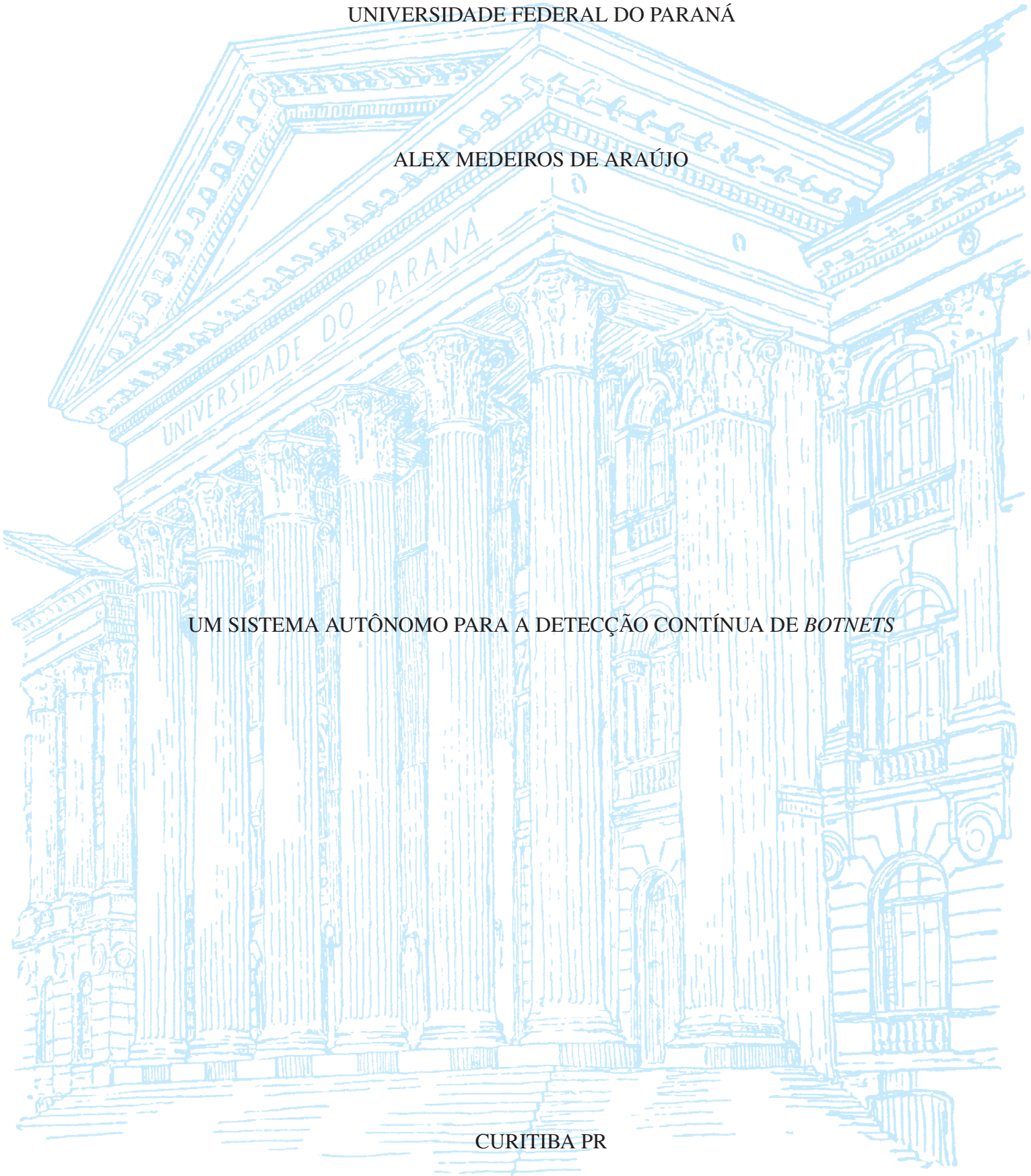
UNIVERSIDADE FEDERAL DO PARANÁ

ALEX MEDEIROS DE ARAÚJO



UM SISTEMA AUTÔNOMO PARA A DETECÇÃO CONTÍNUA DE *BOTNETS*

CURITIBA PR

2022

ALEX MEDEIROS DE ARAÚJO

UM SISTEMA AUTÔNOMO PARA A DETECÇÃO CONTÍNUA DE *BOTNETS*

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Computação*.

Orientador: Profa. Dra. Michele Nogueira Lima.

CURITIBA PR

2022

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)
UNIVERSIDADE FEDERAL DO PARANÁ
SISTEMA DE BIBLIOTECAS – BIBLIOTECA CIÊNCIA E TECNOLOGIA

Araújo, Alex Medeiros de
Um sistema autônomo para a detecção contínua de botnets. / Alex Medeiros de Araújo. – Curitiba, 2022.
1 recurso on-line : PDF.

Dissertação (Mestrado) - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática.

Orientadora: Profa. Dra. Michele Nogueira Lima.

1. Aprendizado de computador. 2. Redes de computação – Segurança. 3. Detecção de botnet. I. Universidade Federal do Paraná. Programa de Pós-Graduação em Informática. II. Nogueira, Lima, Michele. III. Título.

Bibliotecária: Roseny Rivelini Morciani CRB-9/1585

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **ALEX MEDEIROS DE ARAUJO** intitulada: **Um Sistema Autônomo para Detecção Contínua de Botnets**, sob orientação do Prof. Dr. MICHELE NOGUEIRA LIMA, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.
A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 04 de Outubro de 2022.

Assinatura Eletrônica
14/10/2022 09:44:42.0
MICHELE NOGUEIRA LIMA
Presidente da Banca Examinadora

Assinatura Eletrônica
10/10/2022 16:10:06.0
EDELBERTO FRANCO SILVA
Avaliador Externo (UNIVERSIDADE FEDERAL DE JUIZ DE FORA)

Assinatura Eletrônica
06/10/2022 17:49:01.0
ALDRI LUIZ DOS SANTOS
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Aos meus pais, bases fundamentais
da minha educação como indivíduo.

AGRADECIMENTOS

Aos meus pais, Ana Santana e Augusto Neto, que foram os pilares da minha formação como ser humano e pelo amor, incentivo e apoio incondicional durante todo o processo do mestrado

Agradeço a minha professora e orientadora Dra. Michele Nogueira Lima, pelo suporte, orientação e conselhos durante todo o processo de pesquisa. Seus conhecimentos e experiências foram inestimáveis para o sucesso deste trabalho.

Aos membros da banca examinadora pela disposição em avaliar e fornecer críticas construtivas. Suas sugestões e comentários ajudaram a aprimorar e aperfeiçoar este trabalho.

Aos professores e colegas do programa de pós-graduação em Informática da Universidade Federal do Paraná e aos colegas de laboratório CCSC, que forneceram um ambiente de apoio e encorajamento para minha pesquisa. Agradeço também de forma especial ao meu colega Anderson, pela ajuda e apoio em diversas etapas deste trabalho. Sua contribuição foi fundamental para que eu pudesse alcançar os resultados obtidos.

Também gostaria de agradecer ao Prof. Dr. João Paulo, que foi meu primeiro mentor e me ensinou a importância da pesquisa e da dedicação ao conhecimento. Sua orientação e aulas me proporcionaram um aprendizado valioso e me ajudaram a desenvolver habilidades essenciais. E ao Prof. Dr. João Borges pelas suas recomendações e apoio ao longo da minha jornada acadêmica.

RESUMO

Atualmente, vivemos uma revolução tecnológica marcada pelo desenvolvimento de sistemas digitais mais sofisticados e integrados e a ubiquidade da Internet. Essa revolução resultou em uma amplificação do ciberespaço e na abertura do seu acesso, transformando a sociedade e a economia global. Os avanços tecnológicos estão associados com o surgimento de novos tipos de ameaças e riscos. Nesse sentido, à medida que o acesso à Internet se torna mais amplo, é possível observar o surgimento de novas ameaças relacionadas ao ciberespaço. Para evitar prejuízos causados por essas ameaças, as empresas investem fortemente em sistemas de segurança cibernética. Para padronizar as diretrizes e as práticas existentes relacionados à segurança cibernética e facilitar a sua implantação em diferentes organizações, surgiu o *Cybersecurity Framework* (CSF). O CSF é organizado em cinco funções: identificar, proteger, detectar, responder e recuperar. Este trabalho foca na função de detecção, mais especificamente na detecção de botnets, redes de dispositivos comprometidos controlados remotamente por um atacante (botmaster) para a realização de ataques cibernéticos. Os ataques realizados por *botnets* têm se destacado, dado à sua diversidade e o potencial para causar danos econômicos, sociais, institucionais e operacionais. A detecção de *botnets* é dificultada por vários motivos, sendo estes principalmente a diversidade de *botnets* existentes e o desenvolvimento de novas variantes mais eficazes em evitar as técnicas de detecção. Na literatura, as soluções existentes para a detecção de *botnets* geralmente assumem um contexto estático, ou seja, a distribuição dos dados permanece a mesma durante todo o ciclo de vida da aplicação. Porém, uma das consequências da diversidade de ataques realizados por *botnets* é a mudança inesperada no comportamento da rede. Desse modo, este trabalho aborda o problema de como detectar *botnets* de maneira eficaz com o mínimo de intervenção humana, diante de um ambiente caracterizado por mudanças frequentes no contexto de ameaças. Para este fim, é proposto um sistema de detecção de *botnets* que atua em ambientes dinâmicos de maneira autônoma, denominado de ANTE (do inglês, ***A**uto**N**omous bo**T**net d**E**tection*). A avaliação do sistema ANTE, por meio da utilização de um conjunto diversificado de cenários de ataques, comprovaram a sua eficácia em detectar *botnets*. Além disso, foi possível observar que o sistema ANTE seleciona técnicas de detecção diferentes para cenários distintos, indicando a sua capacidade de considerar diferentes aspectos de um ataque na escolha da melhor forma de realizar a detecção de *botnets*.

Palavras-chave: Aprendizado de maquina. Detecção de botnet. AutoML. Lifelong Machine Learning

ABSTRACT

Currently, we are in the midst of a technological revolution marked by the development of more sophisticated and integrated digital systems and the ubiquity of the Internet. This revolution resulted in the amplification of cyberspace and its widespread adoption, transforming society and the global economy. Technological advances are usually associated with the emergence of new types of threats and risks. In this sense, as access to the Internet becomes broader, it is possible to observe the emergence of new threats related to cyberspace. To mitigate these threats, companies invest heavily in cybersecurity systems. To standardize the existing guidelines and practices related to cybersecurity and facilitate its implementation in different organizations, the Cybersecurity Framework (CSF) was created. The CSF is organized into five functions: identify, protect, detect, respond and recover. This work focuses on the detection function, more specifically on the detection of botnets, networks of compromised devices remotely controlled by an attacker (botmaster) to carry out cyber attacks. Attacks carried out by botnets have stood out, given their diversity and potential to cause economic, social, institutional, and operational damage. Botnet detection is a difficult task for several reasons, mainly the diversity of existing botnets and the development of new variants that are more efficient in avoiding detection techniques. In the literature, the existing solutions for botnet detection generally assume a static context, that is, the data distribution remains the same throughout the application lifecycle. However, one of the consequences of the diversity of attacks carried out by botnets is unexpected changes in network behavior. Thus, this work addresses the problem of how to detect botnets effectively with minimal human intervention, in an environment characterized by frequent changes in the context of threats. For this purpose, is proposed a autonomous botnet detection system called ANTE (**A**uto**N**omous bo**T**net d**E**tection). The evaluation of the ANTE system, through the use of a diverse set of attack scenarios, proved its effectiveness in detecting botnets. Furthermore, it was observed that the ANTE system selects different detection techniques for different scenarios, indicating its ability to consider different aspects of an attack when choosing the best way to perform detection.

Keywords: Machine learning. Botnet detection. AutoML. Lifelong Machine Learning

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

AutoML	*Automated Machine Learning* Aprendizagem de Máquina Automatizada
C&C	*Command and Control* Comando e Controle
CEA	*Cybersecurity Enhancement Act* Lei de Aprimoramento da Segurança Cibernética
CSF	*Cybersecurity Framework*
CSV	*Comma-separated values* Valores Separados por Vírgulas
D3	*Discriminative Drift Detector*
DDoS	*Distributed Denial of Service* Negação de Serviço Distribuído
DNS	*Domain Name Service* Serviço de Nomes e Domínios
DoS	*Denial of Service* Negação de Serviço
DVR	*Digital Video Recorder* Gravadores Digitais de Vídeo
HTTP	*Hypertext Transfer Protocol* Protocolo de Transferência de Hipertexto
IA	*Inteligência Artificial*
ICMP	*Internet Control Message Protocol* Protocolo de Mensagens de Controle da Internet
IP	*Internet Protocol* Protocolo de Internet
IRC	*Internet Relay Chat*
IoT	*Internet of Things* Internet das Coisas
MaaS	*Malware-as-a-Service*
NIST	*National Institute of Standards and Technology* Instituto Nacional de Padrões e Tecnologia
OMS	Organização Mundial da Saúde
P2P	*Peer-to-Peer* Ponto-a-Ponto
TCP	*Transmission Control Protocol*

	Protocolo de Controle de Transmissão
UDP	*User Datagram Protocol* Protocolo de Datagrama do Usuário
UTC	Tempo Universal Coordenado
VPN	*Virtual Private Network* Rede Privada Virtual

LISTA DE SÍMBOLOS

p	precisão
r	*recall*

SUMÁRIO

1 INTRODUÇÃO

Atualmente, vivemos uma revolução tecnológica. Considerada por Schwab (2016) como a quarta revolução industrial, essa revolução foi iniciada na virada do século e tem como base a revolução digital. A quarta revolução industrial é marcada pela influência do ciberespaço na cultura contemporânea. O ciberespaço, conforme apresentado no trabalho de Monteiro (2007), é o espaço criado pela conexão entre as infraestruturas de comunicação e as pessoas. O desenvolvimento de tecnologias digitais mais sofisticadas (e integradas) e a ubiquidade da Internet constituem uma das condições gerais para a amplificação do ciberespaço, transformando a sociedade e a economia global (Schwab, 2016).

A evolução do ciberespaço resultou em uma alteração profunda na maneira como vivemos, trabalhamos e nos relacionamos. Como exemplo, é possível citar a realização de compras *online*. Nessa atividade, as pessoas adquirem bens e uma gama crescente de serviços, incluindo entretenimento e mídia, através da Internet. No Brasil, milhões de pessoas começaram a comprar pela Internet pela primeira vez. Em 2021, houve um aumento de 41% na entrada de novos consumidores do *e-commerce* (Ebit, 2021). Uma pesquisa realizada pela PricewaterhouseCoopers (PwC), uma das maiores multinacionais de auditoria do mundo, mostrou que 29% dos 8.700 consumidores pesquisados em 22 territórios pensam que vão gastar mais com entretenimento doméstico - filmes, videogames, música e livros - nos próximos seis meses, contra apenas 20% que pensaram que gastariam menos (PwC, 2021).

Outro exemplo dessa alteração é observado com a adoção do regime de trabalho remoto. Em 2020, a Organização Mundial da Saúde (OMS) declarou que o surto de Covid-19 se tornaria uma pandemia (Cucinotta e Vanelli, 2020). Nos dias seguintes, a maioria dos órgãos federativos passaram a restringir atividades sociais e econômicas a fim de conter a transmissão do vírus. Seguindo essas restrições, muitas empresas interromperam as operações no local de trabalho e fizeram a transição para o trabalho pela Internet. De acordo com Bick et al. (2021), de todos os empregados em maio de 2021 nos Estados Unidos, 35,2% trabalhavam em casa. Como consequência, o tráfego da Internet relacionado a aplicativos de trabalho remoto, como *Virtual Private Networks* (VPN) e aplicativos de videoconferência, teve um aumento de até 200% (Feldmann et al., 2021).

Essas mudanças tecnológicas geralmente estão associadas ao surgimento de novos tipos de ameaças e riscos. Por exemplo, a primeira Revolução Industrial trouxe fábricas e ferrovias junto com roubos de trens e aumento de contrabando, enquanto a segunda Revolução Industrial trouxe eletricidade e carros junto com roubos de automóveis (Holt e Bossler, 2020; McLynn, 2013). Nesse sentido, à medida que o acesso à Internet se torna mais amplo, é possível observar a realização de novos tipos de crimes no ciberespaço, introduzindo novos riscos a indivíduos e organizações. Dentre esses novos tipos de crimes popularizados pelo uso da Internet é possível identificar fraudes de cartão de crédito, *malware*, spam, ataques de *phishing* e ataques realizados por *botnets* (Holt e Bossler, 2020).

