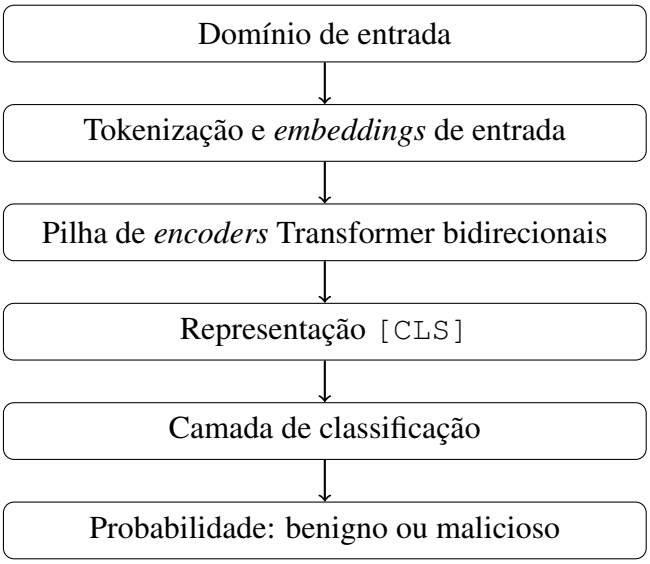
- DistilBERT: uma versão destilada do modelo BERT (Sanh et al., 2019). Utilizando da técnica denominada *knowledge distillation*, ele se torna um modelo significantemente menor e mais rápido, enquanto retém grande parte da performance do BERT.
- Canine: um modelo que opera diretamente sobre caracteres Unicode (Clark et al., 2021), eliminando a necessidade de um tokenizador com vocabulário fixo.

A arquitetura BERT utilizada neste trabalho pode ser resumida como uma pilha de *encoders* Transformer bidirecionais: a sequência de entrada é tokenizada, combinada a *embeddings* posicionais e segmentais, processada por múltiplas camadas de autoatenção e, por fim, conectada a uma cabeça de classificação para estimar a classe do domínio. A Figura Figura 4 apresenta uma visão conceitual dessa estrutura.

2.5 TRABALHOS DE BASE

Durante a pesquisa para a criação do módulo AMAZONAS, alguns trabalhos recentes se tornaram relevantes e foram utilizados como apoio e comparação. Três deles são discutidos

Figura 4 – Visão conceitual da arquitetura BERT para classificação.



Fonte: Autora, adaptado do funcionamento descrito por Devlin et al. (2018).

nesta seção: Desmet et al. (2021), Mahdaouy et al. (2024) e Alhogail & Al-Turaiki (2022).

O sistema Premadoma (Desmet et al., 2021) oferece uma abordagem preventiva: ele prevê a intenção maliciosa no momento de registro do domínio, antes que ele se torne operacional. Não depende de tráfego de DNS ou observações de uso, mas sim se baseia em padrões de registro em massa, similaridade nas informações do registrante e outras características de registro para detectar domínios de alto risco. O modelo foi implementado e avaliado durante 11 meses no registro do TLD .eu, demonstrando redução significativa de registros maliciosos, embora o artigo tenha detectado possíveis padrões de evasão e necessidade de adaptação contínua.

Em contraste, Mahdaouy et al. (2024) propõe um modelo baseado em BERT, sendo este um pré-treinado utilizando MLM (*deMasked Language Modeling*) sobre um grande corpo de domínios, URLs e DGAs. O modelo é então avaliado em tarefas binárias e multiclasse e supera modelos baseados em caracteres e outros baseados em BERT voltados para a área de segurança.

Por fim, o estudo *Improved Detection of Malicious Domain Names Using Gradient Boosted Machines and Feature Engineering* (Alhogail; Al-Turaiki, 2022) propõe um modelo que combina características léxicas do nome do domínio com *features* baseadas em *host* (como IP, localização geográfica, portas abertas, etc.). Utilizando *Gradient-Boosted Machines* (GBM), obteve acurácia de $98,8\%$.

Além disso, estudos recentes para detecção de DGA, como o NIOM-DGA (Jeremiah et al., 2025), reforçam a relevância da engenharia de atributos e da seleção de características para ganho de generalização. Esse ponto dialoga diretamente com a estratégia do AMAZONAS de combinar classificadores especializados com objetivos diferentes.

Cada um dos trabalhos traz consigo características similares ao que o módulo AMAZONAS realiza e podem servir de comparação ao seu desempenho final.

2.6 ESTUDOS RELACIONADOS

Durante o processo de desenvolvimento do módulo AMAZONAS, alguns trabalhos foram incorporados ao sistema, facilitando sua construção e permitindo a utilização de ferramentas mais complexas para atingir seus objetivos. Esta seção apresenta os trabalhos que contribuíram para a construção do sistema.

2.6.1 ENTRADA 2

ENTRADA 2 é a evolução do projeto ENTRADA (Wullink et al., 2016). Em termos práticos, o sistema converte dados capturados em PCAP para estruturas tabulares em Apache Iceberg[5], utilizando Apache Parquet para armazenamento eficiente.

A evolução do ENTRADA para ENTRADA 2 trouxe mudanças em relação à forma de *deployment* e tipos de armazenamento. De forma geral o propósito permanece o mesmo: salvar informações de DNS Passivo utilizando o menor espaço possível da forma mais eficiente possível.

Segundo o artigo publicado por Wullink et al. (2016), os requisitos funcionais são:

- Performance: sistema deve ser capaz de suportar exploração de dados em um tempo de resposta curto, mesmo quando utilizado para explorar dados referentes a grandes períodos de tempo.
- Escalabilidade: ENTRADA iniciou armazenando aproximadamente 85GB/dia por servidor autoritativo. Portanto, o sistema deve ser escalado para suportar o armazenamento sem comprometer sua performance.
- Usabilidade: Usuários devem ser capazes de realizar consultas sem atrito. Foi determinado que a utilização de uma linguagem próxima à SQL seja necessária.
- Extensibilidade: formato dos dados deve ser compatível com vários *engines* de análise de dados, evitando o travamento de software ou corporação.
- Confiabilidade: falha de qualquer máquina do *cluster* não deve resultar na perda de dados ou fazer o *cluster* se tornar inacessível.

2.6.2 DNSCheck

Inicialmente apresentado por Batista (2020), tinha como objetivo reunir classificadores de *machine learning* e disponibilizá-los por meio de APIs.

Atualmente, o projeto transformou seu objetivo em reunir uma série de ferramentas focadas para análise, observação e classificação de domínios (imagens do sistema encontram-se no Apêndice A). Constando com os seguintes módulos:

[5] Formato de alta-performance para grandes tabelas analíticas (Apache, 2025).

- Nevermore: reúne uma série de *blocklists* de *threat intelligence feeds* (fontes de inteligência de ameaças) e as enriquece utilizando consulta ativa de DNS.
- *Watch Domains*: permite que o usuário seja notificado quando algum domínio contido por uma de suas expressões regulares ou nomes de domínio registrados for adicionado em uma *blocklist* monitorada pelo Nevermore.
- *Classifiers*: classificadores de domínios baseados em IA dinamicamente modificáveis conforme pesquisas são avançadas e novos modelos são adicionados. No momento de escrita desta dissertação dois modelos estão disponíveis: Léxico e Ativo, para classificação com base em características léxicas e com base em características de DNS ativo.
- *Dashboards*: contém uma interface dinâmica baseada em pDNS (DNS passivo). Fornece informações acionáveis de segurança para públicos autorizados, como a reitoria da UNESP, a qual fornece os dados e em troca consegue monitorar *hosts* e destinos potencialmente maliciosos.

3 METODOLOGIA

O módulo AMAZONAS conta com diversos sub-módulos que, ao se combinarem realizam a classificação, atribuição de *score* e monitoramento de domínios potencialmente maliciosos. Sua intenção é atuar entre dois momentos distintos: a inicial, onde o domínio é cadastrado e o final onde se monitora o domínio.

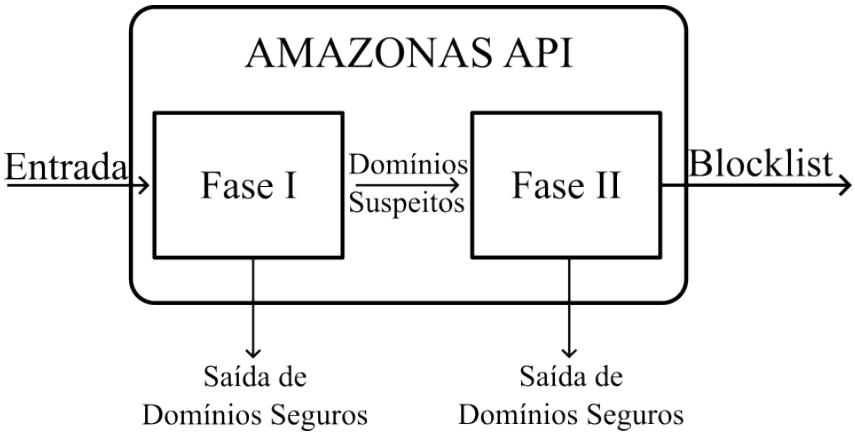
Para isso, o projeto AMAZONAS conta com duas fases principais, nomeadas de “Fase I" e “Fase II", apresentadas na Figura Figura 5. Na “Fase I", ocorre a entrada de um ou mais domínios por um *endpoint* e, após seu processamento, duas alternativas são possíveis:

- 1. O domínio, após passar por todos os classificadores, foi classificado como seguro e, então, é liberado.
- 2. O domínio, após ser processado pelos classificadores, foi considerado como potencialmente fraudulento e deve ser monitorado.

Na “Fase II", os domínios suspeitos são submetidos a um monitoramento contínuo por meio de diversas técnicas. Com base nos dados coletados, um analista decide manualmente se o domínio é, de fato, malicioso ou se representa um falso positivo.

Para apoiar essa etapa humana, foi implementada uma rotina de geração automática de *prompts* para LLMs, consolidando em um único texto as evidências das Fases I e II e a recomendação inicial do sistema. O objetivo é reduzir o tempo de triagem, padronizar o contexto entregue ao analista e tornar a decisão final mais consistente em cenários de alto volume. A implementação foi realizada por meio de um script em Python, apresentado no Apêndice D, e exemplos de saída em Markdown são apresentados nos Apêndices E e F.

Figura 5 – Funcionamento geral do projeto AMAZONAS.



Fonte: Autora.

3.1 BANCO DE DADOS

Tendo em vista que, o módulo AMAZONAS realiza suas ações em duas etapas diferentes, podemos separar seu banco de dados em duas partes:

- Banco de classificadores: contém as informações dos classificadores utilizados no módulo. São necessárias para a fácil alteração de classificadores, ou para alteração de URL. Em base, os campos contidos são:
  - `name`: nome do classificador.
  - `URL`: armazena a URL do classificador.
  - `type`: tipo variável que permite distinguir classificadores (Ex: Léxico, DGA, Typo).
  - `data_name`: retorno do classificador em sua API (Ex: *result*, *percent*, *resultado*).
- Banco de dados de domínio: contém as informações armazenadas de cada domínio a partir de seus coletores. Estes dados não são alteráveis de forma direta ao usuário e servirão principalmente para a criação de *datasets* (ou para futuras classificações). Os campos contidos variam de acordo com o coletor, a Figura Figura 6 demonstra o UML do banco de dados direcionado à armazenar os dados de domínio.

3.2 DATASETS

A construção de modelos de classificação eficazes depende, fundamentalmente, da qualidade e da representatividade dos dados de treinamento e avaliação. Esta seção detalha, principalmente, a origem e a composição de cada um dos *datasets* utilizados nesse trabalho. Devido à diversidade de classificadores trabalhados, diversos *datasets* devem ser utilizados para casos específicos.

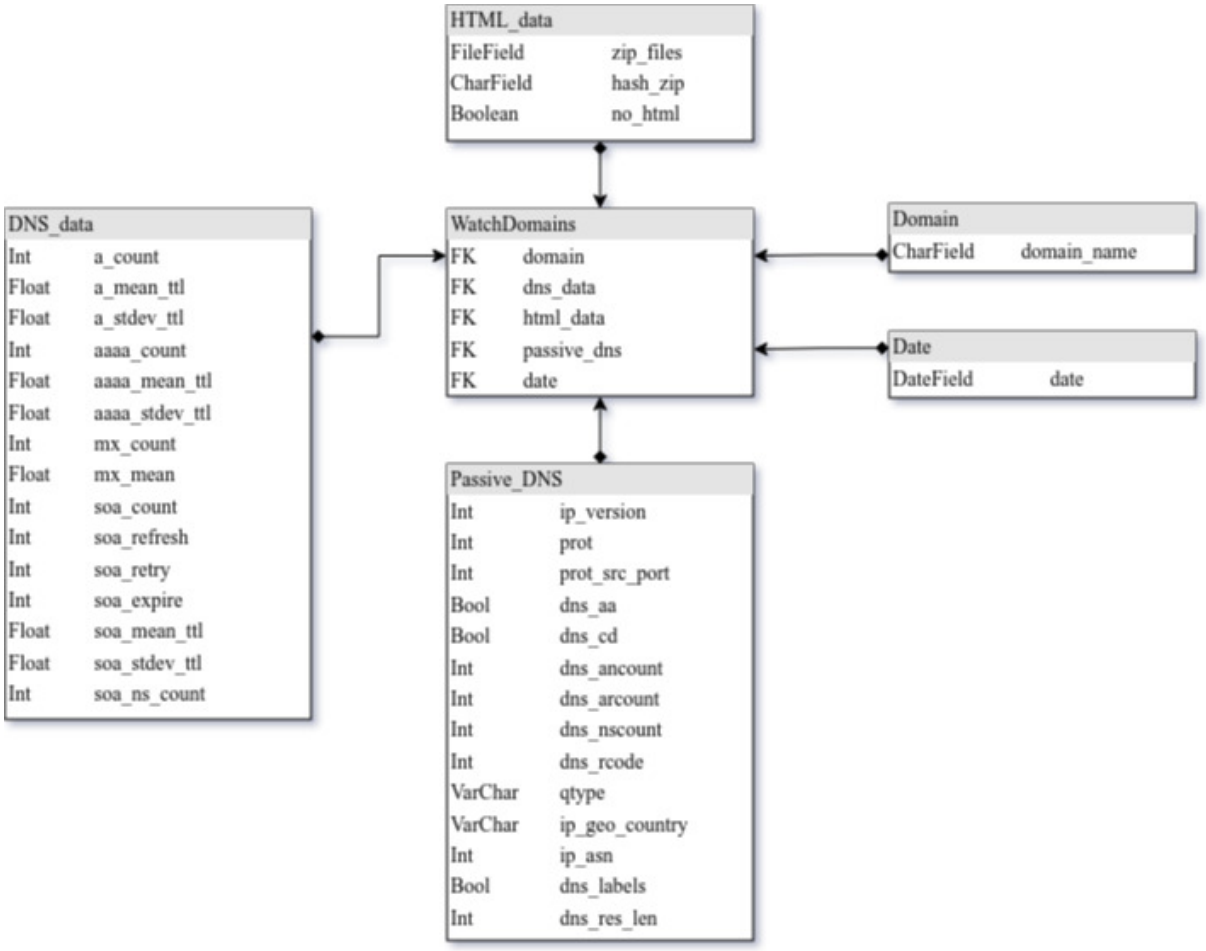
3.2.1 DGAs

O *dataset* utilizado para o classificador de DGA é demonstrado na metodologia descrita em Gregório (2025), utilizando fontes distintas para as classes benignas e maliciosas.

Para a classe de domínios benignos, a lista Majestic Million foi utilizada, sendo esta, uma fonte de dados publicamente disponível que cataloga um milhão de domínios com o maior tráfego na Internet. Para a classe maliciosa, foram agregados domínios provenientes de três fontes especializadas em DGA: DGIArchive, FKIE e uma base dados construída a partir da análise de pDNS.