Uma *botnet* consiste de uma rede de dispositivos comprometidos controlados remotamente. Geralmente, os dispositivos integrantes de uma *botnet* são cooptados por meio da instalação de *softwares* maliciosos em dispositivos com vulnerabilidades (Silva et al., 2013). Após infectado, os dispositivos são utilizados pelo *botmaster* para diferentes fins, dependendo da sua motivação. Dentre as motivações estão os ganhos financeiros, entretenimento, ego, ingresso em causas sociais ou estatuto social (Rodríguez-Gómez et al., 2011).

Dentre as possíveis motivações do *botmaster* os ataques realizados por *botnets*, em sua maior parte, são realizados visando ganhos financeiros (Bottazzi e Me, 2014). Nesse sentido, é possível observar vários casos de usos (Putman et al., 2018). Em 2012, o ataque denominado "Eurograbber" resultou em um roubo estimado de 36 milhões de euros entre 30.000 clientes bancários de vários bancos na Europa. Esse ataque foi realizado por uma versão modificada de um *malware* de *botnet* (Kalige et al., 2012). Relatórios indicam que a *botnet* "ZeroAccess" gerou uma perda de US$ 900.000 por dia para anunciantes com a realização de fraude de cliques. Nesse ataque um *bot* se disfarça como um usuário legítimo de uma página da *web* e simula um clique em um anúncio, os anunciantes acreditando ser um usuário legítimo pagam pela interação (F-Secure, 2012). Uma análise de um provedor de *botnet* por aluguel revelou um lucro de US$ 597.000 em dois anos com a realização de 915.000 ataques (Brunt et al., 2017).

Para evitar prejuízos causados por esses ataques, as empresas precisam investir fortemente em sistemas de segurança cibernética. As estimativas do custo total anual aos negócios, causado por ciberataques, estão na ordem de grandeza de US$ 500 bilhões (Schwab, 2016). Na pesquisa realizada por Barbier et al. (2016), 69% dos executivos entrevistados expressaram que a digitalização é essencial para as estratégias de crescimento atuais. Dentre esses, 64% citaram a segurança cibernética como um impulsionador significativo do sucesso de produtos digitais, serviços e modelos de negócios.

Tendo em vista a crescente adoção de sistemas de segurança cibernética pelas empresas, fazem-se necessárias a organização e a coleta de recursos processos utilizadas por esses sistemas. Para esse fim, surgiu o *Cybersecurity Framework* (CSF) (NIST, 2018). O CSF define uma metodologia baseada em padrões, diretrizes e práticas existentes para avaliar e gerenciar riscos relacionados à segurança cibernética. O CSF é organizado em cinco funções: identificar, proteger, detectar, responder e recuperar (NIST, 2018). Este trabalho foca na função de detecção. Essa função consiste no desenvolvimento e na implementação de atividades para identificar a ocorrência de eventos que comprometem a política de segurança cibernética de uma organização.

1.1 PROBLEMA

Em 2021, 30% de todo o tráfego da Internet foi causado por *bots* (Imperva, 2021). Parte desse aumento é proporcionado pelo mercado de *Malware-as-a-Service* (MaaS). Este mercado evoluiu de uma infraestrutura de redes de *bots* amadora para organizações extremamente organizadas, onde é possível alugar *botnets* como um serviço de assinatura, com garantia de qualidade e também atendimento ao cliente. Os serviços de aluguel de *botnets* tornaram mais acessível a realização de ataques cibernéticos, especialmente para usuários não-técnicos (Holt e Bossler, 2020).

Além de mais frequentes, os ataques realizados por *botnets* cresceram em volume. Em outubro de 2016, as organizações Krebs e OVH foram vítimas de ataques de negação de serviço que ultrapassaram 600 Gbps de dados gerados (Kambourakis et al., 2019). Nos dias seguintes, a empresa Dyn sofreu interrupções devido a um ataque realizado por mais de 100.000 dispositivos, tornando grandes partes da Internet inacessível (Kambourakis et al., 2019). Alguns dos maiores provedores de serviços da Internet como Twitter, Spotify, Amazon e outros foram afetados pelos ataques (Kambourakis et al., 2019). Esse aumento de volume se dá em parte pelo aumento de dispositivos da Internet das Coisas (do inglês, *Internet of Things* - IoT). Os dispositivos da IoT possuem limitações na largura de banda, baixo poder de processamento e pouca memória. Tais aspectos, agregados a vulnerabilidades no projeto de *hardware* e de *software*, tornam esses dispositivos alvos fáceis de atacantes, que se beneficiam comprometendo

não apenas a privacidade de dados sensíveis, mas também com a cooptação desses dispositivos para a construção de *botnets* cada vez maiores (Derakhshan e Ashrafnejad, 2020).

Outro agravante para o cenário de ameaças é o frequente desenvolvimento de novas *botnets* e a criação de variantes. Os desenvolvedores de *malware* utilizam código-fonte de *botnets* existentes para a criação de novos *malwares* customizados (Klaar, 2013). Por exemplo, após a publicação do código-fonte da *botnet* denominada Zeus, foi possível observar uma nova variante que faz uso do código-fonte após apenas 3 meses (Klaar, 2013). Outro exemplo, observado após a publicação do código-fonte da AgoBot em 2003, gerou mais de 580 novas variantes até 2007 (Barford e Yegneswaran, 2007). Isso, por sua vez, possibilita um aumento na diversidade de *botnets* e, consequentemente, dificulta o processo de detecção, pois é necessário reavaliar se as ferramentas existentes são eficazes diante dessas novas ameaças.

Como consequência desses fatores, as *botnets* estão em constante evolução e se desenvolvendo rapidamente, tornando a sua detecção um desafio. Com o advento de novas tecnologias e com o aperfeiçoamento das técnicas de invasão, os *botmasters* estão cada vez mais eficazes em evitar as técnicas de detecção. Gupta e Badve (2017) demonstraram que atacantes conseguem inundar servidores com pacotes similares aos reais, sendo difícil de classificar o fluxo malicioso do real. Chang et al. (2018) verificaram que os atacantes estão intercalando entre enviar pacotes e permanecer temporariamente inativo para dificultar a identificação dos *bots*.

Embora as ameaças cibernéticas estejam em constante evolução, as técnicas de detecção de *botnets* ainda se baseiam em um contexto estático, onde a distribuição dos dados permanece constante ao longo do ciclo de vida da sua aplicação (Cid-Fuentes et al., 2018). Um dos maiores problemas a serem enfrentados por essas soluções é a variedade de técnicas de ataque possíveis (Kedziora et al., 2020). Na base de conhecimento do MITRE ATT&CK[1] foram documentadas 282 técnicas diferentes usadas por um adversário durante um ataque. Uma das consequências dessa diversidade de ataques é a mudança inesperada no comportamento da rede e uma discrepância entre a distribuição de dados observados e a distribuição esperada pela solução empregada. Uma maneira de lidar com essas rápidas mudanças é a implantação simultânea de um conjunto de técnicas para a detecção de *botnets*, específicas para um determinado tipo de ameaça. Porém, o uso dessa estratégia não é viável, pois envolve a construção e gerenciamento manual de um grande conjunto de soluções. Desse modo, este trabalho aborda o problema de como detectar *botnets* de maneira eficaz com o mínimo de intervenção humana, diante de um ambiente caracterizado por mudanças frequentes no contexto de ameaças.

1.2 OBJETIVOS

Este trabalho tem como objetivo geral realizar o gerenciamento e a aplicação de métodos detecção de *botnets* de maneira autônoma. Para esse fim, são definidos dois objetivos específicos: (*i*) a criação e implantação de soluções de alta eficácia para a detecção de *botnets* arbitrárias com o mínimo de intervenção humana e, (*ii*) a realização automática de ajustes de uma solução implantada, visando eliminar a perda de eficácia ao longo do seu ciclo de vida. Para alcançar esses objetivos é proposto o sistema ANTE (do inglês, ***A**uto**N**omous bo**T**net d**E**tection*), um sistema que integra e gerencia o uso de conceitos de aprendizado de máquina distintos para realizar a detecção de *botnets* com um alto grau de automação e precisão.

[1] https://attack.mitre.org Acessado em: 18/10/2021

1.3 CONTRIBUIÇÕES

As principais contribuições desta dissertação incluem um estudo sobre algoritmos de detecção de *botnets* baseados em técnicas de aprendizado de máquina e análise de comportamento de dispositivos. Este estudo é acompanhado de um levantamento de técnicas de automação de aprendizado de máquina, com o objetivo de identificar estratégias eficazes que permitam a detecção de *botnets*, levando em consideração a necessidade de adaptação a uma ampla variedade de ataques.

Tendo como base essas técnicas, foi proposto o sistema ANTE. Em seu funcionamento, o sistema ANTE é capaz de selecionar uma técnica de detecção de *botnets*, junto com o conjunto de hiperparâmetros, mais apropriada para um determinado contexto. O sistema ANTE também considera mudanças no comportamento da rede ao longo da sua execução para realizar a adaptação de soluções empregadas.

Outra contribuição relevante desta dissertação foi a avaliação e análise do sistema proposto. A avaliação foi realizada de forma incremental, onde o sistema foi testado em diferentes etapas para avaliar sua capacidade em lidar com mudanças no ambiente de rede. Primeiramente, foi avaliada a capacidade do sistema em extrair, representar e classificar os comportamentos de dispositivos sem considerar a necessidade de adaptação. Em seguida, foi avaliada a capacidade do sistema em automatizar a construção de um pipeline de classificação, juntamente com um conjunto de hiperparâmetros, para permitir uma rápida adaptação do sistema de detecção às mudanças no ambiente de rede. Por fim, foi avaliada a capacidade do sistema em realizar ajustes automáticos na solução implantada, com o objetivo de eliminar a perda de eficácia ao longo do ciclo de vida. Essa avaliação sistemática permitiu demonstrar a eficácia e impacto dos diferentes módulos do sistema proposto.

1.4 ESTRUTURA DO TEXTO

Este manuscrito está organizado em seis capítulos. O Capítulo 2 apresenta os fundamentos necessários para o entendimento do problema e da solução apresentada. O Capítulo 3 discute os trabalhos relacionados. O Capítulo 4 apresenta o sistema proposto, com uma descrição detalhada dos seus módulos. O Capítulo 5 detalha as avaliações do sistema e discute os resultados obtidos. O Capítulo 6 apresenta as considerações finais, direções futuras e as publicações resultantes deste trabalho obtidas até o momento.

2 FUNDAMENTOS

Este capítulo apresenta os conceitos fundamentais necessários para a compreensão do problema tratado nesta dissertação e do sistema proposto. Assim, a Seção 2.1 apresenta as características, aplicações e arquiteturas de *botnets*. A Seção 2.2 apresenta um conjunto de funções relacionadas à cibersegurança. A Seção 2.3 aborda os conceitos de aprendizado de máquina, detalhando as técnicas utilizadas neste trabalho.

2.1 *BOTNETS*

Uma *botnet*, conforme definido pelo trabalho de Silva et al. (2013), é uma rede de computadores infectados por *malware* chamados de *bots*. Essa definição é ampliada por Bertino e Islam (2017) para incluir detalhes da sua composição, segundo a sua definição uma *botnet* é um conjunto de computadores infectados, controlados por um *botmaster* a partir de uma estrutura de comando e controle (do inglês, *command and control* - C&C). Esta segunda definição ressalta a existência da estrutura de C&C. Um *botmaster*, conforme definido por Klaar (2013), é o indivíduo que controla a *botnet* visando à realização de atividades maliciosas. O *botmaster* utiliza a estrutura de comando e controle para emitir comandos a serem executados pelos *bots*. A estrutura de C&C pode ser implementada de diferentes maneiras com a utilização de protocolos distintos. A maneira como essa implementação é realizada define a arquitetura da *botnet*. Uma visão dos diferentes tipos de arquitetura de *botnets* e do seu modo de funcionamento é apresentada na Subseção 2.1.1.

Dentre as atividades maliciosas executadas por *botnets* é possível destacar a realização de ataques de Negação de Serviço Distribuído (DDoS) pelo seu potencial de danos (He et al., 2017; Jyoti e Behal, 2021). Neste tipo de ataque, uma grande quantidade de solicitações é enviada ao alvo pelos integrantes da *botnet*. Como resultado, o alvo não é capaz de processar a quantidade de tráfego gerada, ficando fora de serviço (Derakhshan e Ashrafnejad, 2020). Outro uso comum é a realização de campanhas de envio de SPAM, o uso de *botnet* permite o envio de milhares de *emails* em algumas horas (Khan et al., 2015).

Nos dois exemplos anteriores as *botnets* são utilizadas para atacar dispositivos externos, que não fazem parte da rede de dispositivos sob o controle do atacante. Porém, o *botmaster* também pode atacar diretamente os dispositivos infectados. Como o *malware* é executado como um aplicativo comum nas máquinas infectadas, esse tem acesso a dados do usuário. Além disso, esse tipo de *malware* é comumente executado com privilégios de administrador, que lhes dão acesso irrestrito a todos os recursos e processos do sistema infectado. Esses privilégios, por exemplo, permitem interceptar pressionamentos de teclas nas máquinas infectadas. Assim, o *botmaster* tem acesso a dados que não estão armazenados no disco, como senhas inseridas em formulários de login (Klaar, 2013).

O poder destrutivo desses ataques foi amplificado nos últimos anos pela existência de *botnets* cada vez maiores. Esse crescimento se dá em parte pela recrutamento de dispositivos IoT. Os fabricantes desses aparelhos focam nas suas funcionalidades e na facilidade de instalação para atrair clientes em detrimento de boas práticas de segurança. Tais aspectos geralmente estão associados a vulnerabilidades no projeto de *hardware* e de *software*, tornando esses dispositivos alvos fáceis para atacantes (Derakhshan e Ashrafnejad, 2020). A Subseção 2.1.1 também apresenta mais detalhes sobre o conceito de dispositivos IoT e exemplos de *botnets* projetadas para infectar esses dispositivos.

2.1.1 Características das *Botnets*

A comunicação de um *botmaster* com a *botnet* é realizada por meio de um canal de C&C. Através do canal de C&C, o *botmaster* é capaz de emitir comandos e receber informações sobre as máquinas infectadas. Essa comunicação geralmente segue a sequência de passos: (i) um *botmaster* deve emitir um comando para os dispositivos que integram a *botnet*; (ii) os *bots* realizam alguma atividade em resposta ao comando; e (iii) o *bot* envia os resultados da execução de suas atividades de volta ao *botmaster* (Fedynyshyn et al., 2011). Na literatura as *botnets* geralmente são classificadas como centralizadas e distribuídas, dependendo da arquitetura do canal de C&C (Raghava et al., 2012).

A arquitetura centralizada é apresentada na Figura 2.1. Nessa figura, é possível observar uma representação de uma rede de dispositivos na área interna ao círculo pontilhado. Essa rede é composta por um servidor da C&C, representado pelo círculo preenchido na cor preta; três *bots*, representados pelos círculos preenchidos na cor cinza; e um dispositivo não infectado, representado pelo círculo na cor branca. Na arquitetura centralizada o *botmaster* se comunica diretamente somente com o servidor de C&C, enviando comandos e recebendo informações das máquinas infectadas. Os *bots* regularmente consultam o servidor central para receber novas ordens e enviar informações. Essa comunicação é representada na figura pelas setas vermelhas pontilhadas. Em *botnets* centralizadas, o canal C&C pode ser implementado usando diferentes protocolos como *Internet Relay Chat* (IRC), *Hypertext Transfer Protocol* (HTTP) e *Email* (Kambourakis et al., 2019).

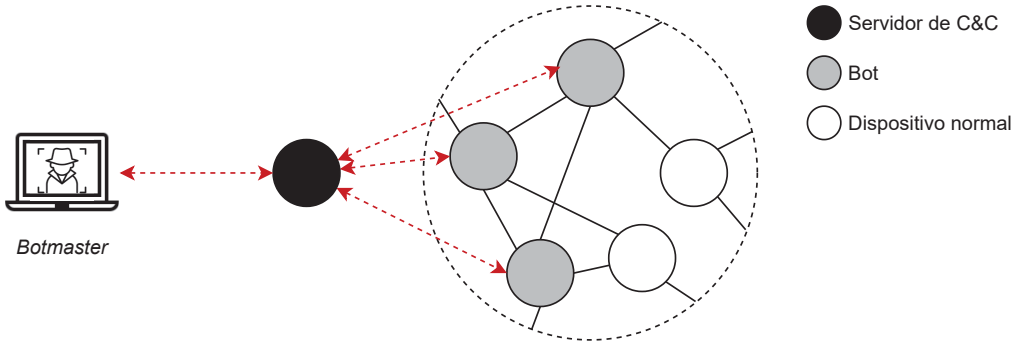


Figura 2.1: Arquitetura de *botnet* centralizada

Um dos maiores problemas da arquitetura centralizada é a existência de um ponto central de falha. Para resolver esse problema, desenvolvedores de *botnets* criaram estruturas de C&C apoiadas por protocolos *Peer-to-Peer* (P2P). Com essa tecnologia, todos os *bots* se comunicam diretamente com outros *bots*. A evolução da tecnologia P2P proporcionou uma nova geração de *botnets* com mais robustez, escalabilidade de resiliência (Klaar, 2013).

A arquitetura descentralizada é ilustrada na Figura 2.2. Comparando essa figura com a Figura 2.1, nota-se que nesta arquitetura não existe um servidor central de C&C. Desse modo, o *botmaster* se comunica diretamente com as máquinas infectadas, indicado na figura pelas setas vermelhas pontilhadas. Em *botnets* descentralizadas seguindo protocolos P2P, cada bot mantém uma lista de vizinhos. Quando um *bot* recebe um comando de um vizinho, esse propaga o comando para outros vizinhos em sua lista, essa comunicação é ilustrada na Figura 2.1 pelas setas vermelhas entre dispositivos infectados. Nesse caso o *botmaster* precisa de acesso a um único dispositivo na rede para obter controle de toda a *botnet*. Se os administrados de rede são capazes de descobrir e isolar um conjunto de *bots*, a comunicação entre os restantes dos *bots* não é interrompida. Porém, existe uma demora maior para propagar mensagens para todos os membros da *botnets* (Raghava et al., 2012).

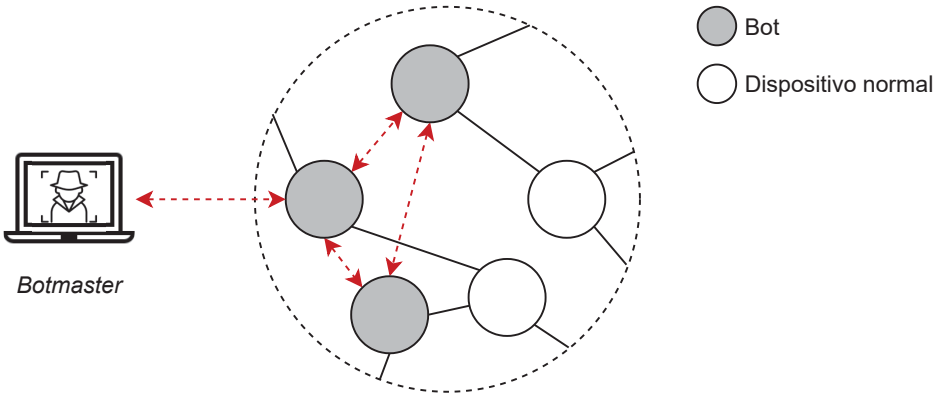


Figura 2.2: Arquitetura de *botnet* descentralizada

2.1.2 *Botnets* e Dispositivos IoT

A Internet das Coisas tem destaque na academia e na indústria, produzindo cada vez mais avanços. Os dispositivos da IoT possuem limitações na largura de banda, baixo poder de processamento e pouca memória. Tais aspectos, agregados a vulnerabilidades no projeto de *hardware* e de *software*, tornam esses dispositivos alvos fáceis de atacantes, que se beneficiam comprometendo não apenas a privacidade de dados sensíveis, mas também com o uso desses dispositivos infectados na construção de ataques maciços e mais danosos. Apesar destas limitações, as estimativas são de crescimento na quantidade de dispositivos conectados à Internet, podendo chegar a um marco de 20,4 bilhões de dispositivos em 2022 (Hassija et al., 2019). Tal crescimento, quando explorado por atacantes, tende a aumentar o potencial danoso das *botnets* (Bertino e Islam, 2017).

O cenário de ameaças relacionado a *botnets* evoluiu tremendamente nos últimos anos, como consequência do desenvolvimento de *botnets* cada vez mais sofisticadas e a exploração do crescimento explosivo de dispositivos IoT. As *botnets* também estão em constante evolução, se tornando cada vez mais especializadas, focando na evasão de determinados mecanismos de proteção de seguranças e técnicas de evasão (Bansal e Mahapatra, 2017).

Um dos exemplos mais proeminentes de ameaças à segurança causadas por dispositivos IoT é a *botnet* Mirai. De acordo com estimativas (Antonakakis et al., 2017), a Mirai infectou quase 65.000 dispositivos IoT nas suas primeiras 20 horas antes de atingir uma população estável entre 200.000 e 300.000 infecções. A Mirai teve como alvo de infecções uma variedade de dispositivos IoT como Gravadores Digitais de Vídeo (do inglês, *Digital Video Recorder* - DVR), Câmeras IP, roteadores e impressoras. Durante a investigação realizada por Antonakakis et al. (2017), a Mirai emitiu 15.194 comandos de ataque DDoS. Esses ataques empregaram uma gama de diferentes estratégias de exaustão de recursos, incluindo a utilização de diferentes protocolos de comunicação. Além de ser uma das *botnets* com o maior número de dispositivos infectados já registrados, a Mirai também é marcada pelo seu número de variantes. Após a divulgação do código-fonte da Mirai em agosto de 2016, foi possível observar uma proliferação de milhares de *botnets* variantes (Kambourakis et al., 2017)

2.2 ARCABOUÇO DE CIBERSEGURANÇA DO NIST

O ciberespaço, conforme apresentado no trabalho de Monteiro (2007), é o espaço criado pela conexão entre as infraestruturas de comunicação e as pessoas. Dentro deste espaço é possível observar a ocorrência de eventos que ameaçam o seu funcionamento, sejam esses intencionais, acidentais ou naturais. Tendo em vista a diversidade desses eventos e a sua constante ocorrência,

fazem-se necessárias a organização e a coleta de recursos, processos e estruturas utilizadas para a proteção do ciberespaço. Para esse fim, o *National Institute of Standards and Technology* (NIST) criou o *Cybersecurity Framework* (CSF). O CSF define uma metodologia baseada em padrões, diretrizes e práticas existentes para avaliar e gerenciar riscos de segurança cibernética.

O CSF foi originalmente desenvolvido pelo NIST em resposta a ordem executiva 13636, que pedia a melhora da segurança cibernética da infraestrutura crítica dos Estados Unidos. Este trabalho foi formalizado a partir da aprovação da Lei de Aprimoramento da Segurança Cibernética (do inglês, *Cybersecurity Enhancement Act* - CEA) de 2014. Nesta lei foi estabelecido que o NIST deve desenvolver um arcabouço de segurança flexível, repetível, eficaz com uma boa relação de custo-benefício. Esta lei então contribuiu para a continuação do desenvolvimento do CSF e forneceu direções futuras (NIST, 2018).

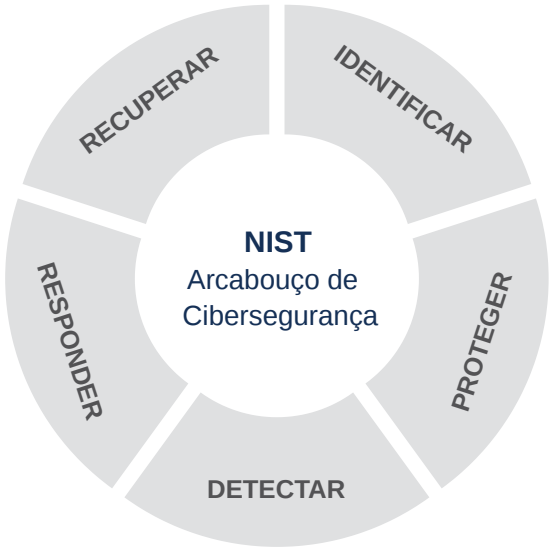


Figura 2.3: Arcabouço de cibersegurança do NIST. Adaptado de NIST (2018)

O CSF é organizado em cinco funções: identificar, proteger, detectar, responder, recuperar, conforme ilustrado na Figura 2.3. Nesta figura, as funções são organizadas em um círculo, evidenciando que as funções não formam um caminho sequencial ou levam a um estado final estático. Em vez disso, as funções são realizadas simultaneamente e continuamente formando uma cultura operacional capaz de lidar com a dinamicidade dos riscos de segurança cibernética (NIST, 2018). As funções são detalhadas a seguir.

- Identificar - desenvolver um entendimento organizacional para gerenciar riscos de segurança cibernética para sistemas, pessoas, ativos, dados e capacidades. Essa função tem como objetivo promover a compreensão do contexto de negócios, os recursos que suportam as funções críticas da organização e os riscos de segurança cibernética relacionados a esses aspectos, permitindo que a organização se concentre e priorize seus esforços de acordo com sua estratégia de gestão de riscos e necessidades de negócios.

- Proteger - desenvolver e implementar salvaguardas adequadas para garantir a entrega de serviços. Com isso, a função de proteção visa limitar ou conter o impacto de um potencial evento de segurança cibernética.

- Detectar - desenvolver e implementar atividades para identificar a ocorrência de um evento de segurança cibernética. A função de detectar permite a descoberta e

compreensão de eventos que comprometem a política de segurança cibernética de uma organização.

- Responder - desenvolver e implementar atividades que serão executadas para conter o impacto de um incidente de segurança cibernética após a sua ocorrência.
- Recuperar - desenvolver e implementar atividades para restaurar quaisquer recursos ou serviços que foram prejudicados devido a um incidente de cibersegurança.

2.3 APRENDIZAGEM DE MÁQUINA

Esta seção apresenta os conceitos de aprendizado de máquina fundamentais para este trabalho. Na subseção 2.3.1, é apresentada uma visão geral da área. A Subseção 2.3.2 detalha o funcionamento um algoritmo de aprendizado de máquina. A Subseção 2.3.4 apresenta uma maneira de escolher automaticamente uma técnica de classificação eficaz para uma determinada tarefa.

2.3.1 Visão geral

A Inteligência Artificial (IA) é um campo da computação que estuda métodos e procedimentos para a construção de sistemas que operam de maneira autônoma. Esses agentes, na maior parte, são sistemas capazes de persistir por um período de tempo prolongado, adaptar a mudanças e criar e perseguir metas (Russell e Norvig, 2020). Na Figura 2.4 são ilustrados os subcampos da IA utilizados nessa trabalho. Diretamente dentro da IA é possível observar o subcampo de aprendizado de máquina. O aprendizado de máquina visa à construção de programas a partir de uma coleção de pares de entrada e saída, com objetivo de gerar uma função que prevê a saída para entradas não observadas durante a construção do programa (Russell e Norvig, 2020).

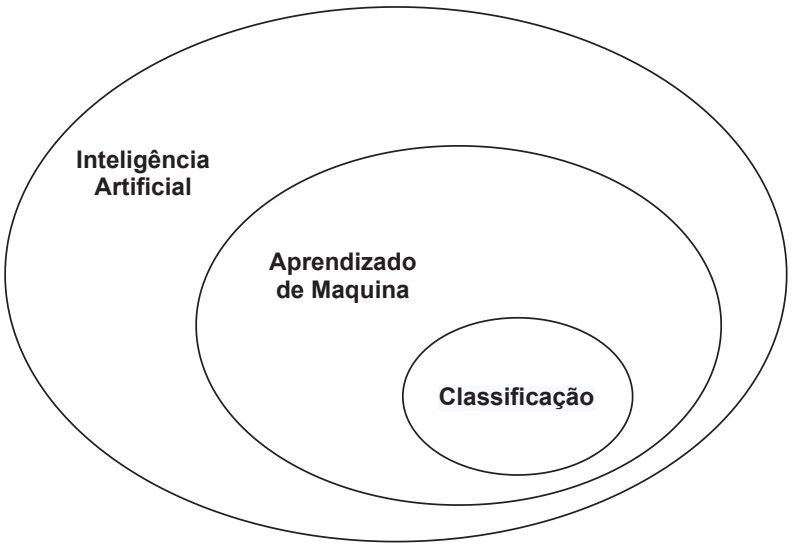


Figura 2.4: Visão geral de técnicas de inteligência artificial

O aprendizado de máquina é definido de maneira formal no trabalho de Tom Mitchell (Mitchell, 1997), pesquisador da área de aprendizado de máquina na *Carnegie Mellon University*. Segundo a sua definição, um programa de computador é capaz de aprender a realizar uma tarefa T a partir de uma experiência E, se o desempenho ao realizar a tarefa T, medido por

uma métrica de avaliação P, melhora com a experiência E. Desse modo, existem diferentes tipos de técnicas de aprendizado de máquina, dependendo da tarefa T e da métrica P usada para avaliar o desempenho.

No trabalho de Murphy (2022), a tarefa T é descrita como a descoberta de um mapeamento f de dados $\mathbf{x} \in \mathcal{X}$ para uma saída $\mathbf{y} \in \mathcal{Y}$. Os dados de entrada, representados por $\mathbf{x}$, são chamados de características ou atributos. Esse valor geralmente é um vetor de números, que representa características de objetos relevantes à tarefa. A saída, representada por $\mathbf{y}$, é uma variável de interesse que deve ser obtida a partir dos atributos. Um exemplo de combinação de entrada e saída seria utilizar a temperatura e umidade registradas em um determinado dia para prever a temperatura do dia seguinte. Esse tipo de tarefa onde a saída é uma variável contínua é chamada de regressão. Outro exemplo seria a detecção de animais em imagens, o conjunto de atributos $\mathbf{x}$ é composto pelos *pixels* da imagem e a saída é um numero que representa a presença de um determinado animal. Nesse caso, como a saída é um conjunto não ordenado de números, a tarefa é chamada de classificação. Quando uma tarefa de classificação tem como saída possível somente dois valores, ela é denominada de classificação binária. Por exemplo, na detecção de *bots*, os valores de $\mathbf{y}$ codificam a categoria dos dispositivos, 1 indica que o dispositivo é um *bot*, enquanto 0 indica o contrário.

2.3.2 Funcionamento de um Algoritmo de Aprendizagem de Máquina

Conforme descrito na Subseção 2.3.1, um algoritmo de aprendizado de máquina é responsável por descobrir um mapeamento f de dados $\mathbf{x} \in \mathcal{X}$ para uma saída $\mathbf{y} \in \mathcal{Y}$. Geralmente, a execução desses algoritmos é dividida em duas fases. Na primeira fase, a combinação de $\mathcal{X}$ e $\mathcal{Y}$ é usada para construir o mapeamento f. Após construído, o mapeamento é utilizado para obter valores de $\mathcal{Y}$ para novas entradas. Nesse caso, como $\mathcal{X}$ e $\mathcal{Y}$ são utilizados para guiar o processo de construção do mapeamento, o algoritmo é denominado de supervisionado. De outro modo, se somente $\mathcal{X}$ for utilizado, ele é denominado de não supervisionado.

Existe uma grande diversidade de algoritmos de aprendizado de máquina, porém é possível encontrar pontos em comum na maioria desses algoritmos. No trabalho de Kedziora et al. (2020), esses pontos são representados como um conjunto de componentes que formam um sistema integrado de transformação de dados. Como ilustra a Figura 2.5, os componentes interagem entre si para realizar a transformação dos dados de entrada em uma saída. Esse sistema atua em dois modos de operação: treinamento e predição. No modo de treinamento os componentes interagem entre si para construir o mapeamento f. Essas interações são detalhadas com a seguinte sequência de passos:

1. O sistema é inicializado por sistemas externos com um conjunto de parâmetros iniciais, e um modo de operação. O modo de operação indica se o sistema está sendo executado em modo de treinamento ou predição. Para essa sequência de passos o sistema opera em modo de treinamento.

2. Os dados de treinamento são inseridos *buffer* de entrada por um sistema externo. Se o sistema implementa um técnica de aprendizado supervisionado, cada instância pertencente a esses dados é composto de um vetor de características x e uma classe y. Caso contrário, somente são inseridos os dados de x.

3. A interface do componente instrui o controlador a inicializar o treinamento, como consequência, o controlador inicia o processo de otimização. O processo de otimização visa aprender o mapeamento f.