Já o pré-processamento dos dados envolveu a vetorização dos nomes de domínio. Cada domínio foi truncado ou preenchido para um comprimento fixo de 70 caracteres, e cada caractere foi então convertido para seu valor numérico correspondente da tabela ASCII. Nesse classificador de DGA, portanto, não foi utilizada uma camada de *embedding* por *look-up table*;

Figura 6 – Representação do banco de dados do projeto AMAZONAS Fase II.



Fonte: Autora.

a entrada corresponde diretamente ao vetor numérico derivado dos caracteres. O *dataset* final foi estruturado com as seguintes colunas: o nome do domínio original, o nome da família do DGA, um rótulo binário com 0 para benigno e 1 para DGA e 70 colunas representando o vetor de características numéricas.

3.2.2 Typosquat

O conjunto de dados de origem da Anvilogic[1] é disponibilizado na plataforma HuggingFace, e é estruturado para aprendizado de similaridade. Sua composição se baseia em pares de domínios, contendo um domínio “âncora", que representa os domínios legítimos, e um domínio “‘positivo", que representa uma variação maliciosa gerada via *typosquatting*.

A geração de exemplos de *typosquatting* neste *dataset* foi feita por meio do algoritmo `ail-typo-squatting`. Este algoritmo aplica, sistematicamente, técnicas de mutação sobre domínios legítimos para gerar variações fraudulentas. Por exemplo, ao solicitarmos a geração

[1] https://huggingface.co/Anvilogic/Embedder-Typosquat-Detect

de *typosquat* para `acmesecurity.org`, os domínios gerados serão: `acmesecurity.app`, `acmesecurity.com`, `acmesecurlty.org`, etc. O *dataset* final foi dividido em aproximadamente 40 mil amostras para treinamento e 10 mil para teste.

3.2.3 Léxico

Para os experimentos de classificação léxica, que serviram tanto como *baseline* quanto para testes com os modelos Transformer, um *dataset* de classificação binária foi construído. A estrutura final do *dataset* consiste em duas colunas: uma contendo o nome do domínio e outra contendo seu rótulo.

A classe de domínios benignos foi amostrada a partir da lista Majestic Million. Já para a classe de domínios maliciosos, foi utlizada uma compilação de listas de bloqueio DNS de alta reputação, mantida pelo projeto "HaGeZi's Threat Intelligence Feeds DNS Blocklist"[2] no GitHub, que agrega múltiplos *feeds* de inteligência de ameaças. Como corte de dados foram retirados 250 mil domínios benignos da Majestic Million e 250 mil domínios maliciosos da TIF, o que representa um total de 500 mil domínios totalmente balanceados para a realização de treinamento e testes.

3.3 FASE I

A "Fase I" pode ser simplificada como um conjunto de classificadores modulares de IA capazes de identificar características específicas de domínios potencialmente maliciosos, como ilustrado na Figura Figura 7. Em outras palavras, combina-se classificadores especializados em domínios gerados por algoritmos, técnicas de *typosquat* e características léxicas gerais. A modularidade decorre da facilidade de introdução de novos classificadores, pois o sistema permite inserir novos módulos por meio de APIs de fácil implementação, como *FastAPI*[3]. A saída depende dos resultados agregados: se nenhum classificador indicar suspeita, o domínio é tratado como seguro e pode ser adicionado à *allow-list*; caso contrário, o domínio é encaminhado para a "Fase II", onde será analisado continuamente.

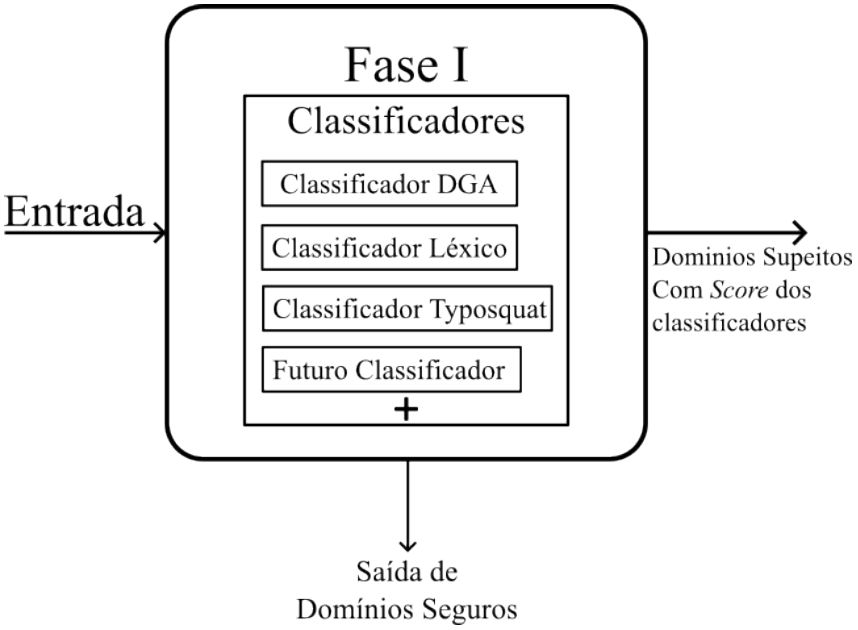
3.3.1 Classificadores

Cada um dos classificadores empregados de forma conjunta possui uma especialidade. Assim como os crimes cibernéticos são executados e estudados de forma diversa, o mesmo ocorre com o abuso de DNS. Um sistema de classificação geral pode não conseguir identificar, de maneira desejada, por exemplo, um domínio gerado por algoritmo, e um classificador de DGA pode não conseguir identificar um domínio que utiliza da técnica de *typosquat*.

[2] https://github.com/hagezi/dns-blocklists

[3] *Web framework* para construção de APIs em Python, com forte suporte a tipagem estática.

Figura 7 – Fase I do módulo AMAZONAS detalhada.



Fonte: Autora.

Portanto, foram utilizados classificadores para dois tipos importantes de domínios maliciosos: DGA e *Typosquat*. Além disso, um classificador geral foi criado na tentativa de cobrir a maior área de ataque possível.

Classificador de DGA

Para detecção de DGA, foi implementada uma adaptação do modelo proposto por Gregório (2025). O modelo original utiliza uma arquitetura de *Deep Learning* baseada em uma camada de *embedding* sequenciada por múltiplas camadas convolucionais e de *max pooling*, utilizando a função de ativação ReLU em suas camadas ocultas.

O treinamento foi realizado sobre o *dataset* detalhado na Seção 3.2, com divisão de 500 mil amostras para treinamento e 500 mil para teste. O processo de treinamento incremental consistiu em 12 etapas, onde, em cada uma, o modelo foi treinado com 500 mil domínios benignos e uma coleção de domínios DGA correspondente a 15 dias de consulta.

Para que o modelo se encaixasse no projeto AMAZONAS, houve a integração de uma API feita a partir do Django REST Framework, facilitando seu uso e consulta.

Classificador de *typosquat*

Foi adaptado de um dos modelos classificadores utilizados pela empresa Anvilogic[4].

É baseado no modelo pré-treinado da empresa Google: o Canine-c, *fine-tuned* para detecção de *typosquat*, ou seja, um modelo de Transformer. O modelo foi treinado com base no *dataset* da

[4] Embora não possua uma publicação sobre o modelo, seus dados estão contidos na plataforma HuggingFace.

empresa Anvilogic, também disponível na plataforma HuggingFace. Os dados de treinamento foram separados em aproximadamente 40 mil domínios para treino e 10 mil para teste, sendo seu *dataset* descrito na Subseção 3.2.

Como sugestão, os criadores do modelo utilizam similaridade por cosseno. Contudo, a metodologia original utiliza o método padrão de busca de similaridade, que, em casos extremos, se torna ineficiente. Portanto, é sugerida a introdução do algoritmo FAISS (Douze et al., 2024), utilizado para busca de similaridades. A implementação completa (incluindo a formulação de API) pode ser vista no Apêndice C, na Listagem C.1.

Classificadores léxicos

Nesta seção, é detalhado o procedimento para a experimentação de análise léxica de nomes de domínio. A metodologia foi dividida em duas frentes, aplicando, de um lado, algoritmos clássicos de Aprendizado de Máquina, que servem de *baseline*, e outro lado aplicando e realizando testes com modelos de *Deep Learning* baseados em Transformer (neste caso, a família de modelos BERT). Todos os modelos foram treinados e avaliados sobre o mesmo conjunto de dados, demonstrados na Seção 3.2.

Para o treinamento e avaliação dos modelos, todos seguiram o mesmo padrão de divisão para teste e treino: 80% para treino e 20% para teste. A padronização garante uma proporção de domínios maliciosos e benignos iguais para os modelos testados, na tentativa de minimizar vieses. Antes do treinamento, as divisões foram embaralhadas na tentativa de evitar que a ordem dos dados influenciasse no aprendizado.

Como *baseline* para o trabalho, foram utilizados modelos de aprendizado de máquina cuja eficácia já foi previamente comprovada. Em Romanholo & Cansian (2024), o modelo utilizado é baseado em *Random Forest*. Para isso os nomes de domínio são convertidos em valores numéricos através de extração de características léxicas, feita à mão. As seguintes características são extraídas:

- O tamanho do nome de domínio;
- Entropia de Shannon;
- Quantidade de pontos;
- Quantidade de hífens;
- Quantidade de números;
- Quantidade de vogais;
- Quantidade de consoantes;
- TLD (*Top-Level Domain*) fora do local adequado;

- Quantidade de vogais consecutivas;
- Quantidade de números consecutivos;
- Quantidade de consoantes consecutivas;
- Tamanho da maior palavra;
- Porcentagem de números relativo ao tamanho do domínio;
- Porcentagem de consoantes relativo ao tamanho do domínio;
- Porcentagem de vogais relativo ao tamanho do domínio;
- Porcentagem da maior palavra relativo ao tamanho do domínio.

A segunda abordagem explora a utilização de modelos de linguagem pré-treinados e bem consolidados para classificação de domínios. Embora nomes de domínio não constituam linguagem natural e semântica, são, fundamentalmente, sequências de caracteres que contêm padrões léxicos. A hipótese é que modelos que seguem mecanismos de atenção podem ser capazes de identificar padrões para identificar domínios benignos e maliciosos. Ao tratar um domínio como uma sequência, o modelo Transformer pode aprender que certas combinações de caracteres ou subdomínios, mesmo que nunca vistas antes, estão altamente relacionadas à atividades maliciosas.

Para testar tal hipótese, três modelos foram selecionados com base na arquitetura Transformer e ajustados para classificação binária: BERT, DistilBERT e Canine-c

O modelo BERT, especificamente a versão `bert-uncased`, foi utilizado como ponto de partida e base para os outros dois modelos. A expectativa inicial para BERT era de um desempenho moderado, servindo principalmente como referência para seus modelos filhos.

Os hiperparâmetros de treinamento de cada um dos modelos podem ser vistos no Apêndice B com as Tabelas 7, 8 e 9.

O código do classificador selecionado pode ser visto no Apêndice C na Listagem C.2.

Na implementação da Fase I, a orquestração foi definida para operar com classificadores externos heterogêneos, configurados por prioridade, *endpoint*, tempo limite e estrutura de envio (domínio a domínio ou em lote). Após a consulta aos classificadores ativos, as respostas são normalizadas para o formato comum {`domain`, `is_malicious`, `score`, `details`}, em que `score` representa sempre probabilidade de malícia no intervalo $[0, 1]$.

Para garantir coerência semântica entre modelos distintos, cada classificador recebe uma regra de normalização específica. No classificador DGA, o *score* é utilizado diretamente. No classificador léxico (DistilBERT), quando o rótulo retornado é benigno (`LABEL_0`), aplica-se inversão da confiança, de modo que `score = 1 - confidence`; quando o rótulo é malicioso (`LABEL_1`), usa-se `score = confidence`. No classificador Typosquat, o *score* corresponde à similaridade apenas quando há indicação de typosquatting; caso contrário, o valor

é definido como zero, indicando ausência de evidência positiva de *typosquatting* e evitando ambiguidade com valores ausentes do tipo `null`.

Com os resultados normalizados, a decisão segue estratégia de pior caso (*worst-case*), utilizando o maior *score* entre classificadores como *score* final do domínio. A classificação operacional adota três faixas empíricas: *allow_list* para valores inferiores a 0.5, *gray_list* para o intervalo $[0.5, 0.75]$ e *block_list* para valores superiores a 0.75. Esses limiares foram definidos como escolha de engenharia para separar casos de baixa confiança e, especialmente, preservar uma zona cinza destinada à coleta de evidências adicionais.

Domínios em *gray_list* são encaminhados automaticamente para a Fase II por meio do conector entre fases, que garante a criação (ou atualização) do domínio monitorado e do ciclo temporal corrente no banco de monitoramento.

3.4 FASE II

A “Fase II" implementa diversas táticas para averiguação constante de potencial maliciosidade de um domínio. Portanto, é um módulo que age ao longo do tempo como visto na Figura Figura 8. Diariamente, são coletadas as seguintes informações de domínios registrados:

- Presença de página web: são extraídas e comprimidas todas as informações contidas na página principal do domínio monitorado, caso exista. Estas informações ficarão salvas enquanto este domínio estiver sendo monitorado. O armazenamento não se tornou um problema pois, diariamente, cerca de 500 KB a 1 MB são consumidos para armazenar todo o conteúdo da página inicial.

- Dados de DNS: informações como os registros A, AAAA e MX são armazenadas no banco de dados para análises posteriores.

- Dados de PDNS: é verificado se usuários estão acessando o site e se estes usuários são conhecidos como potenciais agentes maliciosos, como endereços IP de *Botnets*.

É esperado que a combinação dessas informações deixe o analista de segurança responsável com confiança para determinar se um domínio é realmente malicioso ou se apenas se tratava de um falso positivo.

Na implementação final, a Fase II foi concebida como um processo de enriquecimento contínuo e orientado a evidências, com armazenamento por ciclos diários. O modelo temporal utiliza o par `(domain, date)` como unicidade operacional, permitindo acompanhar a evolução do mesmo domínio ao longo de múltiplos ciclos sem perda de histórico.

No coletor Web, o conteúdo é compactado e deduplicado por *hash* SHA256 do arquivo ZIP. Para domínios inacessíveis (`no_html=True`), o fluxo reutiliza um registro dedicado de ausência de conteúdo; quando necessário, cria entrada com *hash* único derivado de domínio e tempo, evitando colisões de unicidade e mantendo consistência transacional.

No coletor de DNS passivo, os dados do ENTRADA2 são agregados por domínio e IP consultante, com seleção dos 20 IPs mais frequentes por domínio em cada janela de coleta. Esse desenho prioriza sinais acionáveis (distribuição geográfica, diversidade de ASN, volume e códigos de resposta) em vez de armazenar eventos brutos em larga escala.

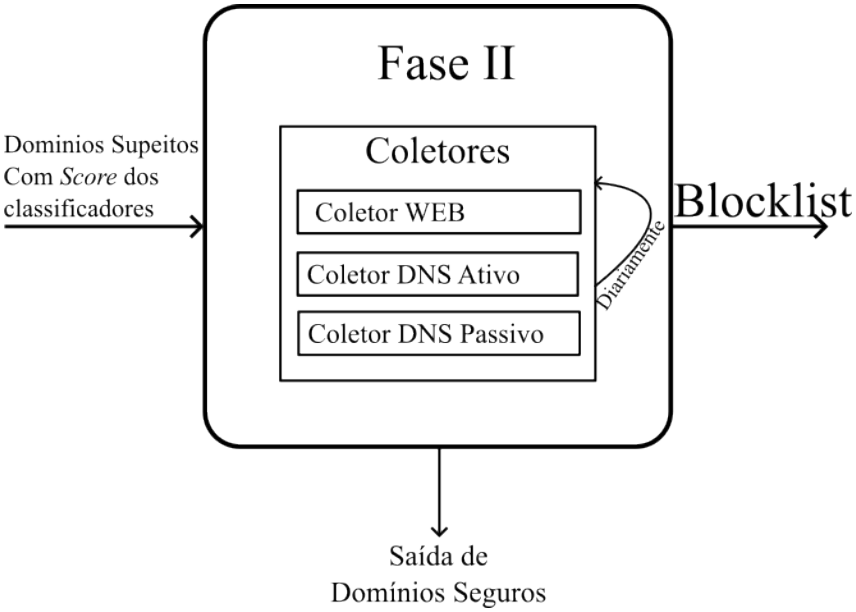
Além da coleta, foi incorporado um cálculo de risco estatístico a partir dos indicadores de pDNS. O *score* bruto (0 a 100) combina quatro componentes: diversidade geográfica de consultas, diversidade de ASNs, taxa de NXDOMAIN e volume de consultas. Em seguida, o valor é normalizado para $[0, 1]$ e utilizado como evidência complementar ao *score* da Fase I.