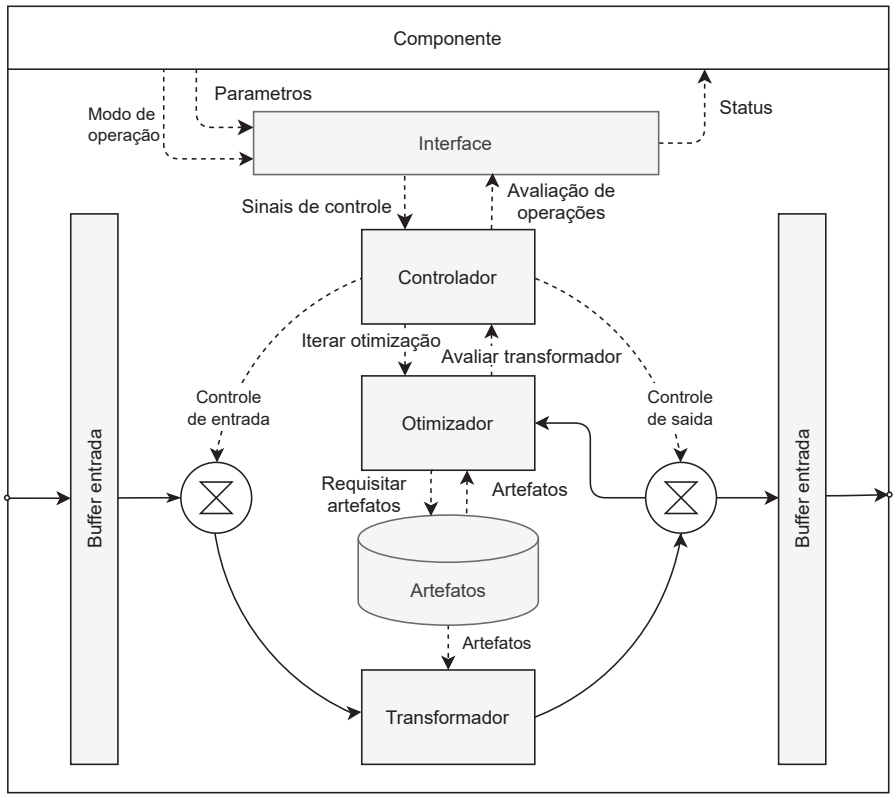


Figura 2.5: Componente básico de aprendizagem de tarefas

4. O controlador direciona os dados de entrada para o transformador. O transformador implementa o mapeamento f, utilizado para realizar a transformação dos dados de entrada. Os parâmetros associados com o mapeamento são salvos na base de artefatos.

5. O controlador direciona os transformados para o otimizador. O otimizador compara os dados transformados com o valor de y. A diferença entre os valores é usada para atualizar a base de artefatos, com o objetivo de aproximar os dados transformados com o valor de y.

6. O controlador utiliza a avaliação do otimizador para determinar se o processo de otimização deve continuar, ou se o valor produzido pelo mapeamento f já é satisfatório. Se o valor não for satisfatório o processo é repetido novamente a partir do passo 4. Porém, dessa vez são utilizados os parâmetros atualizados na base de artefatos.

7. Quando o controlador termina o processo de otimização, os sistemas externos são notificados a partir da interface.

No segundo modo de operação, a predição, o sistema é utilizado para realizar a transformação de novas entradas. Nesta etapa a construção do mapeamento já foi realizada. Desse modo, o controlador somente direciona os dados de entrada para o transformador. Em seguida os dados transformados são direcionados para a saída do sistema.

2.3.3 Mudanças de Conceito

A implantação de um algoritmo de aprendizado de máquina em um determinado ambiente depende muitas vezes da suposição que este ambiente seja estático, ou seja, a distribuição dos

dados que serão utilizados como entrada para o algoritmo permanece a mesma por um período de tempo indefinido. Embora essa suposição possa ser válida em muitas aplicações, ela pode, no entanto, ser irreal em alguns cenários. Nesses cenários, quando ocorre uma mudança na distribuição dos dados, a entrada do algoritmo passa a ser inconsistente com os exemplos utilizados durante o treinamento, sendo necessário realizar uma atualização do algoritmo implantado. Esta alteração na distribuição dos dados é conhecida como mudança de conceito (Tsymbal, 2004; Almeida et al., 2018).

Um exemplo de problema de mudança de conceito pode ser observado na detecção de rostos de usuários por aplicativos de redes sociais. Nesse cenário, o rosto dos usuários mudam constantemente devido a fatores como o envelhecimento, uso de maquiagem, crescimento da barba, diferentes cortes de cabelo, etc. Este cenário pode, portanto, sofrer uma mudança de conceito, uma vez que as imagens coletadas do usuário no passado podem não ser adequadas para reconhecê-lo no presente (Almeida et al., 2018). Outro exemplo são os padrões de preferências de compra de clientes que podem mudar com o dia da semana atual, disponibilidade de alternativas, taxa de inflação, etc (Tsymbal, 2004).

As mudanças de conceito podem se manifestar de diferentes formas ao longo do tempo, conforme ilustrado na Figura 2.6. Nesta figura a mudança na distribuição dos dados ocorre como uma alteração na média dos valores observados. A Figura 2.6(a) ilustra uma mudança abrupta na média dos valores, resultando em uma transição imediata entre conceitos. Um exemplo desse tipo de transição é a variação no comportamento de consumo de um cliente após um aumento de salário. A Figura 2.6(b) apresenta uma mudança incremental na distribuição dos dados, resultando na existência de conceitos intermediários. Essa mudança pode ser causada por um sensor que se desgasta ao longo da sua vida útil, produzindo saídas menos precisas. A Figura 2.6(c) ilustra uma mudança de conceito gradual, nesse cenário existe uma oscilação entre o conceito antigo e o conceito novo. A Figura 2.6(d) também apresenta um cenário onde ocorre uma oscilação entre os conceitos, porém nesse caso os conceitos permanecem por um período maior (Gama et al., 2014).

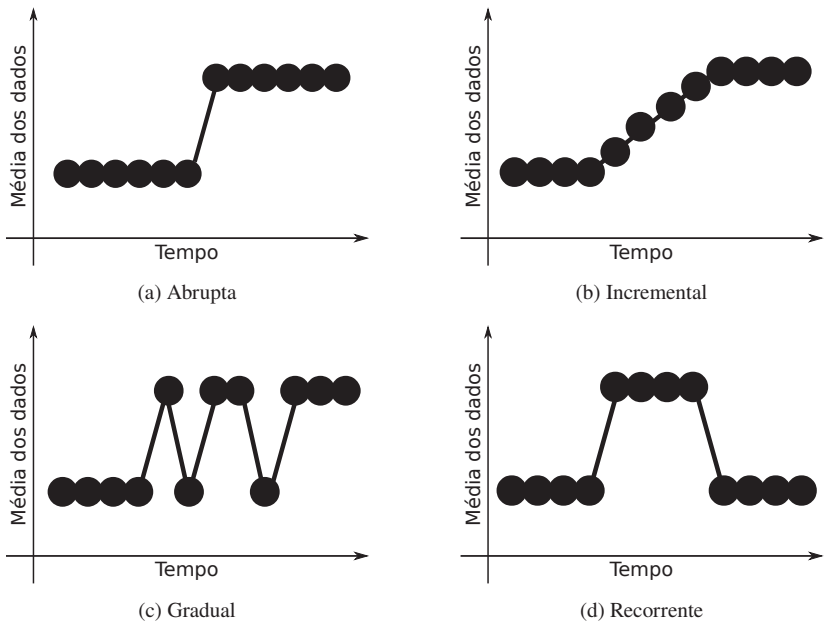


Figura 2.6: Tipos de mudança de conceito. Adaptado de Gama et al. (2014)

De acordo com o trabalho de Lu et al. (2018) as mudanças de conceitos podem ser detectadas com duas classes de métodos. A primeira classe inclui métodos que monitoram

as taxas de erro de um algoritmo de aprendizado de máquina. Se os valores das métricas de desempenho se distanciam dos valores observados durante o treinamento, os dados que estão sendo observados pelo algoritmo não são consistentes com os exemplos de treinamento. A segunda classe é composta por métodos que usam medidas de distância para estimar a similaridade entre as distribuições de dados anteriores e atuais.

2.3.4 AutoML

Os algoritmos de aprendizagem de máquina automatizada (do inglês, Automated Machine Learning - AutoML) têm como objetivo automatizar a escolha de um algoritmo de aprendizado de máquina e o seus parâmetros para uma determinada tarefa. O AutoML pode ser entendido como a busca por um conjunto de parâmetros para a minimização de uma função objetivo L, essa função é usada para avaliar a eficácia de um algoritmo de aprendizagem A, que é aplicado a um conjunto de dados $\mathcal{X}$ e $\mathcal{Y}$, separados em dados de treinamento e validação de D_{treino} e $D_{\text{validação}}$. O algoritmo A é dependente de um conjunto de hiper-parâmetros pertencentes a um espaço de busca, $\lambda \in \Lambda$. Com essa anotação o objetivo do AutoML pode ser descrito pela Equação 2.1.

$$\lambda^* \in \underset{\lambda \in \Lambda}{\operatorname{argmin}} L\left(A_\lambda, D^{(i)}_{\text{treino}}, D^{(i)}_{\text{validação}}\right) \tag{2.1}$$

Conforme ilustra a Figura 2.7, neste trabalho um algoritmo de AutoML é representado como um sistema integrado de componentes. Desse modo, este assume o papel do sistema externo mencionado na Subseção 2.3.2. Em vista disso, o AutoML é responsável pelo gerenciamento da etapa de treinamento de um algoritmo de aprendizado de máquina. Isso inclui fornecimento de parâmetros iniciais e dados de entrada. Durante o seu ciclo de vida o AutoML é responsável por treinar e avaliar uma coleção de algoritmos de aprendizado de máquina, visando encontrar o algoritmo com a melhor eficácia entre eles, avaliado de acordo com a função objetivo. Esse processo de busca é detalhado na sequência de passos a seguir:

1. O sistema é instanciado com um conjunto de restrições definidas por sistemas externos. Essas restrições incluem tempo de busca, algoritmos e métricas para a avaliação dos algoritmos.

2. O controle de operações utiliza as restrições para formar o espaço de busca. Para a construção do espaço de busca é utilizado uma base de dados de experiências para encontrar soluções anteriores que podem ser exploradas.

3. O controle de busca percorre o espaço de busca definido pelo controle de operação, priorizando combinações de parâmetros mais prováveis de obter um bom desempenho de acordo com as métricas de avaliação.

4. Cada combinação de parâmetros escolhida pelo controle é instanciada pelo construtor de pipeline.

5. Depois de instanciado, o modelo de aprendizado de máquina é avaliado conforme as métricas fornecida como parâmetro.

6. Por fim, o controle de operações define qual combinação de parâmetros será escolhida como saída. Antes de finalizar o fim das operações, o controle de operação atualiza a base de dados de experiências, adicionando uma descrição dos dados observados e a melhor solução encontrada.

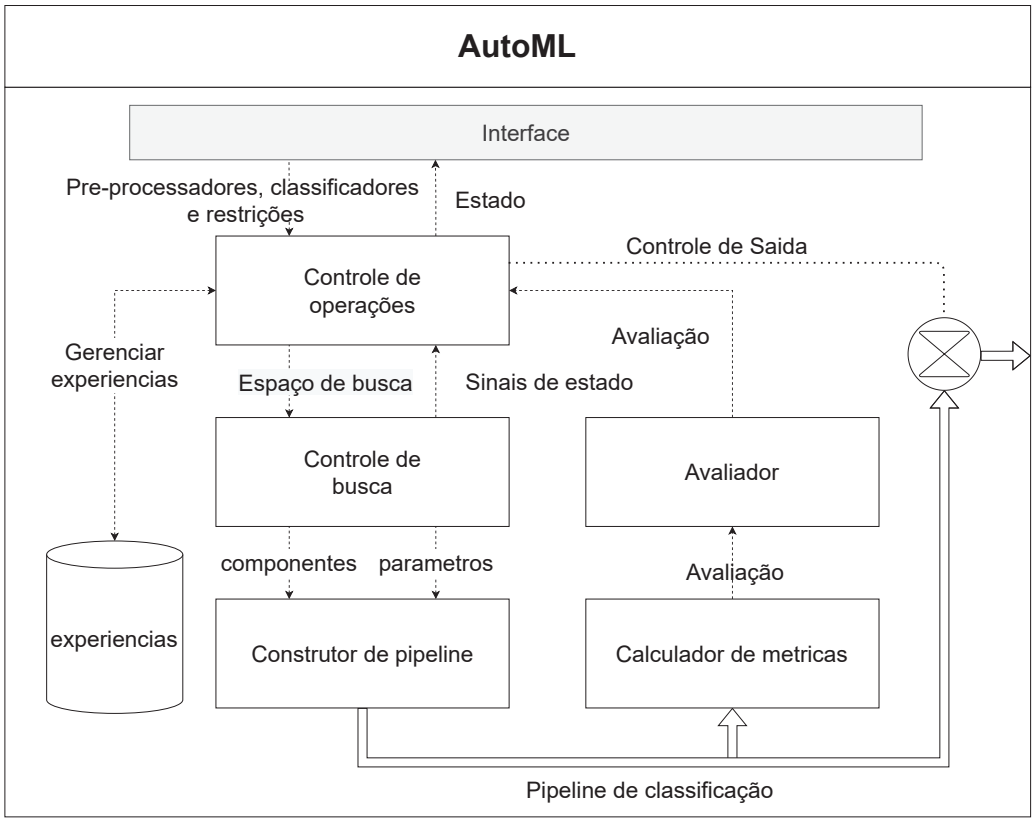


Figura 2.7: Representação de um algoritmo genérico de AutoML

2.4 RESUMO

Este capítulo apresentou as diferentes características de *botnets*, como a sua arquitetura e tipos de ataques. Além disso, foram apresentadas as *botnets* voltadas para dispositivos IoT. Em seguida, foram apresentadas as funções de que compõem o arcabouço de cibersegurança do NIST. Por fim, foi introduzida a área de aprendizado de máquina. Os princípios de aprendizado de máquina apresentados neste capítulo servem como base para o sistema proposto no Capítulo 4, e na avaliação do sistema realizada no Capítulo 5.

3 TRABALHOS RELACIONADOS

Este capítulo revisa a literatura acerca de métodos de detecção de *botnets*. A Seção 3.1 traz uma visão geral sobre o problema de detecção de *botnets*. A Seção 3.2 detalha técnicas de detecção de *botnets* que utilizam aprendizado de máquina, destacando a diversidade de características exibidas pelas técnicas existentes. Por fim, a Seção 3.3 apresenta trabalhos sobre a automatização de aprendizado de máquina.

3.1 DETECÇÃO DE *BOTNETS*

A detecção de uma *botnet* consiste na realização de atividades projetadas para a identificação de uma ou todas as suas partes, sendo estas o *botmaster* (Mizoguchi et al., 2011; Lin e Lee, 2012; Ramsbrock et al., 2008), a estrutura de C&C (Kondo e Sato, 2007; Gu et al., 2009; Fedynyshyn et al., 2011) ou os dispositivos infectados (Khattak et al., 2014). Embora exista uma diferença entre essas atividades, na literatura é comum se referir a identificação dos dispositivos infectados que compõem uma *botnet* como "detecção de *botnet*". Tendo em vista a adoção generalizada desta interpretação, este trabalho segue os exemplos observados na literatura e também adota essa nomenclatura.

Vários estudos sugerem taxonomias diferentes para técnicas de detecção de *botnets* (Khattak et al., 2014; Silva et al., 2013; Shinan et al., 2021; Amini et al., 2015; Zeidanloo et al., 2010). A Figura 3.1 sumariza a taxonomia definida por Silva et al. (2013); Shinan et al. (2021), estes trabalhos apresentam métodos de detecção de *botnets* com base em aprendizado máquina como um conjunto distinto de trabalhos dentro da literatura. Como esta dissertação está localizada dentro deste grupo, essa foi a taxonomia adotada. Conforme observado na figura, nesta taxonomia os métodos são divididos em duas categorias principais: *honeynets* e sistemas de detecção de intrusão.

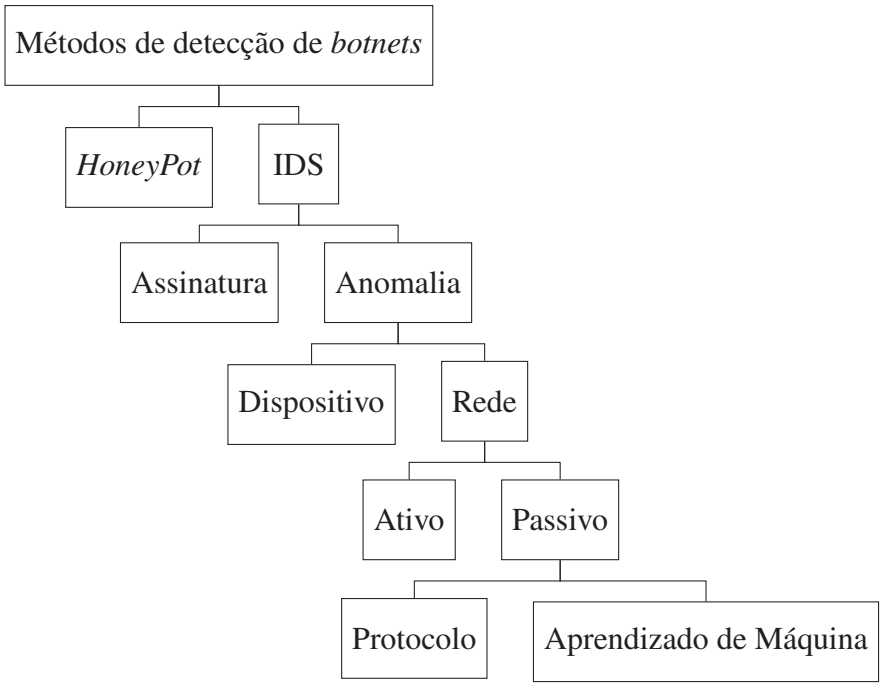


Figura 3.1: Taxonomia de métodos de detecção de *botnets*. Adaptado de Silva et al. (2013); Shinan et al. (2021)

Uma *honeynet* é uma rede de dispositivos intencionalmente configurados com vulnerabilidades criada como uma armadilha visando atrair a atenção de invasores. Esses dispositivos existem em ambientes controlados e seguros, eliminando a possibilidade de danos causados por infecções (Zeidanloo et al., 2010). Todas as atividades realizadas pelos dispositivos na *honeynet* são conhecidas antecipadamente, portanto qualquer tráfego fora do esperado é considerado como uma atividade suspeita que deve ser investigada (Zeidanloo et al., 2010). Embora *honeynets* sejam úteis para adquirir informações detalhadas sobre o processo de infecção, e a obtenção de binários, elas só detectam *bots* que interagem diretamente com os dispositivos que são usados como armadilha (Khattak et al., 2014; Silva et al., 2013).

De acordo com Bace e Mell (2001), a identificação de intrusões é o processo de monitoramento de eventos que ocorrem em um computador ou rede de computadores, visando a detecção de possíveis incidentes. Estes incidentes são definidos como violações ou ameaças iminentes de violação de políticas de segurança, ou seja, eventos que são considerados prejudiciais ao funcionamento de um computador, ou rede de computadores. Nesse sentido, um Sistema de Detecção de Intrusão (do inglês, *Intrusion Detection System* - IDS) é o sistema de *software* ou o *hardware* responsável por automatizar o processo de detecção de intrusão. A detecção de *botnets* com o uso de um IDS é subdividida em duas classes de métodos, sendo estes baseados em assinatura ou anomalias (Silva et al., 2013).

Na detecção baseada em assinatura, os especialistas de segurança analisam dispositivos com o objetivo de extrair padrões de comportamento associados com a atividade de *botnets* (Silva et al., 2013; Gu et al., 2007). Ao encontrar um novo padrão, esse é inserido em uma base de conhecimento, onde é utilizado posteriormente por um IDS, sendo este responsável por identificar a existência do padrão definido (Gu et al., 2007; Roesch et al., 1999; Paxson, 1999). Embora a análise de assinatura seja comumente usada, *botnets* mais sofisticadas podem eludir essas técnicas com pequenas alterações no seu modo de operação (Vuong e Alam, 2011). Outra limitação deste tipo de análise é a necessidade de um conhecimento profundo da *botnet* observada, o que significa que a efetividade da análise depende de atualizações frequentes à base de assinaturas e compartilhamento de informações entre organizações (Silva et al., 2013).

A detecção baseada em anomalias é uma classe de métodos de detecção dedicados a identificar padrões incomuns no comportamento dos dispositivos associados à operação de *botnets* (Stevanovic e Pedersen, 2016). Essas técnicas visam detectar *bots* com base em atividades anormais, como tráfego inesperadamente alto, latência alta e atividades em portas incomuns (Miller e Busby-Earle, 2016). Em contraste com a detecção baseada em assinatura, a detecção baseada em anomalia é geralmente capaz de detectar novas formas de atividades maliciosas que exibem características anômalas similares a outras *botnets* existentes (Nooribakhsh e Mollamotalebi, 2020). No entanto, um dos principais desafios do uso da detecção baseada em anomalias é a existência de falsos positivos, onde comportamentos fora do comum podem não necessariamente estar associados à atividade de *botnets* (Radain et al., 2021; Somani et al., 2017; Srinivasan et al., 2019). No trabalho de Silva et al. (2013); Shinan et al. (2021) as técnicas de detecção de *botnets* baseados em anomalias são subdivididas em nível de *host* e rede.

Na detecção em nível de *host*, os processos internos dos dispositivos são monitorados à procura de qualquer comportamento anômalo, incluindo sobrecargas de processamento, acesso a arquivos questionáveis e comunicação com dispositivos remotos não confiáveis (Zeidanloo et al., 2010; Silva et al., 2013). Através da análise de registros de aplicações e chamadas do sistema é possível capturar informações detalhadas sobre a infecção do dispositivo pelo *malware* de *botnet* e também todas as ações realizadas por um atacante. Porém, o monitoramento de todas as atividades do sistema resulta em um consumo alto de recursos do sistema (Shinan et al.,

2021). Além disso, em alguns casos não é possível instalar o *software* de monitoramento nos dispositivos que devem ser monitorados (Silva et al., 2013).

Na detecção em nível de rede, são coletadas informações através do monitoramento da rede de dispositivos, como pacotes e estatísticas sobre os fluxos de pacotes com o objetivo de descobrir padrões no tráfego de rede que estão fora do comportamento esperado (Silva et al., 2013). A análise dessas informações pode considerar diferentes dimensões, como o uso de portas ou o número e frequência de transmissão de pacotes (Asha et al., 2016; Santos et al., 2017; Moreira et al., 2021; Pelloso et al., 2018). Essa análise também pode investigar o conteúdo dos pacotes (Lu et al., 2009). As técnicas de detecção em nível de rede podem ser classificadas em duas categorias: ativas e passivas (Silva et al., 2013; Shinan et al., 2021). No monitoramento ativo, o mecanismo de detecção participa das atividades da rede, injetando pacotes com objetivo de obter mais informações sobre os dispositivos infectados. No entanto, a interação direta com os *bots* pode alertar o *botmaster*, resultando na aplicação de contramedidas que dificultam a detecção (Silva et al., 2013). De maneira oposta, na detecção passiva o mecanismo de detecção não interage com os *bots*, realizando somente a observação do seu comportamento. Isso permite que a detecção seja feita de maneira offline, a partir da análise de dados históricos, como capturas de pacotes (Shinan et al., 2021).

No trabalho de Shinan et al. (2021), os métodos de detecção passivos são classificados em técnicas baseadas em protocolos e aprendizado de máquina. Os métodos baseados em protocolos encontram padrões associados com diferentes *botnets* que utilizam o mesmo protocolo de comunicação (Goebel e Holz, 2007; Villamarin-Salomon e Brustoloni, 2008; Zhang et al., 2011).

Este trabalho se encontra dentro dos métodos de detecção de *botnets* baseado em aprendizado de máquina. Esses métodos se mostraram promissores na literatura (Stevanovic e Pedersen, 2016; Nooribakhsh e Mollamotalebi, 2020). Esta área é caracterizada pelo reconhecimento automatizado de padrões de tráfego relacionados a *bot* e a capacidade de reconhecer padrões de tráfego malicioso sem conhecimento prévio de suas características, inferindo esse conhecimento a partir de outros padrões conhecidos (Miller e Busby-Earle, 2016; Stevanovic e Pedersen, 2016; Vu et al., 2021).

3.2 DETECÇÃO DE *BOTNETS* COM APRENDIZADO DE MÁQUINA

A literatura apresenta uma diversidade de trabalhos que utilizam técnicas de aprendizado de máquina para a detecção de *botnets* (Shinan et al., 2021; Vu et al., 2021). Na Tabela 3.2 são comparados métodos que empregam princípios de análise que visam características diferentes de uma *botnet*. Além disso, esses métodos são avaliados usando metodologias de avaliação e dados diferentes. Na tabela, o protocolo corresponde ao protocolo de comunicação utilizado pela *botnet* avaliada. A coluna nomeada IoT indica se os dados usados para a avaliação são provenientes de dispositivos IoT. Essa distinção entre tipos de dispositivos é empregada com base na classificação de Koroniotis et al. (2019). Na qual, essa diferenciação é justificada em razão da diferente quantidade e velocidade de dados gerados por esses dispositivos, bem como a sua diversidade. Essa diferença acarreta no desenvolvimento de novos métodos de detecção projetados especificamente para levar em consideração esses aspectos. Por fim, a coluna nomeada solução proposta apresenta a técnica de aprendizado de máquina proposta pelo trabalho.

Geralmente as técnicas de detecção de *botnet* baseadas em aprendizado de máquina fazem a suposição que a distribuição de dados observada durante o treinamento é a mesma durante a implantação. Wang et al. (2017) apresenta uma abordagem para a detecção de *botnets* que considera possíveis mudanças nas distribuições de dados. No método proposto, primeiro é

realizada a identificação de mudanças de conceito. Para esse fim, é feita uma comparação da distância entre a distribuição de dados observadas durante a última regulagem do algoritmo e os dados sendo observados. Se a distância excede um determinado limiar, significa que houve uma mudança de conceito. Nesse caso, são avaliadas as características de tráfego mais relevantes para identificar os *bots* no novo conceito. Em seguida, o algoritmo de classificação é ajustado para dar uma importância maior a essas características.

Schwengber et al. (2020) identifica a mudança de conceito com a utilização do *Discriminative Drift Detector* (D3) (Gözüaçık et al., 2019). Neste algoritmo, um classificador não supervisionado é utilizado para fazer a diferenciação entre dados de treinamento e os dados observados durante a implantação. Nesse caso, se o algoritmo é capaz de fazer a diferenciação significa que os dois conjuntos de dados são diferentes. Implicando que os dados de treinamento não são mais condizentes com os dados sendo observados. Após a identificação de uma mudança de conceito os dados presentes no novo conceito são utilizados para fazer o retreinamento do classificador utilizado para identificar os *bots*.

Trabalho	IoT	Protocolo	Solução proposta
Stevanovic e Pedersen (2015)	Não	TCP, UDP e DNS	Floresta aleatória
Alazzam et al. (2019)	Sim	TCP e UDP	Floresta aleatória
Tuan et al. (2020)	Não	TCP, UDP e ICMP	K-means
Indre e Lemnaru (2016)	Não	TCP e UDP	*Passive Aggressive*
Heydari et al. (2017)	Não	TCP, UDP, DNS e HTTP	Máquina de vetores de suporte
Kozik e Choraś (2017)	Não	TCP e UDP	Floresta aleatória
Rahal et al. (2020)	Não	TCP, UDP, DNS e ICMP	*Causal Graph Process*

Tabela 3.1: Comparação de trabalhos de detecção de *botnets* que utilizam aprendizado de máquina.

Com base nos trabalhos revisados, os métodos de detecção são avaliados a partir de cenários estáticos que podem não refletir a mudança entre diversos tipos de ataques encontrados em cenários reais. Além disso, a maior parte dos trabalhos ignora a necessidade de manutenção de uma técnica de aprendizado de máquina após a sua implantação. Sendo poucos os trabalhos que consideram o problema de mudança de conceito. Dentre esses, é possível encontrar métodos que realizam ajustes no algoritmo de detecção. Porém, é fundamental investigar se somente reajustar o algoritmo é suficiente, ou se em alguns casos é necessário a troca completa do algoritmo. Também é indispensável investigar como essa troca pode ser realizada de maneira eficiente.

3.3 AUTOMATIZAÇÃO DE APRENDIZADO DE MÁQUINA

Conforme observado na Seção 3.2, nos últimos anos houve um número significativo de publicações sobre métodos de detecção de *botnets* que utilizam aprendizado de máquina. Embora esses trabalhos ofereçam muitos benefícios, a implantação bem-sucedida destes métodos requer um grande esforço de especialistas humanos (Arnaldo e Veeramachaneni, 2019; Arp et al., 2020). Antes de tudo, é necessário realizar a escolha de um método a partir deste conjunto diverso de possíveis soluções, levando em consideração informações do contexto em que o método foi desenvolvido, e do ambiente onde ele será implantado. Em seguida, é necessário investir uma quantidade considerável de tempo para realizar o ajuste dos parâmetros da solução escolhida

para que ela obtenha um bom desempenho (Feurer e Hutter, 2019). A automatização destes passos permitiria a validação e implantação de métodos baseados em aprendizado de máquina de maneira reprodutível e confiável (Claesen e Moor, 2015; Feurer e Hutter, 2019). Com esse objetivo surgiu o AutoML.

3.3.1 Seleção de Algoritmos e Otimização de Hiperparâmetros

Antes do surgimento do AutoML, o problema de seleção de algoritmo foi definido no trabalho de Rice (1976). Em termos gerais, este trabalho propôs um modelo abstrato que, dado um conjunto de problemas e um conjunto de algoritmos, pudesse identificar um algoritmo ótimo para a resolução de cada problema de acordo com alguma métrica de desempenho. Estudos mais recentes sobre o problema de seleção de algoritmo demonstraram a viabilidade de usar aprendizado de máquina para essa tarefa (Bischl et al., 2016). Uma vez que, o desempenho destas soluções pode chegar próximo ao de um algoritmo perfeito que sempre faz a melhor escolha (Bischl et al., 2016).

No contexto de aprendizado de máquina, uma versão limitada do problema de seleção de algoritmo vem ganhando atenção, a otimização de hiperparâmetros (do inglês, *Hyperparameter Optimisation* (HPO) (Kedziora et al., 2020; Yang e Shami, 2020). Uma característica dos métodos de aprendizado de máquina é que eles são parametrizados por um conjunto de hiperparâmetros que devem ser definidos para maximizar o seu desempenho (Claesen e Moor, 2015). Os hiperparâmetros são usados para configurar vários aspectos dos algoritmos de aprendizagem, sendo assim a sua escolha tem um impacto profundo sobre o desempenho destes algoritmos (Claesen e Moor, 2015; Feurer e Hutter, 2019). Por exemplo, o trabalho de Bagnall e Cawley (2017) demonstrou que a partir de configurações no conjunto de hiperparâmetros um algoritmo pode atingir um desempenho comparado ao estado da arte, ou pode ser considerado ineficaz. Métodos de HPO visam reduzir o esforço humano necessário para ajustar hiperparâmetros através de uma busca automática de possíveis configurações (Claesen e Moor, 2015; Luo, 2016; Feurer e Hutter, 2019). Além disso, estes métodos são capazes de melhoras significativas no desempenho de algoritmos de aprendizado de máquina, dado que estes se provaram superiores à especialistas humanos na configuração de algoritmos (Komer et al., 2019).

O problema de HPO foi estendido em Thornton et al. (2013) para incluir a seleção de algoritmos de aprendizado de máquina. Esta nova versão do problema foi chamada de seleção conjunta de algoritmo e hiperparâmetros (do inglês, *Combined Algorithm Selection and Hyperparameter Optimization* - CASH). Métodos que resolvem este problema visam escolher simultaneamente um algoritmo de aprendizado e uma configuração de hiperparâmetros que otimizam o desempenho desse algoritmo para uma determinada base de dados, essa classe de métodos ficou conhecida como AutoML (Feurer et al., 2019; Kedziora et al., 2020). A definição dessa classe de problemas é acompanhada de uma nova visão sobre o problema de seleção de algoritmo, nela a escolha de um algoritmo pode ser entendida como o ajuste de um hiperparâmetro adicional. Desse modo, métodos de HPO tradicionais podem ser utilizados diretamente para realizar a escolha de algoritmos (Kedziora et al., 2020).

Uma das soluções mais usadas para problemas de CASH é a busca em grade (do inglês, *grid search*). A busca em grade é um método de força bruta em que todas as combinações possíveis de hiperparâmetros são testadas. Embora simples de implementar, este algoritmo tem uma grande desvantagem, o espaço de busca cresce exponencialmente com o número de parâmetros do algoritmo testado. Este problema é agravado quando se considera a busca de algoritmos. Portanto, o seu uso em alguns casos é impossibilitado devido aos seus grandes requisitos de tempo (Hoos, 2012; Waring et al., 2020; He et al., 2021). Uma alternativa para a busca em grade é a busca aleatória (do inglês, *random search*). Nesta busca são realizados testes

em somente uma amostra de configurações obtidas a partir de todo espaço de busca. O tamanho dessa amostra é definido levando em consideração restrições de tempo e custo computacional. Apesar da busca aleatória não realizar testes em todas as configurações possíveis, é possível obter resultados similares à busca em grade (Bergstra e Bengio, 2012).

O trabalho de Bergstra e Bengio (2012) mostrou que os parâmetros de um algoritmo de aprendizado de máquina não são igualmente importantes para o seu desempenho. Desse modo, as buscas em grade e aleatória alocam muitos testes para a exploração de parâmetros sem importância. Em Feurer et al. (2019), a busca é iniciada em regiões do espaço de soluções mais prováveis de oferecer um bom desempenho. Para isso, é mantida uma memória persistente que armazena um conjunto de características de uma base de dados e a solução que obteve o melhor desempenho para essa base. Ao iniciar o processo de busca, esse conjunto de características é calculado para a nova base de dados de entrada e comparado com as características de bases na memória. Assim, o processo de busca se inicia a partir da melhor solução encontrada para a base de dados mais parecida. Os testes subsequentes são realizados a partir de alterações incrementais na solução inicial.