Operacionalmente, as rotinas de coleta são executadas de forma assíncrona com Celery e retentativa exponencial, aumentando robustez frente a falhas transitórias de rede e serviços externos. O conjunto contempla tarefas de DNS ativo por domínio, coleta web por domínio, coleta pDNS em lote para domínios monitorados, coleta pDNS pontual e um orquestrador periódico de monitoramento.

Também foi disponibilizado um *endpoint* de enriquecimento sob demanda, permitindo que analistas disparem manualmente coletas adicionais de DNS ativo, web e pDNS para um domínio específico quando necessário.

Por fim, os dados consolidados das Fases I e II são utilizados na geração de um *prompt* estruturado para LLM, com o objetivo de apoiar a etapa de validação humana final. O *prompt* organiza: (i) resultados da Fase I e limiares de interpretação, (ii) recomendação estatística com fatores explicativos, (iii) evidências de DNS ativo, web e pDNS, e (iv) solicitação de parecer final estruturado (*allow*, *block* ou *investigate*), reduzindo o esforço manual de correlação de evidências.

Figura 8 – Fase II do módulo AMAZONAS detalhada.



Fonte: Autora.

3.4.1 Extração Web

Diariamente, o módulo de extração Web realizará os seguintes processos:

- Verificando a presença de página web: utilizando-se da biblioteca Python *Requests*, é realizada uma requisição `GET /`, com acompanhamento de redirecionamentos HTTP quando aplicável. Códigos 2xx indicam página acessível; códigos 3xx são analisados pelo destino final, pois um redirecionamento também pode levar a conteúdo malicioso; e códigos 4xx/5xx ou falhas de conexão indicam indisponibilidade no momento da coleta. Se isso ocorrer, o domínio é marcado como não possuindo página web acessível naquele ciclo.

- Se constatada a presença de página web: a página é baixada e, utilizando-se da biblioteca *BeautifulSoup*, todos os *assets*[5] indexados são armazenados em memória.

- Geração de Hash: para que não haja a repetição constante de páginas salvas diariamente, é gerado um *hash* utilizando o algoritmo SHA256, com o qual é verificada a existência prévia de um *hash* idêntico. Isto é, se uma página for salva em um dia e, em outro dia, essa página permanecer idêntica, seus *hashes* também serão idênticos. Com isso, é possível salvar apenas páginas que foram modificadas, reduzindo drasticamente o armazenamento utilizado. Após a verificação de existência, os dados em memória são salvos no banco de dados e os seguintes metadados e informações são aplicados para cada uma das possibilidades:

  - `nome`: representa o nome dado ao arquivo *Zip*, será `None` se a página não existir ou se existir algum *hash* idêntico.
  - `content`: conteúdo do *zip*, será `None` se não existir ou se existir algum *hash* idêntico.
  - `no_html`: será `True` ou `False` se existir ou não a página web, respectivamente.
  - `hash_zip`: contém o *hash* do arquivo zip quando a página web existe. Será utilizado para verificação de não existência no banco de dados.

3.4.2 Extração de DNS ativo

A extração ativa de informações de DNS permite a classificação de determinado domínio em classificadores baseados em métricas de DNS ativo. Portanto, a extração de informações a partir de consultas ativas no DNS ocorre da seguinte maneira:

- É determinado o servidor recursivo do qual serão extraídas tais informações. Durante o desenvolvimento, foi percebido que muitos servidores bloqueiam solicitações em massa,

[5] Representa todos os componentes que juntos formam uma página web

o que isso indica que a expansão indeterminada do sistema AMAZONAS seria comprometida e, portanto, uma alternativa deveria ser encontrada. Logo, uma forma de circunvir tais limitações é a utilização de DoH (*DNS over HTTPS*) que, na maioria dos casos, permite a realização de milhares (até milhões) de consultas por segundo.

- Em seguida é necessário definir os tipos de registros e suas informações a serem extraídas – na Seção 2.1.1 são definidos os tipos mais comuns – sendo elas:
  - A quantidade de registros MX e seu TTL médio
  - A quantidade de registros A, a média e o desvio padrão de seu TTL e a contagem de CNAME em A.
  - A quantidadde de registros AAAA, a média e o desvio padrão de seu TTL.
  - A quantidade de registros SOA, seu tamanho, tempos de *refresh*, *retry* e *expire*, a média e o desvio padrão de seu TTL e a quantidade de NS (*name servers*) registrados.
- Com isso, as informações são salvas, cada uma em seu campo adequado no banco de dados, com descarte de valores nulos.

3.4.3 Extração de DNS Passivo

A obtenção de dados passivos se torna essencial quando utilizamos as informações contidas como dados acionáveis. Os dados extraídos são obtidos diretamente do sistema ENTRADA 2 e são baseados em Silva (2022), o que se da pela possibilidade de automação futura de análise de DNS passivo.

Os dados coletados são referenciados na Tabela 3. Cada campo na tabela é diretamente relacionado ao banco de dados mencionado em 3.1.

Tabela 3 – Campos do conjunto de dados

Feature	**Descrição**	**Tipo**
ip_version	Versão do IP	Inteiro
prot	Protocolo TCP ou UDP	Inteiro
prot_src_port	Porta de origem do protocolo IP	Inteiro
dns_aa	Resposta autoritativa	Booleano
dns_cd	*Checking Disable*[6]	Booleano
dns_ancount	Contagem de respostas	Inteiro
dns_arcount	Contador de número de respostas opcionais	Inteiro
dns_nscount	Contador de número de servidores autoritativos	Inteiro
dns_rcode	Código de resposta DNS	Inteiro
dns_qtype	Tipo de *Query* DNS	String
ip_geo_country	País de origem da requisição	String
ip_asn	*Autonomous System Number* de origem	Inteiro
dns_labels	Títulos do DNS	Booleano
dns_res_len	Tamanho da resposta DNS	Inteiro

Fonte: Adaptado de Wullink et al. (2016), Silva (2022).

4 RESULTADOS

Nesta seção são apresentados os resultados gerais do módulo AMAZONAS. Tendo em vista que o módulo contém duas fases distintas, seus resultados também são apresentados separadamente. A subseção de classificadores traz os resultados para cada um dos classificadores testados, enquanto na subseção de coletores são demonstrados os resultados parciais de informações acionáveis.

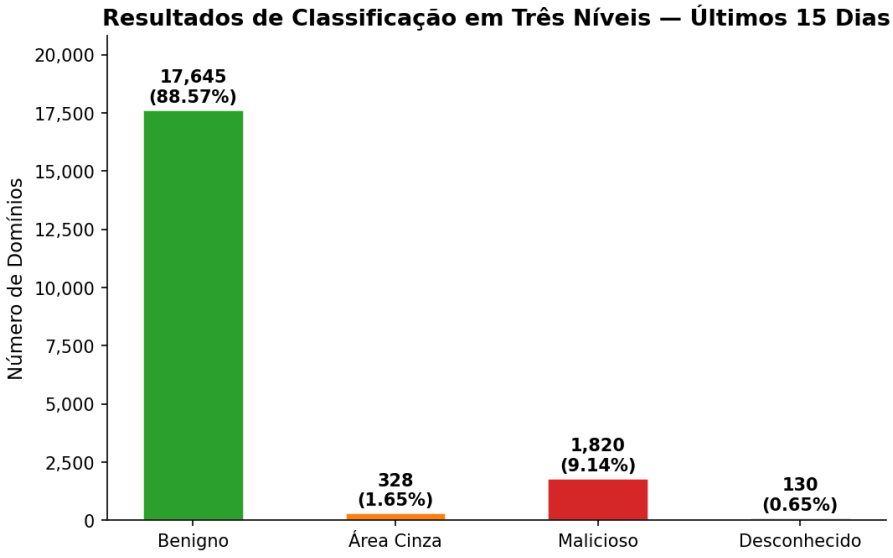
4.1 VISÃO GERAL DOS RESULTADOS OPERACIONAIS

Para avaliar o comportamento integrado do AMAZONAS em cenário de uso contínuo, foi realizado um teste operacional com aproximadamente 20 mil domínios ao longo de 15 dias. A análise considerou: (i) a classificação final de cada domínio, (ii) sua alocação nas listas operacionais, (iii) a distribuição dos *scores*, (iv) o classificador que mais influenciou a decisão e (v) o volume encaminhado para a Fase II.

4.1.1 Distribuição de classificação e listas

A Figura Figura 9 apresenta a distribuição por classe de decisão (benigno, área cinza, malicioso e desconhecido). Em termos agregados, 88.57% dos domínios foram classificados como benignos, 9.14% como maliciosos, 1.65% permaneceram em zona cinza e 0.65% ficaram como desconhecidos.

Figura 9 – Distribuição dos domínios por classe de classificação.

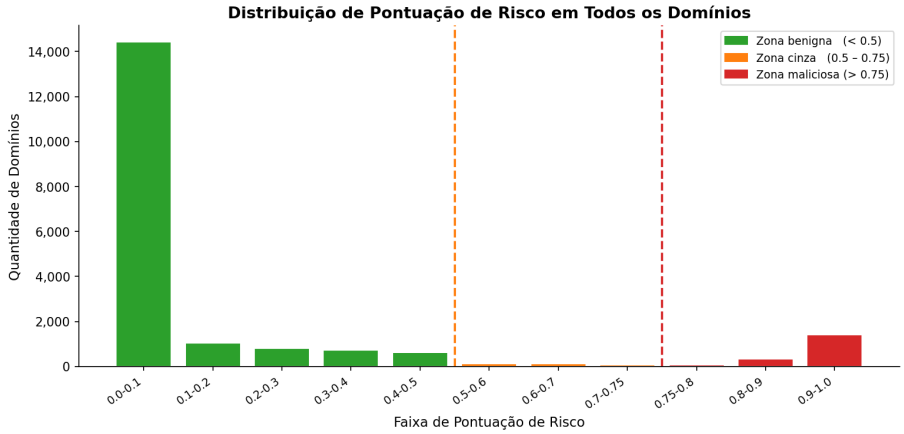


Fonte: Autora.

4.1.2 Distribuição de score e acionamento de classificadores

Na Figura Figura 10, observa-se a distribuição dos *scores* no intervalo $[0, 1]$. A maior concentração ocorre nas faixas de baixa pontuação, com predominância no intervalo de 0.0 a 0.1, enquanto as faixas de alta pontuação concentram os casos potencialmente maliciosos.

Figura 10 – Distribuição da pontuação final de classificação no intervalo $[0, 1]$.



Fonte: Autora.

4.1.3 Encaminhamento para a Fase II

A Figura Figura 11 apresenta a distribuição do classificador que mais influenciou o resultado final. O classificador léxico foi o mais frequente (57.22%), seguido do classificador de DGA (41.90%), enquanto o classificador de Typosquat atuou em menor proporção (0.88%), comportamento coerente com a natureza mais específica desse tipo de ameaça.

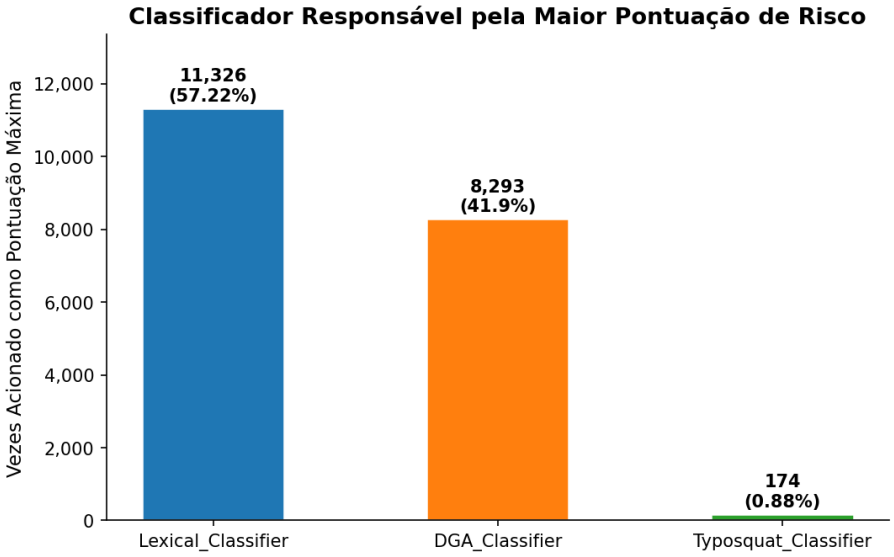
Por fim, a Figura Figura 12 resume a proporção de domínios sinalizados para a Fase II em comparação aos não sinalizados. Apenas 1.65% dos casos foram encaminhados para análise adicional, indicando que o módulo mantém a revisão humana como etapa de exceção. Esse resultado é operacionalmente relevante, pois reduz o volume diário de triagem manual e torna viável a análise especializada apenas nos casos ambíguos.

4.2 RESULTADOS DA FASE I: CLASSIFICADORES

A avaliação da Fase I concentrou-se em aferir o desempenho individual de cada classificador especializado, e logo após, colocá-los lado a lado para uma comparação global.

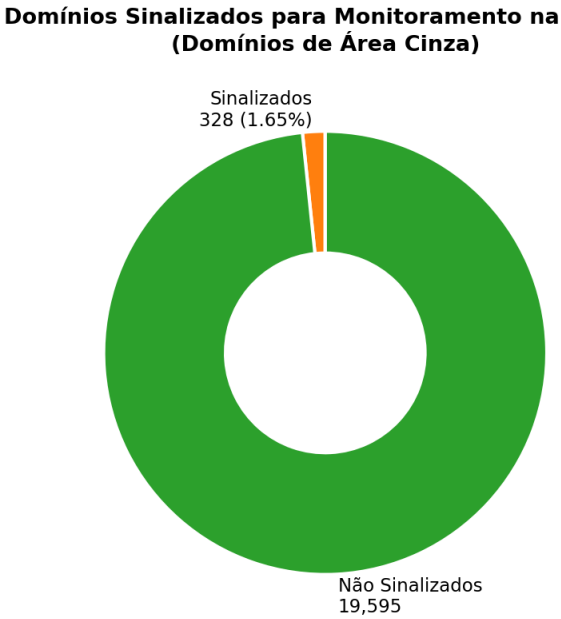
Os experimentos desta seção foram conduzidos em ambiente offline, isto é, sobre bases previamente coletadas e separadas em conjuntos de treino e teste, sem execução em tempo real contra fluxo contínuo de registros de domínio. Essa configuração permitiu comparar os classificadores em condições controladas, medir desempenho de forma reprodutível e isolar a avaliação dos modelos de variações externas de rede, disponibilidade de APIs ou latência operacional.

Figura 11 – Distribuição do classificador acionado na decisão final.



Fonte: Autora.

Figura 12 – Proporção de domínios encaminhados e não encaminhados para a Fase II.



Fonte: Autora.

Assim, os resultados devem ser interpretados como validação experimental dos classificadores, enquanto a execução em produção depende da integração com a rotina de monitoramento descrita na Fase II.

4.2.1 Classificador DGA

Utilizando métricas padrão, o modelo desenvolvido por Gregório (2025) demonstrou acurácia de 98%, precisão de 97% e *recall* de 97%. Neste ponto, cabe destacar que este classificador foi reavaliado no escopo deste trabalho utilizando o *dataset* adotado na Fase I do AMAZONAS, de modo que os resultados apresentados refletem a avaliação desta dissertação e não apenas os números reportados no trabalho original.

4.2.2 Classificador Typosquat

Baseado na arquitetura Transformer Canine-c, apresentou resultados robustos que são detalhados na Tabela 4. Em termos práticos, os valores de F1-score, precisão e *recall* acima de 0.90 indicam bom equilíbrio entre detecção de casos maliciosos e controle de falsos positivos, o que é essencial para reduzir alertas indevidos em um cenário operacional.

Tabela 4 – Métricas de avaliação do modelo Typosquat

Métrica	**Valor**
Precisão Média (Average Precision)	0.9307
F1-score	0.9113
Precisão	0.9197
Revocação (Recall)	0.9031
Threshold	0.6409

Fonte: Autora.

4.2.3 Classificadores Léxicos

Para a tarefa de classificação léxica geral, foram avaliadas e comparadas duas abordagens em arquitetura Transformer: BERT e DistilBERT. Os resultados apresentados na Tabela 5 revelam uma diferença drástica de desempenho.

O modelo BERT obteve uma acurácia de aproximadamente 50%, próxima a uma classificação aleatória, com F1-score e precisão baixos. Esse resultado sugere necessidade de revisão de configuração de treinamento e de pré-treinamento mais específico para domínios, antes de descartar definitivamente seu uso nesse contexto.

Em contrapartida, o modelo DistilBERT alcançou uma acurácia de aproximadamente 82% e um F1-Score de 0.8105, demonstrando ser uma alternativa mais eficaz. Além disso, se mostrou

7 vezes mais rápido quando comparado ao seu modelo base, BERT. A eficiência pode se tornar algo crítico se muitos domínios forem analisados em *batch*.

Tabela 5 – Métricas de avaliação dos modelos BERT e DistilBERT

Métrica	**BERT**	**DistilBERT**
Loss de avaliação	0.7786	0.3927
Tempo de preparação do modelo	0.0029 s	–
Acurácia	50.63%	82.38%
F1-score	0.3404	0.8105
Precisão	0.2564	0.8651
Recall	50.63%	76.23%
Tempo total de avaliação	603.702 s	79.767 s
Amostras processadas por segundo	154.275	1167.601
Etapas por segundo	9.642	18.253

Fonte: Autora.

4.2.4 Resultado geral dos classificadores

A análise conjunta dos resultados permite extrair conclusões importantes para a validação do módulo AMAZONAS e, também, permite a sua comparação com trabalhos relacionados.

Primeiramente, é evidente que os classificadores de DGA e Typosquatting, projetados para identificar padrões de ameaças específicas, alcançaram maiores índices de performance. Isso valida a decisão de arquitetura modular com modelos variados.

Em segundo lugar, a comparação entre BERT e DistilBERT para classificação léxica mostra a visão entre complexidade e desempenho. Um modelo menor e menos específico superou um modelo massivamente maior. Além disso, sua vantagem em velocidade estabelece uma escolha clara para a implementação final de um classificador léxico no AMAZONAS.

Por fim, é possível notar a diferença entre todos os modelos desenvolvidos, por meio da Tabela 6.

De forma conjunta, os resultados indicam um comportamento complementar entre os classificadores: DGA e Typosquat apresentam maior capacidade de detecção em cenários especializados, enquanto DistilBERT atua como filtro léxico geral com melhor compromisso entre

desempenho e custo computacional. Esse arranjo sustenta a decisão de usar uma arquitetura modular, em que a combinação dos classificadores na Fase I aumenta a cobertura de detecção de domínios potencialmente maliciosos antes do encaminhamento para a Fase II.

Ressalta-se que os valores reportados para o classificador de DGA e para o classificador de Typosquat foram extraídos da literatura de referência de cada abordagem, respectivamente do trabalho de Rafael Gregório (Gregório, 2025) e da base/modelo disponibilizada pela Anvilogic (descrita na Seção 3.2). Já os resultados dos classificadores léxicos BERT e DistilBERT correspondem aos experimentos conduzidos no escopo desta dissertação.

Tabela 6 – Comparação das métricas de avaliação entre os classificadores

Métrica	**Classificador de DGA**	**Classificador de Typosquat**	**Classificador Léxico BERT**	**Classificador Léxico DistilBERT**
Acurácia	98%	–	50.63%	82.38%
F1-score	–	0.9113	0.3404	0.8105
Precisão	97.95%	91.97%	25.64%	86.51%
Recall	97.96%	90.31%	50.63%	76.23%

Fonte: Autora.

Quando comparado aos estudos relacionados, principalmente o Mahdaouy et al. (2024) podemos notar algumas similaridades, sendo que, em ambos os casos o modelo BERT se saiu pior que o modelo menor e mais especializado. Em um conjunto de dados próximo ao que foi utilizado nesta dissertação, os valores do classificador DomURL (Mahdaouy et al., 2024) ficaram com as seguintes métricas: 88.73% de acurácia, 0.8833 de F1-Score, 89.30% de precisão e 87.83% de *recall*. Os valores ficaram acima do modelo DistilBERT treinado, mostrando que ainda há abertura para melhoria no modelo proposto do AMAZONAS.

4.3 RESULTADOS DA FASE II: COLETORES

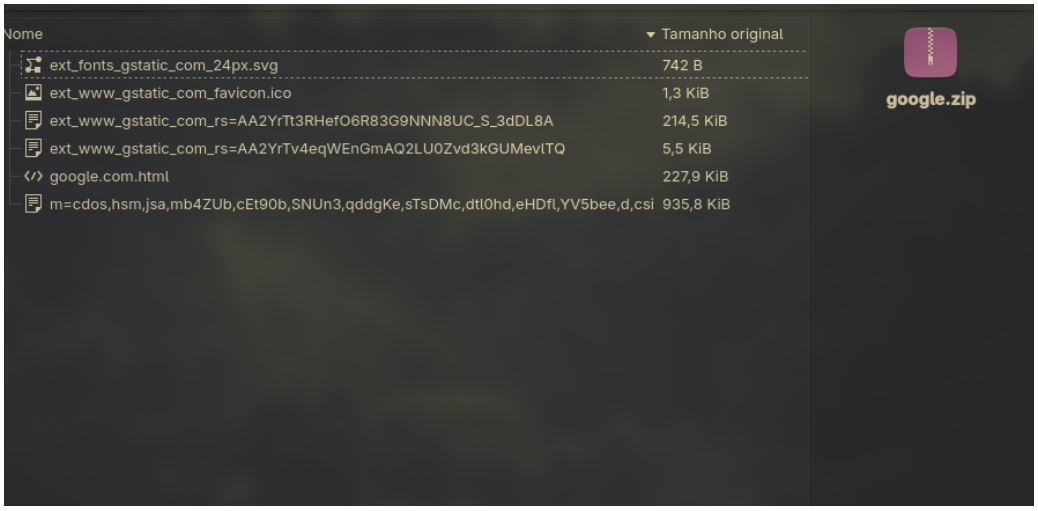
Nesta seção é apresentada a execução de cada um dos coletores e suas devidas respostas, que são armazenadas em banco de dados. Para fins demonstrativos, após a execução de cada coletor, seus resultados são exibidos na tela. Os testes desta etapa foram realizados em ambiente local/controlado, com execução manual ou em lote dos coletores, e não correspondem ainda a uma implantação contínua em produção. Portanto, os resultados validam o funcionamento técnico dos componentes e a estrutura de armazenamento das evidências, mas não medem disponibilidade, escalabilidade ou latência de um serviço operacional em tempo real.

4.3.1 Coletor Web

Ao utilizar o coletor para o site `google.com` de modo verboso, demonstrando cada passo realizado, a saída pode ser vista no Apêndice I na Listagem I.1. Além disso, é possível notar o tamanho do arquivo de aproximadamente 500 KB, consideravelmente pequeno quando é considerado que toda a página está sendo baixada juntamente com todos os seus *assets*.

Por fim, a Figura Figura 13 demonstra o conteúdo contido dentro do arquivo zip.

Figura 13 – Conteúdo do zip gerado pelo coletor web.



Fonte: Autora.

4.3.2 Coletor de DNS Ativo

Para demonstrar o coletor de DNS Ativo, diversos *logs* foram adicionados para demonstrar sua efetividade. Sua saída pode ser vista no Apêndice I na Listagem I.2. É possível notar que, em alguns casos, houve falhas, isso é esperado, pois nem todos os domínios vão conter todos os registros possíveis.

4.3.3 Coletor de DNS Passivo

Os dados de DNS passivo podem ser vistos no Apêndice I na Listagem I.3. Em geral, domínios maliciosos tendem a apresentar menos acessos do que domínios como `google.com` e, portanto, seus resultados serão menores. No exemplo apresentado, apenas 6 acessos foram encontrados.

4.3.4 Apoio à decisão com LLM

Como etapa complementar da Fase II, foi realizado um experimento de apoio à decisão analítica por meio de LLM, utilizando os *prompts* gerados automaticamente pelo script descrito anteriormente (Apêndice D), a partir das evidências coletadas e do resultado da classificação inicial. Os testes foram executados com a versão gratuita do ChatGPT, tendo como entrada os mesmos *prompts* de exemplo apresentados nos Apêndices E e F, com o objetivo de avaliar a utilidade prática de uma segunda opinião textual estruturada para o analista.

Nos dois cenários avaliados (domínio potencialmente malicioso e domínio benigno), a resposta da IA foi consistente com os indícios técnicos apresentados pelo módulo AMAZONAS, reforçando a recomendação de bloqueio no caso malicioso e de liberação no caso benigno. Como avaliação exploratória, a concordância observada foi de 2/2 casos em relação à recomendação esperada, considerando apenas os exemplos documentados nos apêndices. Esse número

não deve ser interpretado como métrica estatística robusta, pois o conjunto é pequeno e foi usado para validar o formato do *prompt* e a utilidade qualitativa da resposta. Esse uso não substitui a decisão humana final, mas contribui para reduzir o tempo de triagem e aumentar a padronização da justificativa técnica.

As respostas completas utilizadas nesta avaliação estão apresentadas no Apêndice G e no Apêndice H.

5 CONCLUSÃO

Este trabalho apresentou o desenvolvimento e a validação do módulo AMAZONAS para detecção, *scoring* e monitoramento de domínios potencialmente maliciosos. A proposta foi estruturada em duas fases complementares: (i) classificação inicial com modelos de IA especializados e (ii) monitoramento contínuo com coleta de evidências técnicas para apoio à decisão analítica.

Os resultados obtidos sustentam a viabilidade da arquitetura proposta. Na Fase I, os classificadores especializados mostraram desempenho consistente, com destaque para os modelos voltados a DGA e Typosquatting e para o classificador léxico DistilBERT, que apresentou melhor equilíbrio entre desempenho e custo computacional quando comparado ao BERT. Na Fase II, os coletores *web*, DNS ativo e DNS passivo demonstraram capacidade de produzir dados acionáveis para investigação, ampliando a interpretabilidade da decisão final.

Como contribuição adicional, o trabalho incorporou uma etapa de apoio ao analista por meio da geração automática de *prompts* para LLM, consolidando evidências operacionais e recomendação inicial em formato padronizado. Nos cenários avaliados, as respostas obtidas foram coerentes com os indícios técnicos observados, reforçando o potencial dessa estratégia como suporte à triagem, sem substituir a decisão humana.

Dessa forma, considera-se que os objetivos definidos para o módulo AMAZONAS foram atingidos: construção de um protótipo funcional, validação de classificadores em cenário aplicado e integração de mecanismos de monitoramento e apoio analítico para o contexto de segurança de domínios.

Em síntese, os resultados obtidos confirmam a efetividade da proposta: os classificadores da Fase I apresentaram desempenho consistente, com acurácia superior a 80% nos modelos validados neste trabalho, enquanto os coletores da Fase II demonstraram capacidade de produzir evidências técnicas acionáveis para suporte à decisão. No experimento operacional com cerca de 20 mil domínios, o módulo manteve comportamento compatível com uso contínuo, separando os casos em listas operacionais e encaminhando apenas uma fração reduzida para análise aprofundada. Esse conjunto de evidências sustenta a conclusão de que o AMAZONAS atende ao objetivo de apoiar, de forma prática e escalável, a identificação e o monitoramento de domínios potencialmente maliciosos.

5.1 TRABALHOS FUTUROS

Como continuidade desta pesquisa, destacam-se os seguintes direcionamentos:

- ampliação da avaliação com janelas temporais maiores e maior diversidade de domínios recém-registrados;

- evolução dos classificadores léxicos com técnicas adicionais de *fine-tuning* e calibração de decisão;
- expansão do pipeline de monitoramento para incorporar novas fontes de inteligência de ameaças;
- avaliação comparativa sistemática entre diferentes LLMs no apoio à decisão analítica;
- disponibilização controlada dos artefatos do projeto para reprodutibilidade e pesquisas futuras.

REFERÊNCIAS

ALHOGAIL, A.; AL-TURAIKI, I. Improved detection of malicious domain names using gradient boosted machines and feature engineering. **Information Technology and Control**, Kaunas University of Technology (KTU), v. 51, n. 2, p. 313331, jun. 2022. ISSN 1392-124X. Disponível em: http://dx.doi.org/10.5755/j01.itc.51.2.30380.

APACHE. **Apache Iceberg**. 2025. Disponível em: https://iceberg.apache.org/.

BATISTA, V. B. Framework para integração e automação de modelos de aprendizado de máquina para classificação de nomes de domínio. **Monografia (Bacharelado em Ciência da Computação)**, Universidade Estadual Paulista Júlio De Mesquita Filho, São José do Rio Preto, 2020.

BEHERA, R.; DAS, K. A survey on machine learning: Concept, algorithms and applications. **International Journal of Innovative Research in Computer and Communication Engineering**, v. 2, 02 2017.

BREIMAN, L. Random forests. **Machine Learning**, Springer Science and Business Media LLC, v. 45, n. 1, p. 532, 2001. ISSN 0885-6125. Disponível em: http://dx.doi.org/10.1023/A:1010933404324.

CLARK, J. H. et al. CANINE: pre-training an efficient tokenization-free encoder for language representation. **CoRR**, abs/2103.06874, 2021. Disponível em: https://arxiv.org/abs/2103.06874.

DESMET, L. et al. Premadoma: An operational solution to prevent malicious domain name registrations in the .eu tld. **Digital Threats**, Association for Computing Machinery, New York, NY, USA, v. 2, n. 1, jan. 2021. Disponível em: https://doi.org/10.1145/3419476.

DEVLIN, J. et al. BERT: pre-training of deep bidirectional transformers for language understanding. **CoRR**, abs/1810.04805, 2018. Disponível em: http://arxiv.org/abs/1810.04805.

DOUZE, M. et al. The faiss library. 2024.

DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. **Journal of Machine Learning Research**, v. 12, n. 61, p. 2121–2159, 2011. Disponível em: http://jmlr.org/papers/v12/duchi11a.html.

EDMONDS, R. Isc passive dns architecture. **Internet Systems Consortium, Inc**, v. 18, 2012.

GLOROT, X.; BORDES, A.; BENGIO, Y. Deep sparse rectifier neural networks. In: GORDON, G.; DUNSON, D.; DUDIK, M. (Ed.). **Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics**. [S.l.: s.n.], 2011. (JMLR Workshop and Conference Proceedings, v. 15), p. 315–323.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. http://www.deeplearningbook.org.

GREGóRIO, J. R. **Detecção de domínios gerados por algoritmos com aprendizado profundo incremental e DNS passivo**. 2025. Dissertação (Dissertação (Mestrado em Ciência da Computação)) — Universidade Estadual Paulista (Unesp), Instituto de Biociências, Letras e Ciências Exatas (Ibilce), São José do Rio Preto, 2025.

ICANN. **DNS Abuse Mitigation Program**. 2024. https://www.icann.org/dnsabuse. Acesso em 25 de julho de 2025.

JEREMIAH, D. et al. Niom-dga: Nature-inspired optimised ml-based model for dga detection. **Computers & Security**, v. 157, p. 104561, 2025. ISSN 0167-4048. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167404825002500.

Kaspersky. **What is Typosquatting?** 2025. https://www.kaspersky.com/resource-center/definitions/what-is-typosquatting. Acesso em 13 de julho de 2025.