3.3.2 Adaptação

Os algoritmos de AutoML são capazes de escolher e ajustar de maneira eficiente um algoritmo de aprendizado de máquina para novos conjuntos de dados. No entanto, após implantados, a maioria desses algoritmos não são capazes de se adaptar a mudanças causadas por dados que evoluem ao longo do tempo (Kedziora et al., 2020; Celik e Vanschoren, 2021). O trabalho de Celik e Vanschoren (2021) investiga os impactos de mudança de conceitos no desempenho de algoritmos de aprendizado de máquina, e quais estratégias de adaptação podem ser empregadas para torná-los mais robustos. Essas estratégias são apresentadas a seguir.

- Detecção e retreinamento: Essa estratégia utiliza um detector de mudanças de conceito que observa os dados de entrada. Como consequência de uma mudança de conceito, ocorre o retreinamento do modelo de aprendizagem de máquina utilizando os novos dados. A Figura 3.2 ilustra o funcionamento dessa estratégia. Na figura as setas indicam a ocorrência de três possíveis eventos: Implantação de um modelo a partir da execução do AutoML, retreinamento do modelo implantado e detecção de mudanças de conceito. As chaves ilustram o tempo de vida de um algoritmo. Inicialmente o AutoML é executado para selecionar um algoritmo e um conjunto de hiperparâmetros (Modelo A), em seguida ocorre uma detecção de mudança de conceito e como consequência a solução inicial é substituída por uma nova instância do mesmo algoritmo com um conjunto de hiperparâmetros diferentes (Modelo A').

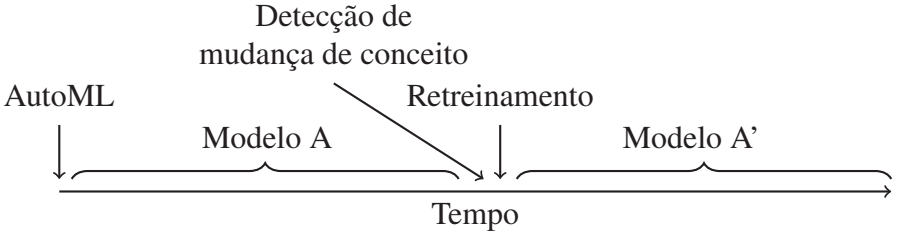


Figura 3.2: Estrategia de adaptação: detecção e retreinamento. Adaptado de Celik e Vanschoren (2021)

- Detecção e reinício: Diferentemente da estratégia anterior, essa estratégia substitui completamente a solução implantada. O AutoML é executado novamente, com uma

restrição no tempo de busca, visando encontrar uma nova solução em um espaço de tempo adequado. Uma desvantagem dessa estratégia é o consumo maior de recursos. Porém, em casos onde a mudança de conceito é extrema essa solução apresenta ganhos de desempenho significativos. Essa estratégia é ilustrada na Figura 3.3, nesse exemplo a segunda solução utiliza um algoritmo diferente.

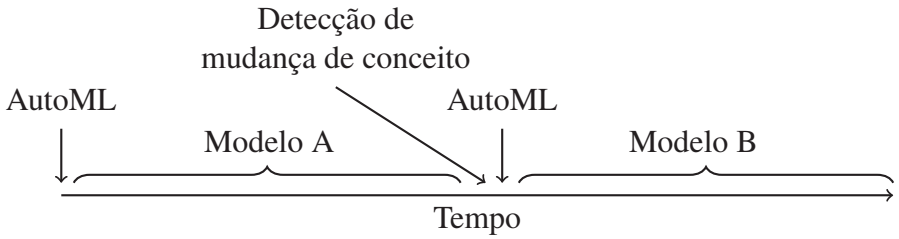


Figura 3.3: Estrategia de adaptação: detecção e reinício. Adaptado de Celik e Vanschoren (2021)

- Reinício periódico: Esta estratégia também consiste de múltiplas execuções do AutoML, exceto que, em vez de usar um detector de mudança de conceito, o AutoML é executado em intervalos regulares de tempo. Esta estratégia testa se um detector de mudança de conceito é realmente necessário ou se vale a pena substituir a solução implantada em intervalos de tempo predefinidos, apesar do custo computacional. Essa estratégia é ilustrada na Figura 3.4, nesse exemplo o AutoML é executado três vezes em intervalos de tempos regulares resultando em três soluções distintas.

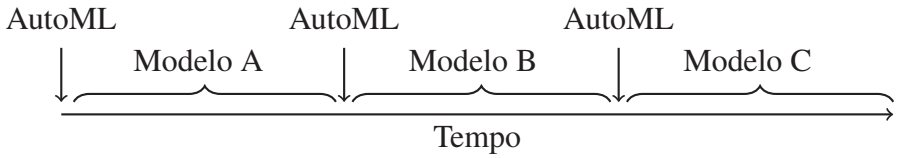


Figura 3.4: Estrategia de adaptação: reinício periódico. Adaptado de Celik e Vanschoren (2021)

3.4 RESUMO

Este capítulo apresentou os métodos e soluções existentes na literatura para o problema de detecção de *botnets*. Primeiramente, foi realizado uma diferenciação entre métodos baseado no modo de realizar a detecção. Em seguida, foram abordados trabalhos que fazem uso de aprendizado de máquina para a detecção de *botnets*, evidenciando a diversidade de soluções propostas por esses estudos.

As técnicas de aprendizado de máquina têm sido aplicadas com sucesso para a detecção de *botnets*. No entanto, a evolução constante e a diversidade de *botnets* dificultam a implantação desses métodos. Desse modo, é difícil encontrar uma técnica de detecção que ofereça uma solução permanente. Portanto, é necessário investigar métodos capazes de realizar essa adaptação, de preferência de maneira autônoma para diminuir o esforço humano. A análise de trabalhos sobre AutoML demonstrou que o cenário atual apresenta uma grande variedade de métodos capazes de fazer o gerenciamento de múltiplas técnicas de aprendizado de máquina e aplicar a técnica mais adequada para um determinado contexto. Além disso, foi possível observar que a utilização de AutoML junto com métodos de detecção de mudança de conceito permitem a adaptação de uma solução implantada em tempo real sem a necessidade de intervenção humana. Esses métodos são usados como base para a solução proposta por essa dissertação.

4 UM SISTEMA PARA A DETECÇÃO AUTÔNOMA E CONTÍNUA DE *BOTNETS*

Este capítulo apresenta o sistema para a detecção autônoma e contínua de *botnets*, denominado ANTE (do inglês, *AutoNomous boTnet dEtection*). Conforme ilustra a Figura 4.1, o sistema ANTE é baseado na fusão de três conceitos distintos: detecção de *botnets*, detecção de mudança de conceito e AutoML. A fusão desses conceitos possibilita a construção, implantação e a adaptação de técnicas de detecção de *botnets* de maneira autônoma, removendo a necessidade de intervenção humana.

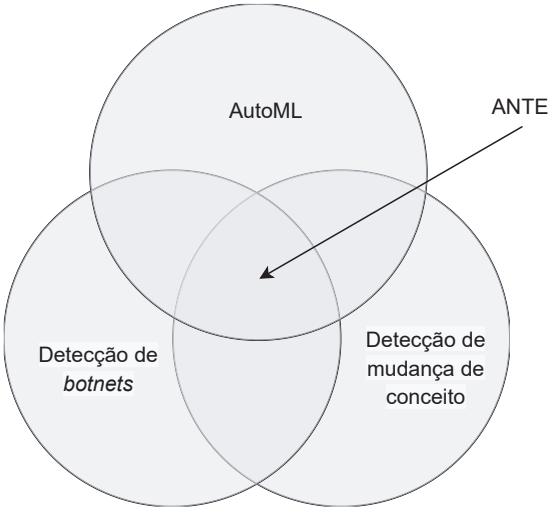


Figura 4.1: Conceitos utilizados pelo sistema ANTE

Este capítulo prossegue conforme descrito a seguir. A Seção 4.1 apresenta uma visão geral sobre o sistema proposto. As seções seguintes descrevem cada módulo do sistema. A Seção 4.2 detalha como o módulo de captura de dados faz a ingestão de tráfego de rede. A Seção 4.3 detalha como o módulo de caracterização de dispositivos calcula atributos de comportamento a partir de dados agrupados. A Seção 4.4 detalha como o módulo de adaptação de comportamento escolhe a melhor forma de detectar *bots* para um determinado contexto. A Seção 4.5 mostra como o módulo de classificação de *bots* realiza a detecção de *bots*.

4.1 VISÃO GERAL DO SISTEMA

O sistema ANTE é composto por quatro módulos principais, conforme ilustra a Figura 4.2. O módulo 1 é responsável por coletar o tráfego de rede utilizado para análise pelos demais módulos e extrair representações do comportamento de dispositivos a partir dos dados de rede brutos. O módulo 2 implementa um mecanismo para a detecção de mudanças de conceitos. Este módulo é responsável por indicar quando uma técnica de detecção deve ser substituída. O módulo 3 tem como objetivo construir um pipeline de aprendizado de máquina para a classificação de dispositivos mais adequado para um determinado contexto. O módulo 4 identifica os *bots* através da execução do pipeline de classificação construído pelo módulo 3. As próximas subseções detalham cada um dos módulos.

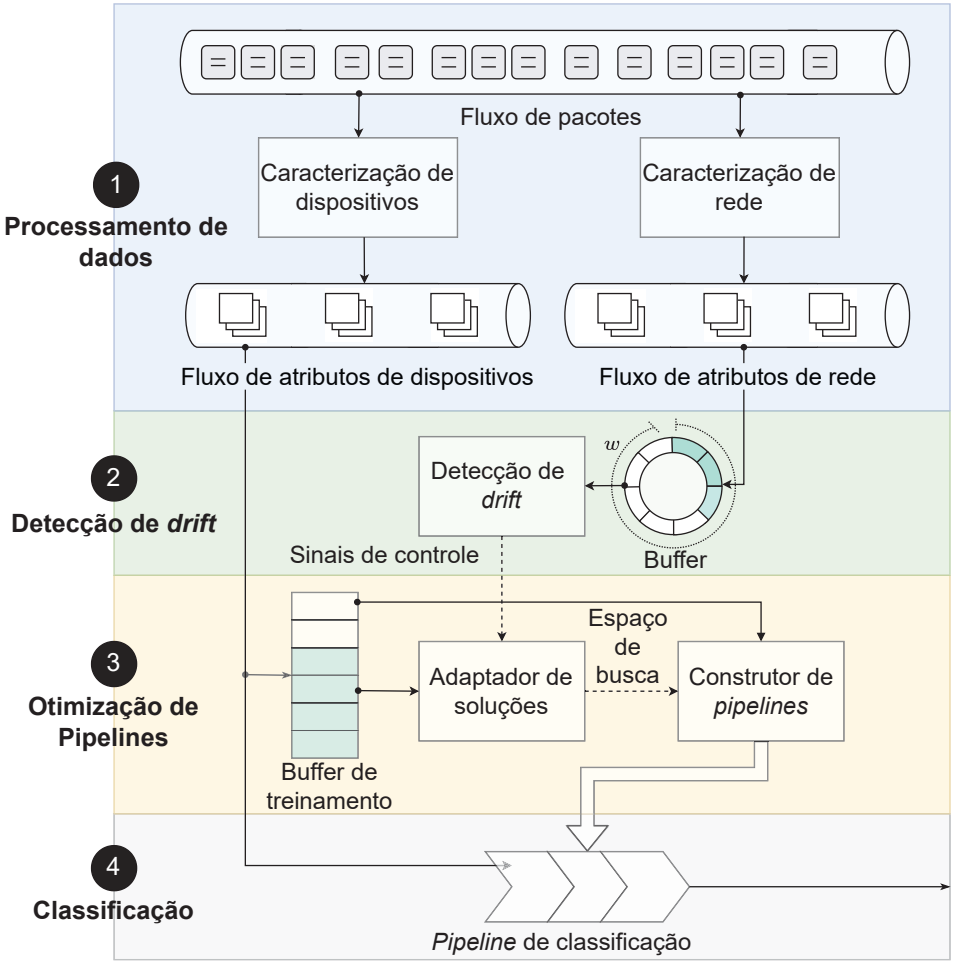


Figura 4.2: Arquitetura do sistema ANTE

4.2 MODULO DE PROCESSAMENTO DE DADOS

O modulo de captura de dados é responsável por monitorar, coletar e disponibilizar dados trafegados entre todos os dispositivos na rede para os demais módulos do sistema. O tráfego pode ser consumido pelo módulo de captura de duas maneiras: processamento em tempo real e bases de dados históricas. No modo de processamento em tempo real, todo o tráfego da rede é espelhado para o módulo de captura, assim é necessário posicionar o ANTE em um ponto central onde é possível observar a comunicação entre todos os dispositivos. No segundo modo de operação, o módulo de captura tem como entrada dados históricos no formato PCAP ou CSV (do inglês, *Comma-separated values*).

No módulo de pré-processamento, o tráfego da rede é segmentado por janelas de tempo. Essa divisão permite a detecção de *botnets* e o ajuste de soluções de maneira online, assim é possível realizar a identificação de *bots* logo após a captura de uma porção do tráfego total, a Figura 4.3 ilustra essa segmentação. Na figura, segmentos da captura distintos são diferenciados por grau de cinza. As janelas são caracterizadas por dois parâmetros: largura da janela e o tamanho do salto temporal. A largura da janela, representada como w, define o comprimento dos segmentos analisados. Já o tamanho do salto temporal, representado como s, define a frequência em que novos segmentos são formados.

A escolha dos parâmetros de segmentação são essenciais para a eficácia e eficiência da implementação do sistema. Um valor de w baixo resulta em um menor custo computacional,

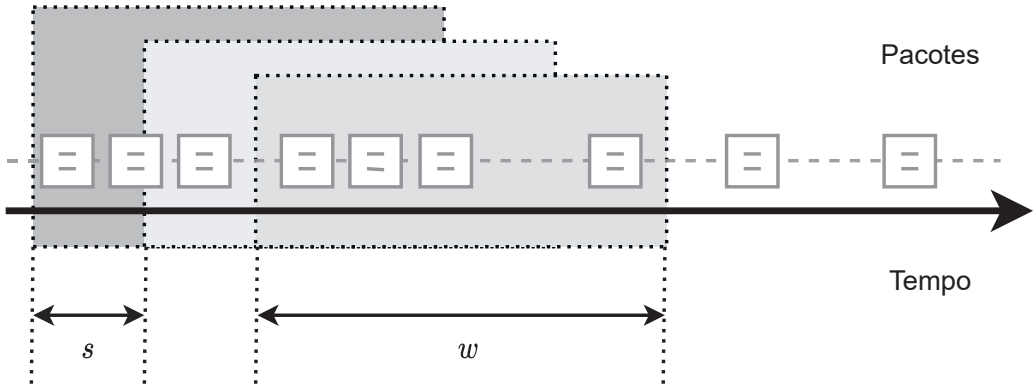


Figura 4.3: Segmentação de tráfego usando janelas de tempo

já que a quantidade de dados presentes em cada segmento é reduzida. Porém, essa redução de custo pode ser acompanhada de uma redução de eficácia, dado que os dados presentes no segmento podem não ser suficientes para capturar de maneira significativa o comportamento de um dispositivo. O custo computacional também pode ser reduzido com um aumento do parâmetro s. No entanto, essa alteração também resulta em complicações na eficiência, uma vez que um salto muito grande pode ignorar comportamentos com uma duração menor do que s.

Com os pacotes de rede agrupados em janelas de tempo, o módulo de processamento de dados realiza a extração de dois fluxos contínuos de dados: O fluxo de atributos de redes e o fluxo de atributos de dispositivos. O fluxo de atributos de rede é composto por informações que caracterizam o comportamento da rede. Para cada janela de tempo são calculados o total de pacotes enviados, a soma de bytes dos pacotes e o tempo mínimo, máximo e médio entre a transmissão de pacotes. Este fluxo é consumido diretamente pelo módulo 2, onde ocorre o monitoramento da rede e a detecção de possíveis mudanças de conceitos.

Já o fluxo de atributos de dispositivos é composto por informações que caracterizam um único dispositivo dentro da rede. Esses atributos são divididos em dois subgrupos: atributos de fluxo de pacotes e atributos de grafos. Os atributos de fluxo de pacotes, conforme proposto no trabalho de Zhao et al. (2013), contém informações estatísticas de todos os pacotes transmitidos entre dois dispositivo. Para a extração dessas informações, os pacotes presentes em uma janela de tempo são agrupados pela combinação do endereço IP da sua origem e destino, esse agrupamento é denominado como fluxo de pacotes (Zhao et al., 2013). A partir dos fluxos de pacotes o módulo de processamento de dados extrai uma lista de 18 atributos, descritos na Tabela 4.1.