KUROSE, J. F.; ROSS, K. W. **Computer Networking: A Top-down Approach**. [S.l.: s.n.], 2017.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, May 2015. Disponível em: https://ideas.repec.org/a/nat/nature/v521y2015i7553d10.1038_nature14539.html.

LECUN, Y. et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, 1998.

LUDERMIR, T. B. Inteligência artificial e aprendizado de máquina: estado atual e tendências. **Estudos Avançados**, Instituto de Estudos Avançados da Universidade de São Paulo, v. 35, n. 101, p. 8594, Jan 2021. ISSN 0103-4014. Disponível em: https://doi.org/10.1590/s0103-4014.2021.35101.007.

MAHDAOUY, A. et al. Domurls_bert: Pre-trained bert-based model for malicious domains and urls detection and classification. 09 2024.

ManageEngine. **What is Typosquatting?** 2025. https://www.manageengine.com/log-management/cyber-security/what-is-typosquatting.html. Acesso em 13 de julho de 2025.

MIT Technology Review Brasil. Data literacy: a importância da alfabetização em dados em um mundo big data. **MIT Technology Review Brasil**, oct 2022. Disponível em: https://mittechreview.com.br/data-literacy-a-importancia-da-alfabetizacao-em-dados-em-um-mundo-big-data/.

MITCHELL, T. M. **Machine Learning**. New York: McGraw-Hill, 1997.

MOCKAPETRIS, P. **Domain names - implementation and specification**. RFC Editor, 1987. RFC 1035. (Request for Comments, 1035). Disponível em: https://www.rfc-editor.org/info/rfc1035.

PAIXÃO, G. M. d. M. et al. Machine learning na medicina: Revisão e aplicabilidade. **Arquivos Brasileiros de Cardiologia**, Sociedade Brasileira de Cardiologia - SBC, v. 118, n. 1, p. 95102, Jan 2022. ISSN 0066-782X. Disponível em: https://doi.org/10.36660/abc.20200596.

PLOHMANN, D. et al. A comprehensive measurement study of domain generating malware. In: . [S.l.: s.n.], 2016.

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

QIAN, N. On the momentum term in gradient descent learning algorithms. **Neural Networks**, v. 12, n. 1, p. 145–151, 1999. ISSN 0893-6080. Disponível em: https://www.sciencedirect.com/science/article/pii/S0893608098001166.

ROMANHOLO, G.; CANSIAN, A. M. Automatização e aprimoramento de classificadores para a detecção de domínios maliciosos. In: **Anais do XXXVI Congresso de Iniciação Científica da Unesp: "Ciência em tempos de crise climática e social"**. Águas de Lindóia, SP: Hotel Bendito Cacao Family Resort, 2024. Acesso em: 25/09/2025. Disponível em: https://www.even3.com.br/anais/xxxvicicunesp/1010160-AUTOMATIZACAO-E-APRIMORAMENTO-DE-CLASSIFICADORES-\PARA-A-DETECCAO-DE-DOMINIOS-MALICIOSOS.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, Springer Science and Business Media LLC, v. 323, n. 6088, p. 533–536, 1986.

SANH, V. et al. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. **ArXiv**, abs/1910.01108, 2019.

SILVA, L. M. d. Early identification of abused domains in tld through passive dns applying machine learning techniques. **International Journal of Communication Networks and Information Security**, v. 14, p. 7685,, 2022. N.

SUBRAMANIAM, D.; JEYANANTHAN, P.; SATHIPARAN, N. Soft computing techniques to predict the electrical resistivity of pervious concrete. **Asian Journal of Civil Engineering**, v. 25, p. 1–12, 07 2023.

UEDA, Y.; KONAKA, C. J. **Serial Experiments Lain: Layer 06 – Kids**. 1998. Television episode. Directed by Ryutaro Nakamura; quoted line spoken by Miho Iwakura.

University of Texas at San Antonio, TechSolutions. **Typosquatting**. 2024. https://www.utsa.edu/techsolutions/Cyber-Awareness/Typosquatting.html. Acesso em 13 de julho de 2025.

VASWANI, A. et al. Attention is all you need. In: **Proceedings of the 31st International Conference on Neural Information Processing Systems**. Red Hook, NY, USA: Curran Associates Inc., 2017. (NIPS'17), p. 60006010. ISBN 9781510860964.

WELCH, J. **Training Large Language Models for Advanced Typosquatting Detection**. 2025. Disponível em: https://arxiv.org/abs/2503.22406.

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.

A DNSCHECK

As Figuras Figura 14, Figura 15 e Figura 16 apresentam, respectivamente, os módulos Nevermore, Classifiers e Dashboards do DNSCheck.

Figura 14 – Visão do módulo Nevermore com a consulta contendo "google.com".

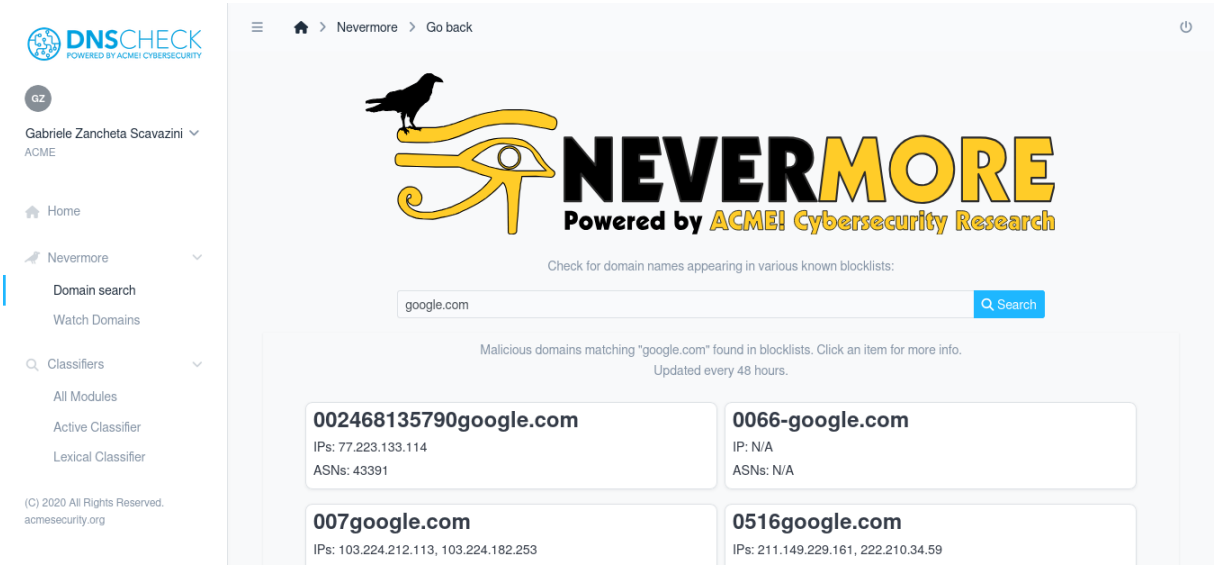


Figura 15 – Visão do módulo de classificadores, contendo os classificadores Léxico e Ativo.

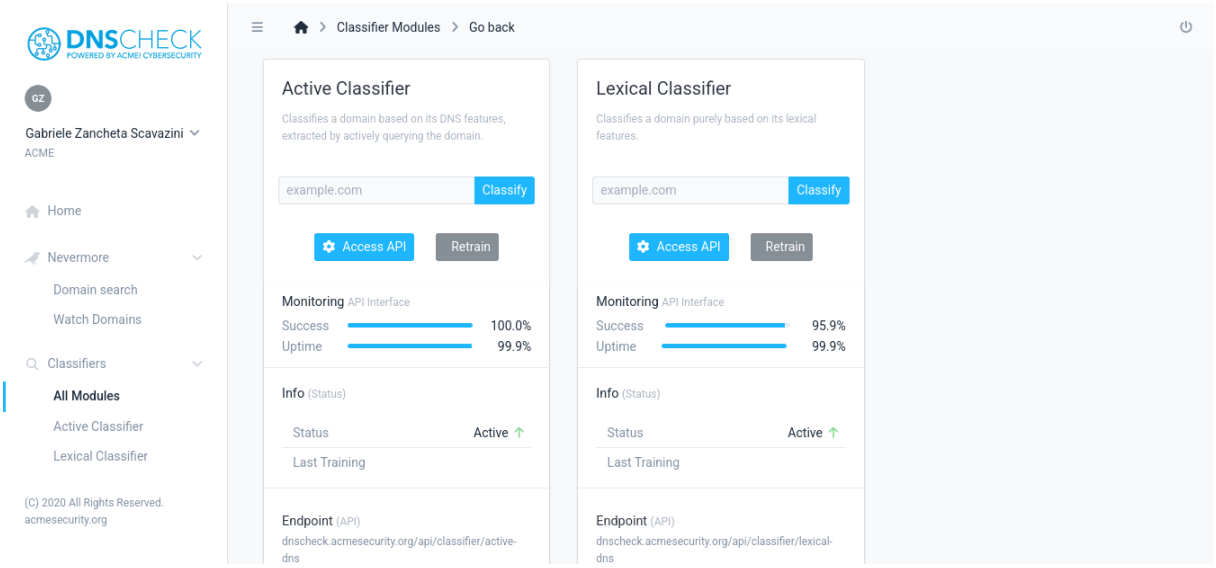
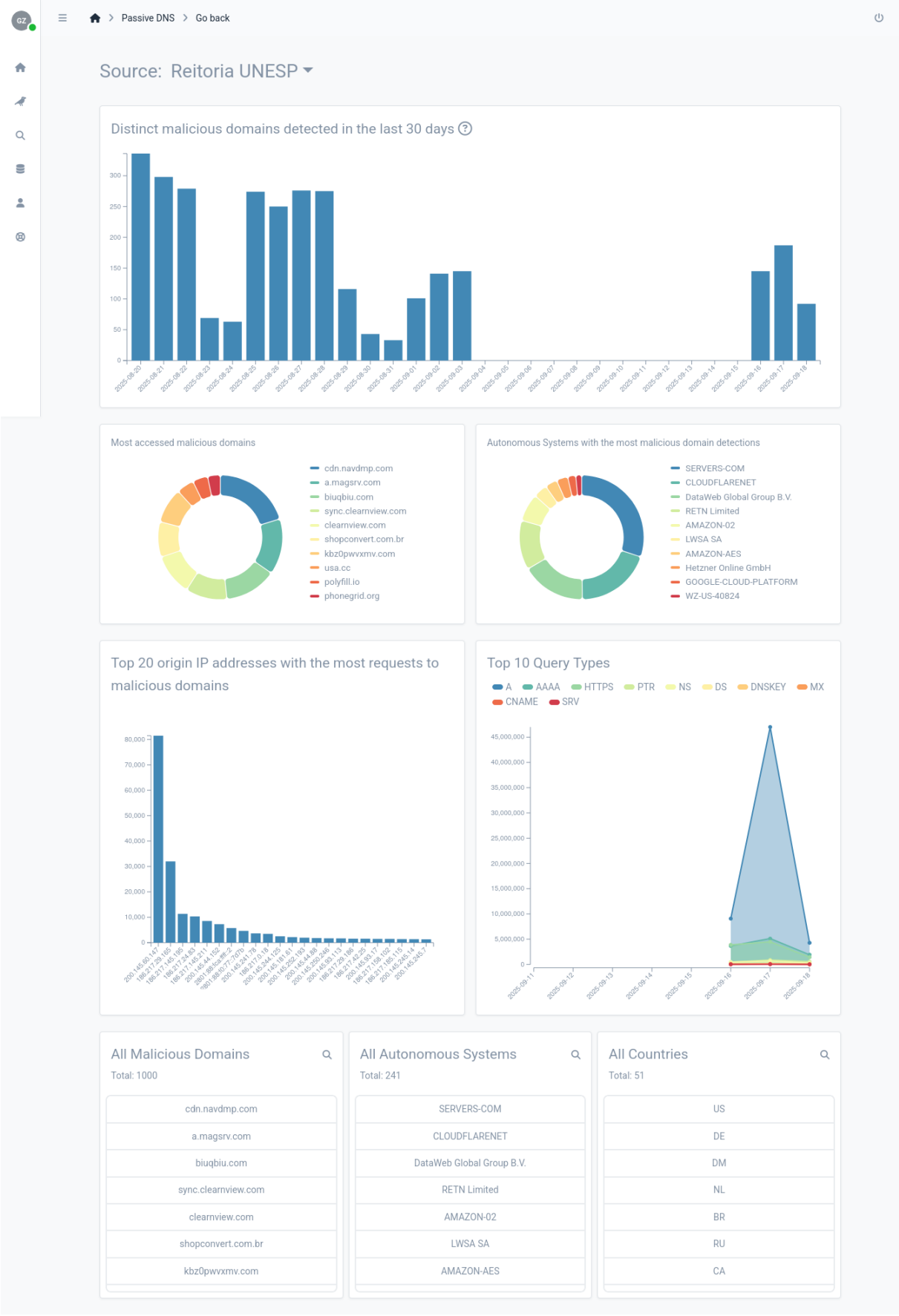


Figura 16 – Visão do módulo de *dashboard* exibindo algumas das informações acionáveis disponíveis.



B HIPERPARÂMETROS DE TREINAMENTO

Tabela 7 – Hiperparâmetros Utilizados no Treinamento do Modelo BERT

Hiperparâmetro	Valor	Descrição
`num_train_epochs`	3	Número total de épocas de treinamento.
`per_device_train_batch_size`	8	Tamanho do batch por dispositivo durante o treinamento.
`per_device_eval_batch_size`	16	Tamanho do batch por dispositivo durante a avaliação.
`warmup_steps`	500	Número de passos de *warm-up* no início do treinamento.
`weight_decay`	0.01	Coeficiente de regularização L2 (decaimento de peso).
`logging_steps`	10	Intervalo (em steps) para salvar informações de log.
`do_eval`	True	Indica se a avaliação deve ser realizada durante o treinamento (substitui `evaluation_strategy`).
`save_steps`	500	Número de passos entre salvamentos do modelo (substitui `save_strategy`).
`logging_first_step`	True	Realiza log já no primeiro passo de treinamento.

Fonte: Autora.

Tabela 8 – Hiperparâmetros Utilizados no Treinamento do Modelo DistilBERT

Hiperparâmetro	**Valor**	**Descrição**
num_train_epochs	10	Número total de épocas de treinamento.
per_device_train_batch_size	64	Tamanho do batch por dispositivo durante o treinamento.
per_device_eval_batch_size	64	Tamanho do batch por dispositivo durante a avaliação.
eval_strategy	epoch	Estratégia de avaliação (neste caso, ao final de cada época).
save_strategy	epoch	Estratégia de salvamento do modelo (ao final de cada época).
logging_steps	10	Intervalo (em steps) para salvar informações de log.
load_best_model_at_end	True	Indica se o melhor modelo (com base na métrica) deve ser carregado ao final do treinamento.
metric_for_best_model	f1	Métrica usada para identificar o melhor modelo.
greater_is_better	True	Define se valores maiores da métrica indicam melhor desempenho.
save_total_limit	1	Número máximo de checkpoints a serem salvos.
seed	42	Semente aleatória para reprodutibilidade.

Fonte: Autora.

Tabela 9 – Hiperparâmetros Utilizados no Treinamento do Modelo Canine-c

Hiperparâmetro	**Valor**	**Descrição**
`num_train_epochs`	`10`	Número máximo de épocas de treinamento (pode ser interrompido antes com *early stopping*).
`per_device_train_batch_size`	`8`	Tamanho do batch por dispositivo durante o treinamento.
`per_device_eval_batch_size`	`16`	Tamanho do batch por dispositivo durante a avaliação.
`warmup_steps`	`500`	Número de passos de *warm-up* no início do treinamento para estabilizar o otimizador.
`weight_decay`	`0.01`	Coeficiente de decaimento de peso (regularização L2).
`learning_rate`	`2e-5`	Taxa de aprendizado usada pelo otimizador.
`eval_strategy`	`epoch`	Estratégia de avaliação (neste caso, ao final de cada época).
`save_strategy`	`epoch`	Estratégia de salvamento do modelo (ao final de cada época).
`logging_steps`	`10`	Intervalo (em steps) para salvar informações de log.
`load_best_model_at_end`	`True`	Indica se o melhor modelo (com base na métrica) deve ser carregado ao final do treinamento.
`metric_for_best_model`	`eval_loss`	Métrica usada para identificar o melhor modelo (neste caso, a perda na avaliação).
`greater_is_better`	`False`	Indica que valores menores da métrica representam melhor desempenho.
`save_total_limit`	`2`	Número máximo de checkpoints a serem salvos.

Fonte: Autora.

C CÓDIGO DOS CLASSIFICADORES

```python
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
from typing import List
from sentence_transformers import SentenceTransformer
import faiss
import numpy as np
import os

# --- 1. Configuration ---

MODEL_PATH = "fresh_model_download"

DOMAIN_LIST_PATH = 'domains.txt'

SIMILARITY_THRESHOLD = 0.8

# --- 2. Model and FAISS Index Loading ---

if not os.path.isdir(MODEL_PATH):
    raise OSError(f"Model directory not found at: {MODEL_PATH}")
if not os.path.isfile(DOMAIN_LIST_PATH):
    raise FileNotFoundError(f"Domain list not found: {DOMAIN_LIST_PATH}")

print("Loading sentence transformer model from local path...")
model = SentenceTransformer(MODEL_PATH)
print("Model loaded successfully.")

print(f"Loading known domains from {DOMAIN_LIST_PATH}...")
with open(DOMAIN_LIST_PATH, 'r') as f:
    top_domains = [line.strip() for line in f if line.strip()]
print(f"Loaded {len(top_domains)} domains.")

print("Generating embeddings for known domains...")
# Generate and normalize embeddings for the known domains.
top_embeddings = model.encode(top_domains, normalize_embeddings=True,
    show_progress_bar=True)
top_embeddings = np.array(top_embeddings).astype('float32')
print("Embeddings generated.")

print("Creating FAISS index...")
# Create a FAISS index for efficient cosine similarity search.
index = faiss.IndexFlatIP(top_embeddings.shape[1])
index.add(top_embeddings)
```

```
43 print("FAISS index created and populated.")
44
45 # --- 3. FastAPI Application ---
46
47 app = FastAPI(
48     title="Typosquatting Detection API",
49     description="Uses a local sentence-transformer model and FAISS to
    detect potential typosquatting domains.",
50     version="1.0"
51 )
52
53 # Pydantic model for the API request body.
54 class DomainListRequest(BaseModel):
55     domains: List[str]
56
57 # --- 4. Core Detection Logic ---
58
59 def find_most_similar(domain_to_check: str):
60     """Encodes a domain and searches the FAISS index for the most similar
    known domain."""
61     domain_embedding = model.encode([domain_to_check], normalize_embeddings
    =True)
62     domain_embedding = np.array(domain_embedding).astype('float32')
63
64     # Search the index for the top 1 most similar vector.
65     # D: Distances (similarity scores), I: Indices of the matched vectors.
66     distances, indices = index.search(domain_embedding, 1)
67
68     most_similar_domain = top_domains[indices[0][0]]
69     similarity_score = float(distances[0][0])
70
71     return most_similar_domain, similarity_score
72
73 # --- 5. API Endpoint ---
74
75 @app.post("/check-typosquatting/")
76 async def check_typosquatting_api(request: DomainListRequest) -> dict:
77     """
78     Checks a list of domains for potential typosquatting against a known
    list.
79
80     :param request: A list of domains to check.
81     :return: A list of results with similarity scores.
82     """
83     if not request.domains:
84         raise HTTPException(status_code=400, detail="Input 'domains' list
    cannot be empty.")
```

```
85
86      results = []
87      for domain in request.domains:
88          most_similar, score = find_most_similar(domain)
89
90          if score > SIMILARITY_THRESHOLD and domain != most_similar:
91              results.append({
92                  "domain_checked": domain,
93                  "is_potential_typosquat": True,
94                  "message": f"'{domain}' may be a typosquat of '{
    most_similar}'",
95                  "similarity_score": score,
96                  "closest_match": most_similar
97              })
98          else:
99              results.append({
100                 "domain_checked": domain,
101                 "is_potential_typosquat": False,
102                 "message": f"'{domain}' appears to be safe.",
103                 "similarity_score": score,
104                 "closest_match": most_similar
105             })
106
107     return {"results": results}
108
109 @app.get("/", include_in_schema=False)
110 async def root():
111     return {"message": "Typosquatting Detection API is running. See /docs
    for usage."}
```

Listing C.1 – Código de API e execução do modelo de detecção de typosquatting.

```
1
2  from contextlib import asynccontextmanager
3  from transformers import DistilBertTokenizer,
       DistilBertForSequenceClassification
4  import torch
5  from pydantic import BaseModel
6  from fastapi import FastAPI, Depends
7
8  MODEL_PATH = './distilbert-base-uncased'
9
10 tokenizer = DistilBertTokenizer.from_pretrained(MODEL_PATH)
11 model = DistilBertForSequenceClassification.from_pretrained(MODEL_PATH)
12 model.eval()
13
14 app = FastAPI(
15     title="DistilBERT for malicious domain detection",
```

```
16      description="An API to perform malicious domain detection using a
        DistilBERT model as base.",
17      version="1.0.0",
18  )
19
20  class DomainRequest(BaseModel):
21      domain: str
22
23  class DomainResponse(BaseModel):
24      result: str
25      confidence: float
26
27  @app.post("/predict/", response_model=DomainResponse)
28  async def predict(
29          request: DomainRequest,
30  ):
31      """
32      Performs sentiment analysis on the input text.
33
34      - **text**: The text to be analyzed.
35      """
36
37      inputs = tokenizer(request.domain, return_tensors="pt", truncation=True
        , padding=True)
38
39      with torch.no_grad():
40          outputs = model(**inputs)
41
42      logits = outputs.logits
43      probabilities = torch.softmax(logits, dim=-1)
44      predicted_class_id = torch.argmax(probabilities, dim=-1).item()
45      confidence = probabilities[0][predicted_class_id].item()
46      result = model.config.id2label[predicted_class_id]
47
48      return DomainResponse(result=result, confidence=confidence)
```

Listing C.2 – Código do classificador DistilBert integrado em API.

D SCRIPT DE GERAÇÃO AUTOMÁTICA DE PROMPT PARA LLM

O script da Listagem D.1 implementa a geração automática de *prompts* em formato Markdown para apoiar a análise humana na Fase II.

```
"""LLM Prompt Generator for Domain Analysis
Generates comprehensive prompts for external LLM validation
"""
import json
from typing import Dict, Optional
from datetime import datetime

class DomainAnalysisPromptGenerator:
    """Generate structured prompts for LLM analysis of domain classification"""

    @staticmethod
    def generate_prompt(
        domain_name: str,
        classification_data: Dict,
        phase_two_data: Dict,
        ml_recommendation: Dict,
        decision_history: list = None
    ) -> str:
        """
        Generate a comprehensive prompt for LLM analysis

        Args:
            domain_name: The domain being analyzed
            classification_data: Phase I classification results
            phase_two_data: Phase II monitoring data (DNS, HTML, pDNS)
            ml_recommendation: ML recommendation with features
            decision_history: Previous analyst decisions

        Returns:
            Formatted prompt string ready for LLM input
        """

        prompt_sections = []

        prompt_sections.append(DomainAnalysisPromptGenerator._generate_header(domain_name))
        prompt_sections.append(DomainAnalysisPromptGenerator._generate_role_and_task())
        prompt_sections.append(DomainAnalysisPromptGenerator._generate_classification_section(classification_data))
```

```
38          prompt_sections.append(DomainAnalysisPromptGenerator.
    _generate_ml_section(ml_recommendation, phase_two_data))
39
40          if phase_two_data:
41              prompt_sections.append(DomainAnalysisPromptGenerator.
    _generate_phase_two_section(phase_two_data))
42
43          if decision_history:
44              prompt_sections.append(DomainAnalysisPromptGenerator.
    _generate_history_section(decision_history))
45
46          prompt_sections.append(DomainAnalysisPromptGenerator.
    _generate_analysis_request())
47
48          return "\n\n".join(prompt_sections)
49
50      @staticmethod
51      def _generate_header(domain_name: str) -> str:
52          return f"""# AMAZONAS Domain Security Analysis Request
53
54 **Domain Under Analysis:** `{domain_name}`
55 **Analysis Date:** {datetime.now().strftime('%Y-%m-%d %H:%M:%S UTC')}
56 **System:** AMAZONAS (Automated Malicious Domain Analysis System)"""
57
58      @staticmethod
59      def _generate_role_and_task() -> str:
60          return """## Your Role
61
62 You are a cybersecurity expert specializing in malicious domain detection
    and threat intelligence. Your task is to:
63
64 1. **Review** all provided evidence about this domain
65 2. **Analyze** the classification scores, DNS data, web content, and
    traffic patterns
66 3. **Validate** or challenge the ML recommendation
67 4. **Provide** a final recommendation: ALLOW, BLOCK, or NEEDS MORE
    INVESTIGATION
68 5. **Explain** your reasoning with specific evidence references
69
70 Consider common threat indicators:
71 - Domain Generation Algorithm (DGA) patterns
72 - Typosquatting of legitimate brands
73 - Suspicious lexical patterns
74 - DNS misconfigurations or anomalies
75 - Minimal or suspicious web content
76 - Unusual traffic patterns
77 - Geographic anomalies in traffic sources"""
```

```
78
79      @staticmethod
80      def _generate_classification_section(classification_data: Dict) -> str:
81          score = classification_data.get('score', 0)
82          classification = classification_data.get('classification', 'unknown
        ')
83          results = classification_data.get('results', {})
84
85          classifier_details = []
86          if isinstance(results, dict):
87              for classifier_name, result in results.items():
88                  if isinstance(result, dict) and 'results' in result:
89                      for res in result.get('results', []):
90                          classifier_details.append(
91                              f"  - **{classifier_name}**: Score {res.get('
        score', 'N/A'):.3f} "
92                              f"({'MALICIOUS' if res.get('is_malicious') else
         'BENIGN'})"
93                          )
94
95          classifier_text = "\n".join(classifier_details) if
        classifier_details else "  - No detailed classifier results available"
96
97          return f"""## Phase I Classification Results
98
99 **Overall Risk Score:** {score:.3f} / 1.0
100 **Classification Category:** {classification.upper().replace('_', ' ')}
101
102 ### Individual Classifier Results:
103 {classifier_text}
104
105 **Interpretation:**
106 - Score < 0.5: Likely benign
107 - Score 0.5-0.75: Gray area (requires Phase II analysis)
108 - Score > 0.75: Likely malicious"""
109
110     @staticmethod
111     def _generate_ml_section(ml_recommendation: Dict, phase_two_data: Dict)
        -> str:
112         recommendation = ml_recommendation.get('recommendation', 'unknown')
113         confidence = ml_recommendation.get('confidence', 0)
114         reasons = ml_recommendation.get('reasons', [])
115         features = ml_recommendation.get('features', {})
116
117         reasons_text = "\n".join([f"{i+1}. {reason}" for i, reason in
        enumerate(reasons)])
118
```

```
119
120         key_features = []
121         if features:
122             key_features.append(f"- Base Classification Score: {features.
    get('base_score', 0):.3f}")
123             key_features.append(f"- Has Valid DNS Response: {'Yes' if
    features.get('has_dns') else 'No'}")
124             key_features.append(f"- DNS Response Code: {features.get('
    dns_response_code', 'N/A')}")
125
126
127             html_actual = phase_two_data.get('html') if phase_two_data else
     None
128             if html_actual:
129                 if html_actual.get('no_html'):
130                     key_features.append(f"- Has HTML Content: No (site
    unreachable)")
131                     key_features.append(f"- HTML Size: 0 bytes")
132                 else:
133                     file_size = html_actual.get('file_size', 0)
134                     key_features.append(f"- Has HTML Content: Yes")
135                     key_features.append(f"- HTML Size: {file_size:,} bytes
    ({file_size/1024:.1f} KB)")
136             else:
137                 key_features.append(f"- Has HTML Content: Not collected yet
    ")
138                 key_features.append(f"- HTML Size: N/A")
139
140
141             pdns_actual = phase_two_data.get('passive_dns') if
    phase_two_data else None
142             if pdns_actual:
143                 actual_count = pdns_actual.get('query_count', 0)
144                 key_features.append(f"- Passive DNS Query Count: {
    actual_count:,}")
145                 if actual_count == 0:
146                     key_features.append("- [ALERTA] No passive DNS traffic
    recorded (may indicate new/unused domain)")
147                 elif actual_count > 1000:
148                     key_features.append("- [OK] High traffic volume
    suggests legitimate, active domain")
149             else:
150                 key_features.append(f"- Passive DNS Query Count: Not
    collected yet")
151
152             key_features.append(f"- Unique Querier IPs: {features.get('
    pdns_unique_ips', 0)}")
```

```
153
154             # Risk indicators (only show if actually true based on real
    data)
155             if features.get('dns_nxdomain'):
156                 key_features.append("- [ALERTA] **ALERT:** Domain returns
    NXDOMAIN (doesn't resolve)")
157             if html_actual and html_actual.get('no_html'):
158                 key_features.append("- [ALERTA] **ALERT:** No web content
    found (site unreachable)")
159             if html_actual and not html_actual.get('no_html'):
160                 size = html_actual.get('file_size', 0)
161                 if 0 < size < 1000:
162                     key_features.append("- [ALERTA] **ALERT:** Suspiciously
     small HTML content (<1KB)")
163
164         features_text = "\n".join(key_features) if key_features else "- No
    feature data available"
165
166         return f"""## ML-Based Recommendation
167
168 **Recommendation:** {recommendation.upper().replace('_', ' ')}
169 **Confidence Level:** {confidence * 100:.1f}%
170
171 ### Reasoning:
172 {reasons_text}
173
174 ### Key Features Analyzed:
175 {features_text}"""
176
177     @staticmethod
178     def _generate_phase_two_section(phase_two_data: Dict) -> str:
179         sections = ["## Phase II Monitoring Evidence"]
180
181         dns_data = phase_two_data.get('dns')
182         if dns_data:
183             response_code = dns_data.get('response_code', 'Unknown')
184             response_meaning = {
185                 0: "NOERROR (successful)",
186                 1: "FORMERR (format error)",
187                 2: "SERVFAIL (server failure)",
188                 3: "NXDOMAIN (domain doesn't exist)",
189                 4: "NOTIMP (not implemented)",
190                 5: "REFUSED (query refused)"
191             }.get(response_code, "Unknown")
192
193             sections.append(f"""### Active DNS Analysis
194 - **Query Type:** {dns_data.get('query_type', 'Unknown')}
```

```
- **Response Code:** {response_code} - {response_meaning}
- **TTL (Time To Live):** {dns_data.get('ttl', 0)} seconds
- **Nameserver:** {dns_data.get('nameserver', 'Unknown')}
- **Last Updated:** {dns_data.get('updated_at', 'Unknown')}

**DNS Response Details:**
```json
{json.dumps(dns_data.get('response', {}), indent=2)}
```""")

        html_data = phase_two_data.get('html')
        if html_data:
            if html_data.get('no_html'):
                sections.append("""### Web Content Analysis
- **Status:** [ALERTA] No HTML content retrieved
- **Possible Reasons:** Site unreachable, no web server, blocking requests""")
            else:
                sections.append(f"""### Web Content Analysis
- **Archive Status:** [OK] Available for download
- **File Size:** {html_data.get('file_size', 0):,} bytes ({html_data.get('file_size', 0) / 1024:.2f} KB)
- **Archive Hash (SHA256):** `{html_data.get('hash_zip', 'N/A')}`
- **Created:** {html_data.get('created_at', 'Unknown')}

Analysis Notes:
- Very small HTML (<5KB): Often indicates parking pages or minimal content
- Large HTML (>500KB): Typically legitimate sites with full functionality
- Medium size (5-500KB): Requires content inspection""")

        pdns_data = phase_two_data.get('passive_dns')
        if pdns_data:

            geo = pdns_data.get('geolocation')
            if isinstance(geo, dict):
                country = geo.get('country', 'Unknown')
                asn = geo.get('asn', 'Unknown')
                asn_org = geo.get('asn_org', 'Unknown')
            else:
                # geolocation is just a country code string like 'US', 'BR', etc
                country = geo if geo else 'Unknown'
                asn = 'Unknown'
                asn_org = 'Unknown'

            sections.append(f"""### Passive DNS Analysis (ENTRADA2)
- **Total Queries:** {pdns_data.get('query_count', 0):,}
```

```
- **Querier IP:** {pdns_data.get('querier_ip', 'Unknown')}
- **Geographic Location:** {country}
- **ASN:** {asn} ({asn_org})
- **First Seen:** {pdns_data.get('first_seen', 'Unknown')}
- **Last Seen:** {pdns_data.get('last_seen', 'Unknown')}
- **Query Types:** {', '.join(map(str, pdns_data.get('query_types', [])))}
- **Response Codes:** {', '.join(map(str, pdns_data.get('response_codes', [])))}
- **DNSSEC Validated:** {'Yes' if pdns_data.get('dnssec_validated') else 'No'}

**Traffic Pattern Analysis:**
- High query count (>100): Indicates legitimate use or active botnet
- Geographic diversity: Multiple countries suggest legitimate service
- Single IP with low queries: Possible targeted attack or testing
- DNSSEC validation: Adds legitimacy to domain configuration""")

        risk_score = phase_two_data.get('risk_score')
        if risk_score is not None:
            sections.append(f"""### Combined Risk Assessment
**Phase II Risk Score:** {risk_score:.3f} / 1.0""")

        analyst_notes = phase_two_data.get('analyst_notes')
        if analyst_notes:
            sections.append(f"""### Previous Analyst Notes
{analyst_notes}""")

        return "\n\n".join(sections)

    @staticmethod
    def _generate_history_section(decision_history: list) -> str:
        if not decision_history:
            return ""

        history_items = []
        for decision in decision_history[:5]:
            history_items.append(
                f"- **{decision.get('created_at', 'Unknown')}**: "
                f"{decision.get('previous_status', 'unknown')} → {decision.get('new_status', 'unknown')} "
                f"by {decision.get('analyst_name', 'Unknown')}\n"
                f"  Reason: {decision.get('reason', 'No reason provided')}"
            )

        history_text = "\n".join(history_items)

        return f"""## Decision History
```

```
283
284 {history_text}"""
285
286     @staticmethod
287     def _generate_analysis_request() -> str:
288         return """## Your Analysis Task
289
290 Please provide a comprehensive security analysis addressing:
291
292 ### 1. Evidence Assessment
293 - Which pieces of evidence are most significant?
294 - Are there any red flags or concerning patterns?
295 - What evidence suggests benign vs. malicious intent?
296
297 ### 2. ML Recommendation Validation
298 - Do you agree with the ML recommendation?
299 - What factors support or contradict the recommendation?
300 - Is the confidence level appropriate given the evidence?
301
302 ### 3. Threat Classification
303 Based on all evidence, classify this domain as:
304 - **ALLOW (Benign)**: Safe to whitelist
305 - **BLOCK (Malicious)**: Should be blocked immediately
306 - **INVESTIGATE**: Needs more evidence or monitoring
307
308 ### 4. Specific Recommendations
309 - Should this domain be allowed, blocked, or monitored further?
310 - What additional evidence would be helpful?
311 - Are there any immediate actions security teams should take?
312
313 ### 5. Risk Summary
314 Provide a 2-3 sentence summary explaining your final determination and key
    reasoning.
315
316 ---
317
318 **Format your response with clear sections and bullet points for easy
    review by security analysts.**"""
319
320     @staticmethod
321     def generate_markdown_report(prompt: str, llm_response: str = None) ->
    str:
322         """Generate a markdown report combining prompt and LLM response"""
323
324         report = [prompt]
325
326         if llm_response:
```

```
327             report.append("\n\n" + "="*80)
328             report.append("\n# LLM ANALYSIS RESPONSE\n")
329             report.append(llm_response)
330             report.append("\n" + "="*80)
331 
332         return "\n".join(report)
```