Os atributos de grafos foram propostos pelo trabalho de Daya et al. (2019) com a intenção de capturar padrões de comunicação complexos que não podem ser obtidos a partir de atributos de fluxos. Esse atributos compreendem características de como os dispositivos se comunicam, independente dos detalhes dos pacotes transmitidos. Para a extração desse conjunto de atributos é necessária uma representação intermediária dos pacotes como grafo. Nesse grafo todos os dispositivos são representados como um vértice. Se um dispositivo transmite algum pacote para outro dispositivo, uma aresta é adicionada entre os vértices correspondentes (Chowdhury et al., 2017). Uma descrição de todos os atributos de grafos calculados pelo módulo de processamento de dados pode ser observada na Tabela 4.1.

4.3 MODULO DE DETECÇÃO DE *DRIFT*

O módulo de detecção de *drift* é fundamentado em um algoritmo baseado em similaridades para detectar mudanças no comportamento da rede. Mais especificamente, é utilizado uma

Tipo de atributo	Descrição
Fluxo	Soma em bytes do tamanho dos pacotes recebidos
Fluxo	Soma em bytes do tamanho dos pacotes enviados
Fluxo	Contagem de pacotes recebidos
Fluxo	Contagem de pacotes enviados
Fluxo	Média dos pacotes enviados
Fluxo	Tamanho em bytes do menor pacote enviado
Fluxo	Tamanho em bytes do maior pacote enviado
Fluxo	Variação em bytes dos pacotes enviados
Fluxo	Média dos pacotes recebidos
Fluxo	Tamanho em bytes do menor pacote recebido
Fluxo	Tamanho em bytes do maior pacote recebido
Fluxo	Variação em bytes dos pacotes recebidos
Fluxo	Menor intervalo entre o envio dos pacotes
Fluxo	Maior intervalo entre o envio dos pacotes
Fluxo	Média do intervalo entre o envio dos pacotes
Fluxo	Menor intervalo entre o recebimentos dos pacotes
Fluxo	Maior intervalo entre o recebimento dos pacotes
Fluxo	Média do intervalo entre o recebimento dos pacotes
Grafo	Número de arestas que entram em um vértice
Grafo	Número de arestas que saem de um vértice
Grafo	Número de caminhos mais curtos que passam pelo vértice
Grafo	Centralidade um vértice

Tabela 4.1: Atributos utilizados para a representação dos dispositivos

versão modificada do algoritmo OCDD proposto por (Gözüaçık e Can, 2021). A principal ideia por trás dessa técnica é utilizar um algoritmo de detecção de novidades para identificar quando uma porção dos dados de rede mais recentes diferem da porção mais antiga. Neste caso, quando existe uma diferença é indicado que ocorreu uma mudança de conceito.

A implementação do algoritmo é descrita no Algoritmo 1. O *buffer* ilustrado na Figura 4.2 armazena as últimas w amostras de atributos de rede obtidas a partir do fluxo de atributos, representado por S_r. As amostras são armazenadas no *buffer* de acordo com a sua ordem de chegada. Na fase de inicialização, o *buffer* é preenchido até totalizar w elementos. Após preenchido, os dados presentes no *buffer* são padronizados para valores com média igual a 0 e um desvio padrão igual a 1. Em seguida esses dados são utilizados como entrada para o treinamento do detector C, esse detector é uma instanciação do LocalOutlierDetection uma implementação de um algoritmo de detecção de novidades presente na biblioteca Scikit-learn (Pedregosa et al., 2011). Com o *buffer* preenchido e a primeira versão do detector C treinado, o algoritmo inicia a fase de detecção. Durante esta fase o algoritmo de detecção de novidades é executado sobre os dados presentes no *buffer*. Se o *buffer* contém uma proporção de dados classificados como novidade α, maior do que um limiar p, então um *drift* é detectado. Neste caso o módulo de detecção de *drift* sinaliza ao módulo de otimização de pipelines que a solução de detecção existente deve ser substituída. Após essa sinalização a porção de dados mais antiga é excluída do *buffer* e o detector C é treinado novamente somente com a porção de dados mais recente, esses novos dados representam o novo estado da rede.

Algoritmo 1 Pseudocódigo da Detecção de Drift

```
C ← LocalOutlierFactor()
S ← StandardScaler()
buffer ← ∅
while len(buffer) < w do
    X ← read_next(S_r)    // Obtém um elemento do fluxo de atributo de redes
    buffer.append(X)
end while
S.fit(buffer)
X_scaled ← S.transform(buffer)
C.fit(X_scaled)
for X ← read_next(S_r) do
    if len(buffer) < w then
        buffer.append(X)
    else
        X_scaled ← S.transform(X)
        P ← C.predict(X_scaled)
        α ← Σ_{i=1}^{w} P_i / w
        if α ≥ p then
            drift ← True    // Sinalização que ocorreu um drift
            del buffer[0:w × (1 − p)]
            C.fit(buffer)    // Retreinamento
            S.fit(buffer)    // Retreinamento
        else
            drift ← False
            del buffer[0]
        end if
    end if
end for
```

4.4 MÓDULO DE OTIMIZAÇÃO DE *PIPELINES*

Módulo de Otimização de *Pipelines* é responsável pela construção de um mecanismo para a detecção de *bots* e a implementação de estratégias de adaptação a mudanças no comportamento da rede. Conforme pode ser observado na Figura 4.2, o módulo de otimização de *pipelines* é constituído de três partes: um *buffer* de dados, o Componente de Adaptação de Soluções (CAS) e o componente de construção de *pipelines* (CCP). O *buffer* de dados é utilizado para armazenar informações de dispositivos disponibilizadas no fluxo de atributo de dispositivos. Este *buffer* tem um tamanho máximo especificado pelo administrador do sistema. Após a sinalização da ocorrência de um *drift*, ou a execução inicial do sistema, o módulo de otimização de *pipelines* inicializa o preenchimento do *buffer* com dados de dispositivos da rede.

Após o preenchimento do *buffer*, ocorre a execução do componente CAS. Este componente determina o conjunto de algoritmos de aprendizado de máquina que podem ser utilizados a partir dos dados presentes no *buffer*. Se os dados presentes no *buffer* forem enriquecidos com rótulos de dispositivos normais e *bots* por sistemas externos, ou pelo administrador do sistema, o CAS determina que é possível construir uma solução para a detecção de *bots* com técnicas de aprendizado de máquina supervisionado. Neste caso, o AutoSklearn (Feurer et al., 2019) será instanciado para a construção da solução. Se os os dados de treinamento possuem somente rótulos

de dispositivos normais, o CAS determina que a solução deve utilizar uma técnica de aprendizado semi supervisionado, neste caso é utilizado o algoritmo de *positive-unlabeled learning* proposto por (Elkan e Noto, 2008).

Utilizando como entrada o conjunto de possíveis soluções, o CCP é responsável por realizar o treinamento de técnicas de aprendizado de máquina e realizar a sua encapsulação em um mecanismo executável para a detecção de *bots*. Este mecanismo é composto por um conjunto de operações que são aplicadas de maneira sequencial, onde o resultado de uma operação é utilizado como entrada para a operação posterior, este fluxo de trabalho é denominado pipeline. A última operação do pipeline terá como saída a classificação de um dispositivo em *bot* ou dispositivo normal. As etapas intermediárias visam garantir a execução da última etapa sem a ocorrência de erros, um exemplo desse tipo de operação é o preenchimento de valores obrigatórios esperados pela etapa de classificação.

4.5 MÓDULO CLASSIFICAÇÃO DE BOTS

Após a definição do *pipeline* de classificação o Módulo Classificação de Bots (MCB) é responsável por invocar a sua execução. O MCB direciona a saída do módulo de caracterização para a entrada do *pipeline* de classificação. Esse módulo também é responsável por integrar a saída do sistema ANTE, com sistemas externos. Como por exemplo, a criação de alertas em sistemas de monitoramento.

4.6 RESUMO

Este capítulo apresentou o ANTE, um sistema para a detecção autônoma e contínua de *botnets*. Inicialmente, foi apresentada uma visão geral da sua arquitetura e da interação dos seus módulos. Em seguida, foi detalhado o funcionamento de todos os módulos individualmente. Estes módulos atuam em conjunto para resolver diferentes problemas associados com a detecção de *botnets*.

5 AVALIAÇÃO

Este capítulo apresenta uma avaliação preliminar da eficácia do sistema ANTE na detecção de *botnets*. Esta avaliação analisa os módulos propostos pelo sistema e examina de maneira incremental a integração desses módulos na formação de um sistema de detecção de *botnets* autônomo. A Seção 5.1 descreve as bases de dados usadas para os testes. A Seção 5.2 apresenta as métricas utilizadas na avaliação dos experimentos. A Seção 5.3 valida o uso de classificadores supervisionados para a detecção de *botnets*, e verifica a existência de um classificador capaz de obter o melhor desempenho para todos os cenários. A Seção 5.3 avalia uma maneira de remover a necessidade de intervenção humana no processo de seleção de um classificador. Subsequentemente, na Seção 5.5 é investigada uma maneira de construir de maneira autônoma e eficiente todo o *pipeline* de classificação.

5.1 CENÁRIOS AVALIADOS

A avaliação do sistema ANTE ocorreu a partir da sua execução em 9 cenários distintos. As bases de dados utilizadas para avaliar o sistema foram escolhidas visando criar um ambiente de testes que apresenta uma grande variedade de tipos de *botnets* e protocolos diferentes. Além disso, foram selecionadas bases de dados provenientes de dispositivos IoT, visando à avaliação do sistema com diferentes tipos de dispositivos. Os cenários de testes são apresentados a seguir.

Cenário 1: O projeto Stratosphere[1] disponibiliza um conjunto de bases de dados denominado CTU-13 (García et al., 2014). Esse conjunto de bases possui 13 capturas de tráfego de *botnets* gravados na Universidade Técnica Tcheca na República Tcheca. Utilizando parte dos dados do projeto, foi definido que o primeiro cenário do presente trabalho seria correspondente a captura 52 ou cenário 11 da CTU-13[2]. Esta base foi selecionada por capturar um ataque DDoS utilizando o Protocolo de Mensagens de Controle da Internet (do inglês, *Internet Control Message Protocol* - ICMP). Além disso, essa base possui aproximadamente 4 GB de fluxo e 3 dispositivos infectados, porém um dos *bots* pouco atua durante o ataque. O ponto inicial do ataque ocorreu às 10:52:39 do dia 18 de agosto 2011 e foi obtido por meio da documentação da base.

Cenário 2: Como no cenário anterior este também faz uso de uma base do conjunto CTU-13. Neste caso foi escolhida a captura 51 ou cenário 10 da CTU-13[3], por capturar um ataque DDoS usando o Protocolo de Datagrama do Usuário (do inglês, *User Datagram Protoco* - UDP). Esta base é bem maior que a anterior visto que possui aproximadamente 66 GB de fluxo. Foram coordenados ataques no dia 18 agosto de 2011 durante dois períodos, onde foram realizados ataques com características diferentes. No primeiro período foram conduzidos ataques de SYN Flood e ACK DDoS e no segundo período foram testados ataques do tipo ICMP Flood.

Cenário 3: Este cenário também faz uso da captura 51 do conjunto CTU-13 mencionado no cenário 2. Porém, os dados analisados fazem referência ao segundo período onde um ataque do tipo ICMP Flood foi conduzido. Este ataque foi iniciado no dia 18 agosto de 2011 às 14:43:27 CEST e conta com 10 dispositivos infectados. Essa base foi selecionada visando à ampliação da variedade de ataques testados, incorporando um ataque que faz uso do protocolo ICMP.

Cenário 4: A Universidade de New Brunswick (UNB) possui um projeto onde são disponibilizados conjuntos de dados contendo *bots*. Assim, este trabalho utiliza o conjunto

[1] www.stratosphereips.org Acessado em: 18/10/2021

[2] https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-52 Acessado em: 18/10/2021

[3] https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-51 Acessado em: 18/10/2021

CICDDoS2019[4] para definir o quarto cenário. Este conjunto foi publicado no ano de 2019 e conta com diferentes tipos de ataques coletados em dois dias, totalizando 20 GB de dados (Sharafaldin et al., 2019). Dos 12 ataques reportados nessa base, este cenário analisa o ataque *PortMap*. Esse ataque foi escolhido pela presença baixa de *bots* em relação a dispositivos normais, visando à avaliação o sistema em cenários com poucos dados. Segundo a documentação, o ataque ocorreu no primeiro dia de coleta durante o período de 9:43 a 9:51. Porém, durante a análise dos dados coletados não foi possível encontrar fluxo de rede no horário reportado. Então como o objetivo era analisar o primeiro ataque conduzido no primeiro dia, foi verificado que o primeiro pico de tráfego de pacotes ocorreu no dia 12 de janeiro de 2019 às 14:36:10 Tempo Universal Coordenado (UTC).

Cenário 5: Este cenário utiliza outro contexto do CICDDoS2019. O ataque selecionado para este cenário foi o NetBIOS. O NetBIOS é uma interface de programação de aplicativos que pode fazer uso do Protocolo de Controle de Transmissão (do inglês, *Transmission Control Protocol* - TCP) ou UDP. Esse ataque foi selecionado por utilizar ambos os protocolos. Esse foi o segundo ataque conduzido no primeiro dia do experimento. Segundo a documentação este ataque ocorreu entre às 10:21 e 10:30 do primeiro dia. Porém, analisando os dados disponibilizados não foi possível encontrar fluxo de rede neste horário, deste modo, o pico de tráfego ocorrido em 12 de janeiro às 15:06:10 UTC foi escolhido para ser o ponto base da análise. Algumas características do cenário anterior também são verificadas neste. Por exemplo, a presença de apenas um *bot* ativo, sendo que este dispositivo gera muito tráfego na rede e ele está presente antes do início do ataque. Contudo, este cenário possui menos dispositivos conectados na rede ao longo da análise chegando a um pico de 110 dispositivos.

Cenário 6: O cenário 6 é derivado do trabalho da Universidade de Victoria no Canadá e foi nomeado como ISOT HTTP Botnet Dataset[5]. Este cenário possui 780 MB de fluxo de rede e 9 dispositivos *bots* (Alenazi et al., 2017). Esta base de dados foi escolhida por capturar um ataque utilizando o protocolo HTTP. O ponto central da análise foi definido sendo as 21:39:31 do dia 30 maio 2017, visto que, o objetivo do ataque era o roubo de informações.

Cenário 7: Este cenário foi definido com base em uma captura frequentemente utilizada na literatura, a base de dados denominada DDoS Attack 2007 disponibilizada pela CAIDA[6]. Este cenário foi selecionado por apresentar somente tráfego associado a *botnets*. Isso significa que, tirando os alvos do ataque, todos os outros dispositivos são *bots*. O ponto central da análise é dia 4 de agosto de 2007 às 21:14:00.

Cenário 8: Este cenário é parte de um projeto disponibilizado pela UNB denominado CSE-CIC-IDS2018. Este cenário foi escolhido por sua variedade de ataques. Ele conta com sete ataques diferentes e com mais de 200 GB de fluxo de dados. Entre os ataques tem-se a ação de *bots* para quebrar senhas utilizando força bruta, *bots* para roubar dados sensíveis e ataques de negação de serviço. O ataque escolhido foi o DDOS-HOIC conduzido no dia 21/02/2018 sendo que o ponto central da análise ocorreu às 14:09:00.

Cenário 9: O último cenário escolhido foi o conjunto de dados BoT-IoT, criado no *Cyber Range Lab* da UNSW Canberra. Esse cenário incorpora uma combinação de tráfego normal e de *botnet*. Esta base foi selecionada por capturar tráfego de rede oriundo de dispositivos comuns e dispositivos IoT. Os arquivos de captura totalizam 69 GB de tamanho, com mais de 72.000.000 registros.