Listing D.1 – Script para geração automática de *prompts* para validação em LLM

E EXEMPLO DE *PROMPT* PARA DOMÍNIO POTENCIALMENTE MALICIOSO

A Listagem E.1 apresenta um exemplo de *prompt* gerado automaticamente para um domínio com alta pontuação de risco.

```
# AMAZONAS Domain Security Analysis Request

**Domain Under Analysis:** `paypa1-secure-login.com`
**Analysis Date:** 2026-03-25 21:47:11 UTC
**System:** AMAZONAS (Automated Malicious Domain Analysis System)

## Your Role

You are a cybersecurity expert specializing in malicious domain detection and threat intelligence. Your task is to:

1. **Review** all provided evidence about this domain
2. **Analyze** the classification scores, DNS data, web content, and traffic patterns
3. **Validate** or challenge the ML recommendation
4. **Provide** a final recommendation: ALLOW, BLOCK, or NEEDS MORE INVESTIGATION
5. **Explain** your reasoning with specific evidence references

Consider common threat indicators:
- Domain Generation Algorithm (DGA) patterns
- Typosquatting of legitimate brands
- Suspicious lexical patterns
- DNS misconfigurations or anomalies
- Minimal or suspicious web content
- Unusual traffic patterns
- Geographic anomalies in traffic sources

## Phase I Classification Results

**Overall Risk Score:** 0.941 / 1.0
**Classification Category:** MALICIOUS

### Individual Classifier Results:
  - **DGA_Classifier**: Score 0.412 (BENIGN)
  - **Typosquatting_Classifier**: Score 0.941 (MALICIOUS)
  - **Lexical_Classifier**: Score 0.883 (MALICIOUS)

**Interpretation:**
- Score < 0.5: Likely benign
- Score 0.5-0.75: Gray area (requires Phase II analysis)
```

```
- Score > 0.75: Likely malicious

## ML-Based Recommendation

**Recommendation:** BLOCK LIST
**Confidence Level:** 95.0%

### Reasoning:
1. Very high classification score (0.94)
2. Suspiciously small HTML content

### Key Features Analyzed:
- Base Classification Score: 0.941
- Has Valid DNS Response: Yes
- DNS Response Code: 0
- Has HTML Content: Yes
- HTML Size: 487 bytes (0.5 KB)
- [ALERTA] ALERT: Suspiciously small HTML content (<1KB)
- Passive DNS Query Count: 43
- Unique Querier IPs: 3

## Phase II Monitoring Evidence

### Active DNS Analysis
- **Query Type:** A
- **Response Code:** 0 - NOERROR (successful)
- **TTL (Time To Live):** 300 seconds
- **Nameserver:** ns1.hostingprovider-anon.ru
- **Last Updated:** 2026-03-23 03:17:42

**DNS Response Details:**
```json
{
  "A": ["185.220.101.47"],
  "NS": ["ns1.hostingprovider-anon.ru", "ns2.hostingprovider-anon.ru"],
  "MX": [],
  "TXT": []
}
```

Web Content Analysis
- **Archive Status:** [OK] Available for download
- **File Size:** 487 bytes (0.48 KB)
- **Archive Hash (SHA256):** `
  f9e8d7c6b5a43210fedcba9876543210fedcba9876543210fedcba9876543210`
- **Created:** 2026-03-23 04:02:18

```

```
Analysis Notes:
- Very small HTML (<5KB): Often indicates parking pages or minimal content
- Large HTML (>500KB): Typically legitimate sites with full functionality
- Medium size (5-500KB): Requires content inspection

Passive DNS Analysis (ENTRADA2)
- **Total Queries:** 43
- **Querier IP:** 177.92.4.18
- **Geographic Location:** BR
- **ASN:** AS28573 (Claro S.A.)
- **First Seen:** 2026-03-23 04:45:30
- **Last Seen:** 2026-03-25 19:11:02
- **Query Types:** A
- **Response Codes:** NOERROR
- **DNSSEC Validated:** No

Traffic Pattern Analysis:
- High query count (>100): Indicates legitimate use or active botnet
- Geographic diversity: Multiple countries suggest legitimate service
- Single IP with low queries: Possible targeted attack or testing
- DNSSEC validation: Adds legitimacy to domain configuration

Combined Risk Assessment
Phase II Risk Score: 0.937 / 1.0

Your Analysis Task

Please provide a comprehensive security analysis addressing:

1. Evidence Assessment
- Which pieces of evidence are most significant?
- Are there any red flags or concerning patterns?
- What evidence suggests benign vs. malicious intent?

2. ML Recommendation Validation
- Do you agree with the ML recommendation?
- What factors support or contradict the recommendation?
- Is the confidence level appropriate given the evidence?

3. Threat Classification
Based on all evidence, classify this domain as:
- **ALLOW (Benign)**: Safe to whitelist
- **BLOCK (Malicious)**: Should be blocked immediately
- **INVESTIGATE**: Needs more evidence or monitoring

4. Specific Recommendations
- Should this domain be allowed, blocked, or monitored further?
```

```
- What additional evidence would be helpful?
- Are there any immediate actions security teams should take?

### 5. Risk Summary
Provide a 2-3 sentence summary explaining your final determination and key reasoning.

---

**Format your response with clear sections and bullet points for easy review by security analysts.**
```

Listing E.1 – *Prompt* gerado para domínio potencialmente malicioso

F EXEMPLO DE *PROMPT* PARA DOMÍNIO SEGURO

A Listagem F.1 apresenta um exemplo de *prompt* gerado automaticamente para um domínio classificado como seguro.

```
# AMAZONAS Domain Security Analysis Request

**Domain Under Analysis:** `example.com`
**Analysis Date:** 2026-03-25 21:47:03 UTC
**System:** AMAZONAS (Automated Malicious Domain Analysis System)

## Your Role

You are a cybersecurity expert specializing in malicious domain detection and threat intelligence. Your task is to:

1. **Review** all provided evidence about this domain
2. **Analyze** the classification scores, DNS data, web content, and traffic patterns
3. **Validate** or challenge the ML recommendation
4. **Provide** a final recommendation: ALLOW, BLOCK, or NEEDS MORE INVESTIGATION
5. **Explain** your reasoning with specific evidence references

Consider common threat indicators:
- Domain Generation Algorithm (DGA) patterns
- Typosquatting of legitimate brands
- Suspicious lexical patterns
- DNS misconfigurations or anomalies
- Minimal or suspicious web content
- Unusual traffic patterns
- Geographic anomalies in traffic sources

## Phase I Classification Results

**Overall Risk Score:** 0.031 / 1.0
**Classification Category:** BENIGN

### Individual Classifier Results:
  - **DGA_Classifier**: Score 0.021 (BENIGN)
  - **Typosquatting_Classifier**: Score 0.008 (BENIGN)
  - **Lexical_Classifier**: Score 0.031 (BENIGN)

**Interpretation:**
- Score < 0.5: Likely benign
- Score 0.5-0.75: Gray area (requires Phase II analysis)
```

```
- Score > 0.75: Likely malicious

## ML-Based Recommendation

**Recommendation:** ALLOW LIST
**Confidence Level:** 95.0%

### Reasoning:
1. Low classification score (0.03)
2. Valid DNS and HTML content
3. High legitimate traffic (184302 queries)

### Key Features Analyzed:
- Base Classification Score: 0.031
- Has Valid DNS Response: Yes
- DNS Response Code: 0
- Has HTML Content: Yes
- HTML Size: 1,256 bytes (1.2 KB)
- Passive DNS Query Count: 184,302
- [OK] High traffic volume suggests legitimate, active domain
- Unique Querier IPs: 9,847

## Phase II Monitoring Evidence

### Active DNS Analysis
- **Query Type:** A
- **Response Code:** 0 - NOERROR (successful)
- **TTL (Time To Live):** 3600 seconds
- **Nameserver:** a.iana-servers.net
- **Last Updated:** 2026-03-24 18:02:11

**DNS Response Details:**
'''json
{
  "A": ["93.184.216.34"],
  "NS": ["a.iana-servers.net", "b.iana-servers.net"],
  "MX": [],
  "TXT": ["v=spf1 -all"]
}
'''

### Web Content Analysis
- **Archive Status:** [OK] Available for download
- **File Size:** 1,256 bytes (1.23 KB)
- **Archive Hash (SHA256):** `
   a1b2c3d4e5f67890abcdef1234567890abcdef1234567890abcdef1234567890`
- **Created:** 2026-03-20 09:14:55
```

```
85
86 **Analysis Notes:**
87 - Very small HTML (<5KB): Often indicates parking pages or minimal content
88 - Large HTML (>500KB): Typically legitimate sites with full functionality
89 - Medium size (5-500KB): Requires content inspection
90
91 ### Passive DNS Analysis (ENTRADA2)
92 - **Total Queries:** 184,302
93 - **Querier IP:** 200.133.0.1
94 - **Geographic Location:** US
95 - **ASN:** AS15169 (Google LLC)
96 - **First Seen:** 2026-03-10 00:03:21
97 - **Last Seen:** 2026-03-25 21:44:59
98 - **Query Types:** A, AAAA, NS, TXT
99 - **Response Codes:** NOERROR
100 - **DNSSEC Validated:** Yes
101
102 **Traffic Pattern Analysis:**
103 - High query count (>100): Indicates legitimate use or active botnet
104 - Geographic diversity: Multiple countries suggest legitimate service
105 - Single IP with low queries: Possible targeted attack or testing
106 - DNSSEC validation: Adds legitimacy to domain configuration
107
108 ### Combined Risk Assessment
109 **Phase II Risk Score:** 0.028 / 1.0
110
111 ## Your Analysis Task
112
113 Please provide a comprehensive security analysis addressing:
114
115 ### 1. Evidence Assessment
116 - Which pieces of evidence are most significant?
117 - Are there any red flags or concerning patterns?
118 - What evidence suggests benign vs. malicious intent?
119
120 ### 2. ML Recommendation Validation
121 - Do you agree with the ML recommendation?
122 - What factors support or contradict the recommendation?
123 - Is the confidence level appropriate given the evidence?
124
125 ### 3. Threat Classification
126 Based on all evidence, classify this domain as:
127 - **ALLOW (Benign)**: Safe to whitelist
128 - **BLOCK (Malicious)**: Should be blocked immediately
129 - **INVESTIGATE**: Needs more evidence or monitoring
130
131 ### 4. Specific Recommendations
```

```
- Should this domain be allowed, blocked, or monitored further?
- What additional evidence would be helpful?
- Are there any immediate actions security teams should take?

### 5. Risk Summary
Provide a 2-3 sentence summary explaining your final determination and key
    reasoning.

---

**Format your response with clear sections and bullet points for easy
    review by security analysts.**
```

Listing F.1 – *Prompt* gerado para domínio seguro

G RESPOSTA DA IA PARA DOMÍNIO POTENCIALMENTE MALICIOSO

A Listagem G.1 apresenta a resposta da IA para o cenário de domínio potencialmente malicioso.

```
## 1. Evidence Assessment

### Most Significant Indicators (High Risk)

* **Typosquatting Pattern (Critical)**

  * Domain: `paypa1-secure-login.com`
  * Clear impersonation of **paypal** using:

    * `paypa1`  substitution of lowercase **l with 1**
  * Combined with **secure-login**, a common phishing lure phrase
    -> This is a **classic credential harvesting domain pattern**

* **Very High Classification Scores**

  * Overall: **0.941 (MALICIOUS)**
  * Typosquatting: **0.941 (MALICIOUS)**
  * Lexical: **0.883 (MALICIOUS)**
    These are strong, consistent indicators of intentional abuse.

* **Suspicious Hosting Infrastructure**

  * Nameservers: `hostingprovider-anon.ru`
  * IP: `185.220.101.47`
  * Characteristics:

    * Anonymous / offshore hosting
    * Commonly associated with abuse-tolerant providers
      -> Strong infrastructure-level red flag

* **Minimal HTML Content (487 bytes)**

  * Extremely small page size (<1KB)
  * Likely uses:

    * Redirects
    * Lightweight phishing form
    * Obfuscated scripts loaded externally
      -> Typical of phishing landing pages

* **Low & Concentrated Traffic**
```

```
42
43   * Only **43 queries** and **3 unique IPs**
44   * Single ASN (Claro S.A., Brazil)
45   * Pattern consistent with:
46
47     * Early-stage phishing campaign
48     * Targeted attack or testing phase
49
50 * **Short TTL (300 seconds)**
51
52   * Suggests potential **fast infrastructure changes**
53   * Common in malicious campaigns to evade detection
54
55 * **No DNSSEC**
56
57   * Lack of validation adds to distrust (not decisive alone, but relevant
       combined with other signals)
58
59 ---
60
61 ### Benign Indicators (Minimal / Negligible)
62
63 * Valid DNS response (NOERROR)  expected even for malicious domains
64 * DGA score low  domain is human-crafted (consistent with phishing)
65
66 -> No meaningful benign evidence to offset risk
67
68 ---
69
70 ## 2. ML Recommendation Validation
71
72 ### [OK] Agreement with ML: **YES (Strongly)**
73
74 * ML Recommendation: **BLOCK LIST**
75 * Confidence: **95%**  Appropriate and justified
76
77 ### Supporting Factors
78
79 * Extremely high typosquatting score
80 * Strong lexical phishing indicators (secure-login)
81 * Known malicious hosting patterns
82 * Behavioral indicators (low, targeted traffic)
83 * Small HTML consistent with phishing kits
84
85 ### No Contradictions Found
86
87 All signals align consistently toward malicious intent.
```

```
88
89 ---
90
91 ## 3. Threat Classification
92
93 ### [BLOQUEAR] **FINAL VERDICT: BLOCK (Malicious)**
94
95 This domain is highly likely to be used for **phishing and credential theft
       **, specifically impersonating PayPal.
96
97 ---
98
99 ## 4. Specific Recommendations
100
101 ### Immediate Actions (HIGH PRIORITY)
102
103 * [BLOQUEAR] **Block domain at all layers**:
104
105   * DNS filtering
106   * Web proxies
107   * Email gateways
108 * [BLOQUEAR] Add to **threat intelligence feeds / blocklists**
109
110 ### Detection & Hunting
111
112 * Search logs for:
113
114   * DNS queries to this domain
115   * HTTP/S connections to `185.220.101.47`
116 * Identify potential compromised users
117
118 ### User Protection
119
120 * Trigger **phishing awareness alerts** if exposure detected
121 * Reset credentials for any potentially affected accounts
122
123 ### Infrastructure Monitoring
124
125 * Monitor for:
126
127   * Similar domains (e.g., `paypaI`, `paypal-secure-login`, etc.)
128   * Same nameservers (`hostingprovider-anon.ru`)
129   * IP reuse across campaigns
130
131 ### Additional Evidence (Optional)
132
133 * Full HTML/content analysis (likely phishing form)
```

```
* SSL certificate inspection (often self-signed or recently issued)
* Sandbox detonation (to confirm credential harvesting behavior)

---

## 5. Risk Summary

This domain exhibits clear and deliberate typosquatting of PayPal, combined
    with phishing-oriented keywords and suspicious hosting infrastructure.
   The extremely high ML risk score, minimal content, and low-volume
   targeted traffic strongly indicate an active phishing campaign.
   Immediate blocking is required to prevent credential compromise and user
    exploitation.
```

Listing G.1 – Resposta da IA para domínio potencialmente malicioso

H RESPOSTA DA IA PARA DOMÍNIO BENIGNO

A Listagem H.1 apresenta a resposta da IA para o cenário de domínio benigno.

```
## 1. Evidence Assessment

### Most Significant Indicators

* **Extremely Low Risk Scores (0.031 overall)**
  All classifiers (DGA, typosquatting, lexical) strongly indicate benign
    characteristics. No randomness, brand impersonation, or suspicious
    naming patterns detected.

* **Well-formed DNS Configuration**

  * Valid A record (`93.184.216.34`)
  * Trusted nameservers (`iana-servers.net`)
  * DNSSEC validation present [OK]
    These are strong indicators of a legitimate, well-maintained domain.

* **High Passive DNS Traffic**

  * **184,302 queries** and **9,847 unique IPs**
  * Associated with **AS15169 (Google LLC)**
    This strongly suggests widespread legitimate usage rather than targeted
     or botnet-driven traffic.

* **Stable, Clean DNS Behavior**

  * Consistent NOERROR responses
  * No fast-flux behavior or frequent IP changes
  * TTL of 3600 seconds (normal)

### Minor Observations / Potential Flags

* **Small HTML size (1.2 KB)**

  * Could indicate:

    * Placeholder page
    * Test domain
    * Static minimal content
  * However, this alone is **not suspicious**, especially for well-known
    reserved/example domains.

* **Single dominant querier IP (Google ASN)**
```

```
39
40   * Likely due to DNS resolvers (e.g., public DNS services), not malicious
       concentration.
41
42 ### Benign Signals Summary
43
44 [OK] Clean lexical profile
45 [OK] No typosquatting indicators
46 [OK] Stable DNS + DNSSEC
47 [OK] High, diverse traffic volume
48 [OK] Reputable infrastructure (IANA, Google ASN)
49
50 [NENHUM] No meaningful malicious indicators observed
51
52 ---
53
54 ## 2. ML Recommendation Validation
55
56 ### Agreement with ML: [OK] YES
57
58 The ML system recommends **ALLOW LIST (95% confidence)**  this is
       appropriate.
59
60 ### Supporting Factors
61
62 * Extremely low classification scores across all models
63 * Strong infrastructure legitimacy (IANA nameservers are highly trusted)
64 * Traffic patterns consistent with legitimate global usage
65 * No behavioral anomalies (no DGA, no fast flux, no suspicious spikes)
66
67 ### Confidence Level Assessment
68
69 * **95% confidence is justified**, possibly even conservative
70 * The domain exhibits characteristics of a **reference or canonical domain
       **, often used in testing/documentation
71
72 No contradictory evidence found.
73
74 ---
75
76 ## 3. Threat Classification
77
78 ### [OK] **FINAL VERDICT: ALLOW (Benign)**
79
80 This domain is safe to whitelist.
81
82 ---
```

```
83
84 ## 4. Specific Recommendations
85
86 ### Immediate Actions
87
88 * [OK] Add to **allowlist / whitelist**
89 * [OK] Exclude from future alerting (reduce noise in SOC pipelines)
90
91 ### Monitoring
92
93 * No active monitoring required beyond standard baseline logging
94
95 ### Additional Evidence (Optional, Not Required)
96
97 If further validation is desired:
98
99 * Historical WHOIS data (expected to be stable)
100 * Content inspection (likely static placeholder)
101 * Reputation feeds (should confirm benign/global trust)
102
103 ---
104
105 ## 5. Risk Summary
106
107 The domain demonstrates all characteristics of a legitimate, stable, and
    widely trusted resource, including clean lexical structure, strong DNS
    integrity, and high-volume global traffic. There are no indicators of
    malicious behavior or abuse. Based on all available evidence, the domain
     is safe to allow and poses negligible security risk.
```

Listing H.1 – Resposta da IA para domínio benigno

I OUTPUT DOS COLETORES

```
1
2 [*] Starting download from: http://google.com
3     -> Downloaded HTML: google.com.html
4     -> Found external asset: http://www.gstatic.com/images/branding/
    searchlogo/ico/favicon.ico
5         '- Downloaded as: ext_www_gstatic_com_favicon.ico
6     -> Found internal asset: http://google.com/xjs/_/ss/k=xjs.hd.
    eNk6R3pk9ko.L.B1.O/am=
    QAhQKhAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAIAAAAAAAAAQAAAAAADEAggAAAAABwAAK
    /d=1/ed=1/br=1/rs=ACT90oEsa85GsHd8U8kE0ds8q_j3tNCWGw/m=cdos,hsm,jsa,
    mb4ZUb,cEt90b,SNUn3,qddgKe,sTsDMc,dtl0hd,eHDfl,YV5bee,d,csi
7         '- Downloaded as: m=cdos,hsm,jsa,mb4ZUb,cEt90b,SNUn3,qddgKe,sTsDMc,
    dtl0hd,eHDfl,YV5bee,d,csi
8     -> Found external asset: https://www.gstatic.com/og/_/ss/k=og.asy.
    dFw2Fd13_TI.L.W.O/m=ll_tdm,adcgm3,ll_fw,abld/excm=/d=1/ed=1/ct=zgms/rs=
    AA2YrTv4eqWEnGmAQ2LU0Zvd3kGUMevlTQ
9         '- Downloaded as: ext_www_gstatic_com_rs=
    AA2YrTv4eqWEnGmAQ2LU0Zvd3kGUMevlTQ
10    -> Found internal asset: http://google.com/xjs/_/js/k=xjs.hd.nl.
    EL8H24I_OVQ.2019.O/...
11        '- Downloaded as: m=cdos,hsm,jsa,mb4ZUb,cEt90b,SNUn3,qddgKe,sTsDMc,
    dtl0hd,eHDfl,YV5bee,d,csi
12    -> Found external asset: https://www.gstatic.com/og/_/js/k=og.asy.en_US
    .RoVEtdDprBg.2019.O/rt=j/m=_ac,_awd,ada,lldp,qads,abld/exm=/d=1/ed=1/rs=
    AA2YrTt3RHefO6R83G9NNN8UC_S_3dDL8A
13        '- Downloaded as: ext_www_gstatic_com_rs=
    AA2YrTt3RHefO6R83G9NNN8UC_S_3dDL8A
14    -> Found external asset: https://fonts.gstatic.com/s/i/productlogos/
    googleg/v6/24px.svg
15        '- Downloaded as: ext_fonts_gstatic_com_24px.svg
16 [*] Download phase complete. Total assets found: 6
17
18 [*] Zipping 6 assets into 'google_com_archive_20250929_114638.zip'...
19    -> Added to zip: google.com.html
20    -> Added to zip: ext_www_gstatic_com_favicon.ico
21    -> Added to zip: m=cdos,hsm,jsa,mb4ZUb,cEt90b,SNUn3,qddgKe,sTsDMc,
    dtl0hd,eHDfl,YV5bee,d,csi
22    -> Added to zip: ext_www_gstatic_com_rs=
    AA2YrTv4eqWEnGmAQ2LU0Zvd3kGUMevlTQ
23    -> Added to zip: ext_www_gstatic_com_rs=
    AA2YrTt3RHefO6R83G9NNN8UC_S_3dDL8A
24    -> Added to zip: ext_fonts_gstatic_com_24px.svg
25
```

```
[SUCCESS] Archive 'google_com_archive_20250929_114638.zip' created
    successfully!

[*] Verifying file creation...
-rw-r--r-- 1 root root 507K Sep 29 11:46 google_com_archive_20250929_114638
    .zip
```

Listing I.1 – Coletor Web ao site google.com

```
[*] Starting feature extraction for 2 domains in batches of 100000
[*] Processing batch 1 of 1 (2 domains)
[*] Starting asynchronous feature extraction for 2 domains with concurrency
    1024

100%|| 2/2 [00:00<00:00, 22.99it/s]

[*] Processing domain: google.com
[*] Processing domain: acmesecurity.org
[*] Performing DoH query for SOA record on google.com
[*] Performing DoH query for SOA record on acmesecurity.org
[*] Successfully retrieved SOA record for google.com
[*] Performing DoH query for A record on google.com
[*] Performing DoH query for MX record on google.com
[*] Performing DoH query for NS record on google.com
[*] Performing DoH query for AAAA record on google.com
[*] Performing DoH query for CNAME record on google.com
[*] Successfully retrieved SOA record for acmesecurity.org
[*] Successfully retrieved A record for google.com
[*] Performing DoH query for A record on acmesecurity.org
[*] Performing DoH query for MX record on acmesecurity.org
[*] Performing DoH query for NS record on acmesecurity.org
[*] Performing DoH query for AAAA record on acmesecurity.org
[*] Performing DoH query for CNAME record on acmesecurity.org
[*] Successfully retrieved A record for acmesecurity.org
[*] Successfully retrieved NS record for google.com
[*] Successfully retrieved MX record for acmesecurity.org
[*] Successfully retrieved MX record for google.com
[*] Successfully retrieved CNAME record for google.com
[!] Error performing DoH query for CNAME record on google.com: 'Answer'
[*] Successfully retrieved AAAA record for google.com
[*] Parsing MX records...
[*] Parsing AAAA records...
[*] Parsing A and CNAME records...
[*] Parsing SOA and NS records...
[*] Finished processing domain: google.com
[*] Successfully retrieved NS record for acmesecurity.org
[*] Successfully retrieved CNAME record for acmesecurity.org
```

```
[!] Error performing DoH query for CNAME record on acmesecurity.org: 'Answer'
[*] Successfully retrieved AAAA record for acmesecurity.org
[!] Error performing DoH query for AAAA record on acmesecurity.org: 'Answer'
[*] Parsing MX records...
[*] Parsing AAAA records...
[*] Parsing A and CNAME records...
[*] Parsing SOA and NS records...
[*] Finished processing domain: acmesecurity.org
[*] Asynchronous feature extraction complete.
[*] Feature extraction complete. Returning DataFrame.
[*] Contents of dataframe:
              domain  MX Count  MX Medium TTL  AAAA Count  AAAA Stdev TTL  \
0        google.com         1          207.0           1               0
1  acmesecurity.org         5          407.8           0               0

   AAAA Medium TTL  A Count  A Stdev TTL  A Medium TTL  CNAME Count  NS Count  \
0              119        1            0           180            0         4
1                0        1            0          1799            0         2

   SOA Count  SOA Serial Length  SOA Refresh  SOA Retry  SOA Expire Length  \
0          1                  9          900        900                  4
1          1                 10        43200       3600                  6

   SOA Minimum  SOA Stdev TTL  SOA Medium TTL
0           60  136739.600000   273485.200000
1         3601     848.999542     2400.333333
```

Listing I.2 – Coletor de DNS Ativo para os domínios acmesecurity.org e google.com.

```
    Connecting to TRINO database...
Connection established
Cursor created
Trying to query TRINO database...
Fetched all rows
           0   1   2      3      4      5   6   7   8   9   10  11     12  13  \
0  trkbid.com   6  17  53249  False  False   0   5   8   0  65  BR  53166   2
1  trkbid.com   6  17  41134  False  False   0   5   8   0   1  BR  53166   2
2  trkbid.com   4  17  54906   True  False   0   1   1   0  65  BR  53166   2
```

```
3  trkbid.com   4  17  59501  False  False   0   0   1   0  65  BR  53166
    2
4  trkbid.com   4  17  41729   True  False   1   1   0   0   1  BR  53166
    2
5  trkbid.com   4  17  49754  False  False   1   0   0   0   1  BR  53166
    2

    14
0  542
1  542
2   89
3   81
4   55
5   44
```

Listing I.3 – Resultado da coleta de pDNS para o domínio malicioso trkbid.com.

DADOS CURRICULARES

IDENTIFICAÇÃO	
	Gabriele Zancheta Scavazini 04/01/2001
Nacionalidade	Brasileira
Nome em citações bibliográficas	Scavazini, Gabriele Scavazini, G.
Currículo Lattes	http://lattes.cnpq.br/8910748381683958
ORCID	https://orcid.org/0009-0006-5917-0938
FORMAÇÃO ACADÊMICA	
2020/2024	Bacharelado em Ciência da Computação Universidade Estadual Paulista Júlio de Mesquita Filho, UNESP, Brasil
PARTICIPAÇÃO EM EVENTOS CIENTÍFICOS	
Workshop Pós-Graduaçao., 2012, (São José do Rio Preto). AMAZONAS - Módulo de scoring e monitoramento de domínios recém registrados. 2025. (Apresentação de Trabalho/Conferência ou palestra). GTER (São Paulo). DNSCheck Framework para Administradores de Redes Contra Ameaças Baseadas em DNS. 2025. (Apresentação de Trabalho/Conferência ou palestra).	