---

[4] www.unb.ca/cic/datasets/ddos-2019.html Acessado em: 18/10/2021

[5] https://www.uvic.ca/ecs/ece/isot/datasets/botnet-ransomware/index.php Acessado em: 18/10/2021

[6] Tivemos acesso a esta base devido à parceria entre a Universidade Federal do Paraná (UFPR)/Brasil e a Universidade Cernegie Mellon (CMU)/EUA. URL: `https://www.caida.org/data/passive/ddos-20070804_dataset.xml` Acessado em: 18/10/2021

5.2 MÉTRICAS

Para avaliar o desempenho do sistema ANTE na detecção de *botnets*, foram utilizadas as métricas de acurácia, precisão, *recall* e o *F1-Score*. A acurácia relaciona o total de acertos com o total de itens classificados, quantificando quantos dispositivos foram corretamente classificados. Porém em casos onde a distribuição das classes é desbalanceada, esta métrica pode apresentar falso sentimento de bons resultados, pois o classificador pode acertar todas as previsões para a classe majoritária e errar todas as previsões para a classe minoritária e mesmo assim possuir alta acurácia. Para a plena análise de um classificador, é necessário calcular as métricas de precisão, *recall* e o *F1-Score*. Essas métricas são baseadas na quantidade de verdadeiros positivos (TP), falsos positivos (FP), falsos negativos (FN), verdadeiros negativos (TN). Deste modo, é possível obter essas métricas para a classe dos não *bots* e dos *bots*. Devido à variação dos TP, FP, FN e TN é necessário convencionar o uso deles neste trabalho, deste modo, a expressão "verdadeiros positivos" representa os elementos que foram classificados como *bots* e realmente eram *bots*. Os "falsos positivos" são os elementos que foram classificados como *bot*, porém eram dispositivos comuns. Os "falsos negativos" correspondem aos elementos que foram classificados como não *bots*, porém eram *bots*. Os "verdadeiros negativos" são os elementos que não eram *bots* e foram classificados como dispositivos comuns. As fórmulas das métricas estão apresentadas a seguir.

$$acuracia = \frac{TP + TN}{TP + FP + FN + TN} \tag{5.1}$$

$$precisao = \frac{TP}{TP + FP} \tag{5.2}$$

$$sensibilidade = \frac{TP}{TP + FN} \tag{5.3}$$

$$F1 - Score = 2. \frac{precisao\ .\ sensibilidade}{precisao + sensibilidade} \tag{5.4}$$

5.3 AVALIAÇÃO DE CLASSIFICADORES PARA A DETECÇÃO DE BOTS

Esta seção tem dois objetivos inter-relacionados. O primeiro é realizar uma análise empírica da aplicação de técnicas de aprendizado de máquina para a detecção de bots. A partir dessa análise, o segundo objetivo é verificar a existência de um classificador capaz de obter o melhor desempenho. A metodologia de testes é apresentada na Subseção 5.3.1. Para realizar a comparação dos resultados foram utilizadas as métricas de *precisão*, *recall* e *F1-score*, os resultados são apresentados na Subseção 5.3.2.

5.3.1 Metodologia

A análise empírica foi realizada com uma implementação preliminar do sistema ANTE, conforme ilustrado na Figura 5.1. Nesta implementação foram desenvolvidos os módulos de captura de dados e caracterização de dispositivos. O módulo de processamento consome dados históricos no formato PCAP. Após ingeridos esses dados são segmentados com janelas de tempo

distintas, os atributos extraídos de cada janela são usados como entrada para o algoritmo de classificação.

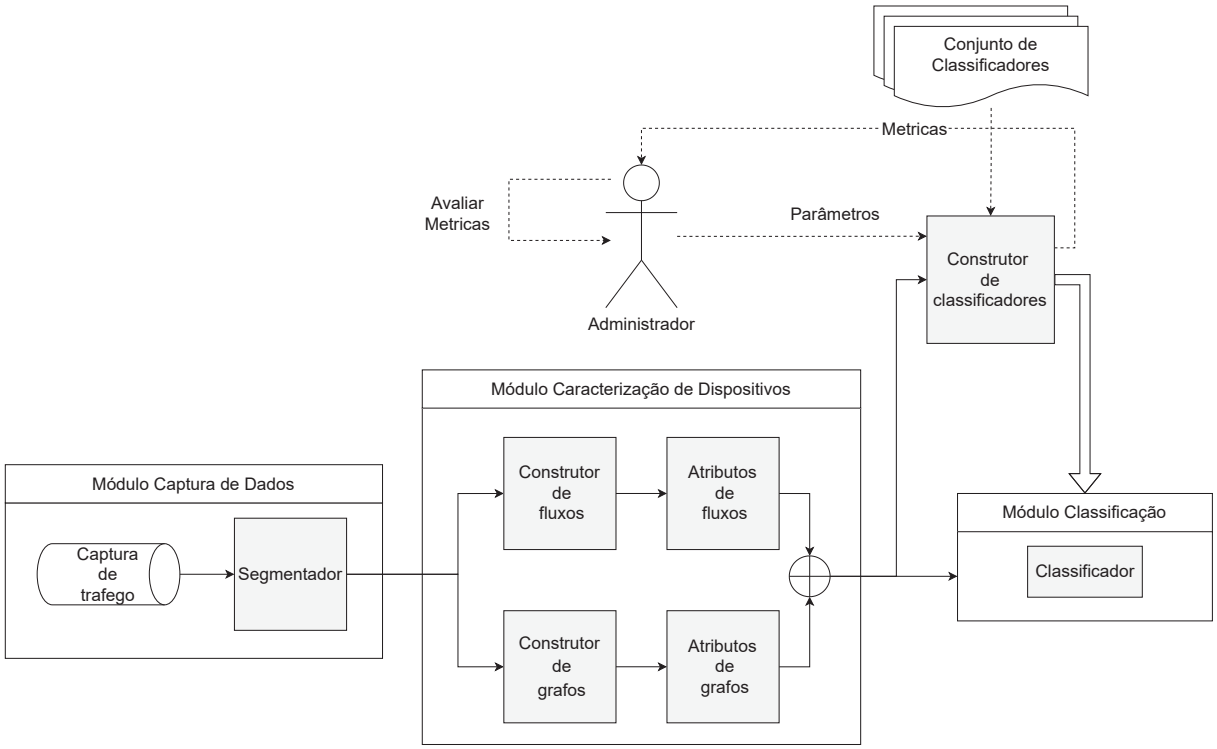


Figura 5.1: Arquitetura do sistema ANTE para a escolha manual de um classificador

Ao todo, o tráfego é dividido em 6 segmentos. Esses segmentos são definidos tendo como referência o horário do pico de um ataque. Para o treinamento das técnicas de classificação são usados os atributos extraídos a partir do primeiro segmento da captura, esse segmento se inicia dois minutos antes do pico do ataque e tem duração de um minuto. Para os testes, são usados os 5 segmentos subsequentes. O primeiro teste visa avaliar a capacidade do classificador em detectar *bots* antes do momento mais prejudicial do ataque, para isso foi utilizado um segmento da captura que tem início um minuto antes do ataque com uma duração de um minuto. Os quatro segmentos de testes restantes ocorrem depois do ataque e todos tem duração de 30 segundos, o objetivo desses testes é avaliar a performance do classificador durante o ataque. Figura 5.2 ilustra a segmentação.

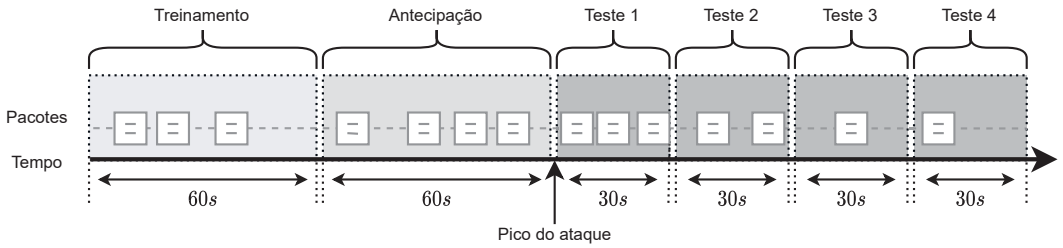


Figura 5.2: Segmentação de tráfego para a avaliação de um classificador

Para a detecção de *botnets*, foram utilizadas as técnicas de classificação supervisionada implementadas no scikit-learn (Pedregosa et al., 2011). Essa biblioteca oferece um conjunto diverso de implementação de algoritmos de aprendizado de máquina. Os classificadores foram instanciados com os parâmetros definidos como padrão pelo scikit-learn.

5.3.2 Resultados

Esta Subseção apresenta uma avaliação da implementação do sistema ANTE descrita na Subseção 5.3.1. Para esta avaliação foram utilizados os cenários de 1 a 8. O cenário 9 foi excluído dessa avaliação, visto que objetivo era verificar a diversidade de soluções de detecção para *botnets* que exibem um comportamento similar.

Cenário 1: Este cenário foi treinado com a presença de dois *bot*, pois antes do início do ataque esse *bots* já estavam ativos enviando e/ou recebendo informações. Apesar da quantidade de fluxo capturado, as etapas de treinamento, antecipação e testes foram executadas rapidamente, deste modo a etapa mais onerosa foi a extração de características. Para o classificador KMeans, o resultado foi ruim em todas as janelas. Para a janela antes do ataque o algoritmo que melhor identificou a presença de *bots* foi o classificador de Regressão Logística. O classificador *gradient boosting* e o *random forest* apresentaram bons resultados no primeiro e no segundo teste após o início do ataque, quando existiam mais dispositivos na rede (9 e 12 dispositivos respectivamente). Porém, quando a quantidade de dispositivos diminuiu no terceiro e no quarto período (7 e 3 dispositivos, respectivamente), eles passaram a errar mais que o classificador *regressão logística*. A Tabela 5.1 apresenta a acurácia e a média da precisão, *recall* e *F1-score* para os principais resultados.

Algoritmo	Métricas	Antecipação	Teste 1	Teste 2	Teste 3	Teste 4
Gradient Boosting	**Acurácia**	75%	**89%**	**83%**	71%	67%
	Precisão	65%	**93%**	**80%**	36%	33%
	Recall	61%	**83%**	**89%**	50%	50%
	F1-Score	62%	**86%**	**81%**	42%	40%
Regressão Logística	**Acurácia**	**92%**	78%	75%	**100%**	**100%**
	Precisão	**88%**	88%	75%	**100%**	**100%**
	Recall	**94%**	67%	83%	**100%**	**100%**
	F1-Score	**90%**	68%	73%	**100%**	**100%**

Tabela 5.1: Principais resultados para o cenário 1.

Cenário 2: Este cenário conta com mais dispositivos que o cenário anterior, deste modo o treinamento contou com dados de 33 dispositivos sendo 10 *bots*. O classificador *gradient boosting* e o *random forest* identificaram todos os *bots* na janela anterior ao início do ataque e nas duas primeiras janelas do ataque. Porém, quando a quantidade de dispositivos diminuiu esses algoritmos foram incapazes de diferenciar os dispositivos normais dos *bots*. A Tabela 5.2 apresenta os principais resultados obtidos com a análise deste cenário.

Algoritmos	Métricas	Antecipação	Teste 1	Teste 2	Teste 3	Teste 4
Random Forest	**Acurácia**	100%	100%	100%	9%	9%
	precisão	100%	100%	100%	5%	5%
	Recall	100%	100%	100%	50%	50%
	F1-Score	100%	100%	100%	9%	9%

Tabela 5.2: Principal resultado para o cenário 2.

Cenário 3: Este cenário conta com 30 dispositivos, sendo que 10 deles são *bots*. Neste cenário a quantidade excessiva de pacotes enviados pelos *bots* facilitou a sua detecção. Dos quatro algoritmos testados, os classificadores regressão logística e *random forest* acertaram todas as classificações.

Cenário 4: As características deste cenário fazem dele um desafio para os algoritmos de aprendizagem de máquina, visto que, no treinamento, apenas um *bot* estava ativo entre os 36 dispositivos que compunham a rede. Essa diferença torna-se ainda maior ao longo dos testes quando a rede possui 523 dispositivos normais e um malicioso. A principal diferença está relacionada com a grande quantidade de informação originada por este dispositivo, sendo que no início do ataque, o *bot* enviou 6.135.105 pacotes a mais do que o segundo dispositivo mais ativo. Como pode ser observado na tabela 5.3, todos os métodos encontraram dificuldades em identificar o *bot* neste contexto. Na janela de antecipação, o classificador *random forest* e o classificador *regressão logística* foram os que melhor conseguiram separar o *bot* do fluxo normal, gerando 1 e 2 falsos negativos respectivamente. Ainda na fase anterior ao início do ataque, o classificador KMeans gerou dois agrupamentos, um com 8 dispositivos incluindo o *bot* e outro com 31 dispositivos. Considerando que todos deste agrupamento fossem classificados como *bots* o classificador KMeans geraria 7 falsos negativos. Apesar desta alta taxa do erro, esse foi o único capaz de separar o *bot* depois do início do ataque.

Algoritmo	Métricas	Antecipação	Teste 1	Teste 2	Teste 3	Teste 4
KMeans	**Acurácia**	82%	97%	99%	99%	99%
	Precision	56%	57%	60%	75%	62%
	Recall	91%	99%	100%	100%	99%
	F1-Score	56%	62%	66%	83%	70%

Tabela 5.3: Principal resultado para o cenário 4.

Cenário 5: Este cenário possui menos dispositivos conectados na rede ao longo da análise chegando a um pico de 110 dispositivos. Essa mudança causou resultados diferentes da análise anterior, sendo que neste caso os classificadores *regressão logística* e *gradient boosting* acertaram todas as classificações na janela de antecipação e nas janelas de testes.

Cenário 6: Este cenário foi treinado com a presença de 6 *bots* entre os 33 dispositivos que possuíam troca de pacotes antes do início do ataque. Em todas as janelas o classificador KMeans criou os agrupamentos com a maioria dos dispositivos normais junto com os *bots*, e os agrupamentos que possuíam poucos dispositivos não possuíam *bots*. O algoritmo que melhor classificou a janela de antecipação foi o classificador *random forest* acertando todas as classificações. Após o início do ataque o classificado *random forest* produziu apenas 2 falsos positivos no teste 2 e 1 falso positivo no teste 4, todos os resultados estão na Tabela 5.4.

Algoritmo	Métricas	Antecipação	Teste 1	Teste 2	Teste 3	Teste 4
Random Forest	**Acurácia**	100%	100%	95%	100%	97%
	Precision	100%	100%	97%	100%	98%
	Recall	100%	100%	86%	100%	94%
	F1-Score	100%	100%	90%	100%	96%

Tabela 5.4: Principal resultado para o cenário 6.

Cenário 7: O treinamento contou com 12 dispositivos, sendo 11 *bots* e a vítima, porém ao longo dos testes foi possível verificar um crescimento na quantidade de dispositivos trocando informações, chegando em um pico de 1286 dispositivos na última janela de teste. O classificador KMeans e o *random forest* conseguiram separar o servidor dos dispositivos infectados, tanto na janela de antecipação quanto nas janelas de teste.

Cenário 8: Apesar da existência de dez *bots*, este foi o único cenário que não contou com a presença de *bots* no treinamento. Provavelmente, esse fator influenciou no ruim resultado

dos algoritmos de aprendizagem de máquina supervisionado, visto que, nenhum algoritmo foi capaz de classificar corretamente um *bot* sequer. O algoritmo KMeans conseguiu separar parte do fluxo malicioso do normal. Utilizando o menor grupo como aquele dos *bots* foi possível observar falsos positivos e falsos negativos, porém, exceto na janela de antecipação, o algoritmo classifica mais corretamente os *bots* do que erra, como é possível verificar na Tabela 5.5.

Algoritmo	Métricas	Antecipação	Teste 1	Teste 2	Teste 3	Teste 4
KMeans	**Acurácia**	99,85%	99,89%	99,89%	99,93%	99,93%
	Precision	70%	89%	88%	92%	92%
	Recall	83%	94%	100%	100%	100%
	F1-Score	75%	91%	93%	95%	95%

Tabela 5.5: Principais resultados para o cenário 8.

5.4 AVALIAÇÃO DA SELEÇÃO AUTÔNOMA DE CLASSIFICADORES PARA A DETECÇÃO DE BOTS

Esta seção tem como objetivo avaliar estratégias para a seleção e construção autônoma de técnicas para classificação de *bots*. A Seção 5.4.1 detalha a metodologia e a Seção 5.4.2 apresenta a comparação dos resultados.

5.4.1 Metodologia

Para testar a seleção autônoma de uma técnica de classificação, o módulo de adaptação de comportamento proposto na Seção 4.4 é adicionado à implementação da seção anterior, conforme ilustrado na Figura 5.3. Mais especificamente, Esta seção avalia a implementação do componente de otimização de pipelines (COP) para a escolha de uma técnica de classificação. A fim de testar estratégias diferentes para a otimização da escolha do classificador, foram realizadas duas implementações diferentes de COP. O ANTE-AG é um componente baseado no AutoGluon (Erickson et al., 2020), uma solução de AutoML que visa testar inicialmente soluções confiáveis e de baixo custo, deixando para um segundo momento a avaliação de soluções que exigem mais poder computacional. Já o ANTE-H2 é baseado no H20 (LeDell e Poirier, 2020), essa solução de AutoML cria uma *ensemble* de classificadores com foco em diversidade de soluções. Os dois componentes são testados com a mesma restrição de tempo, nesse caso 5 minutos para a escolha da melhor técnica.

Para a avaliação da implementação proposta nessa seção, a captura é dividida em três partes: treinamento, antecipação e testes. Os dados de treinamento são obtidos a partir de um segmento com duração de dois minutos, tendo fim 30 segundos antes do pico do ataque. A antecipação tem duração de 30 segundos, tendo início após o fim da janela de treinamento. Os testes restantes são realizados usando o período de dois minutos seguintes ao pico do ataque. Diferentemente da seção anterior, a captura é segmentada usando janelas do mesmo tamanho, com o mesmo salto temporal, conforme ilustra a Figura 5.4.

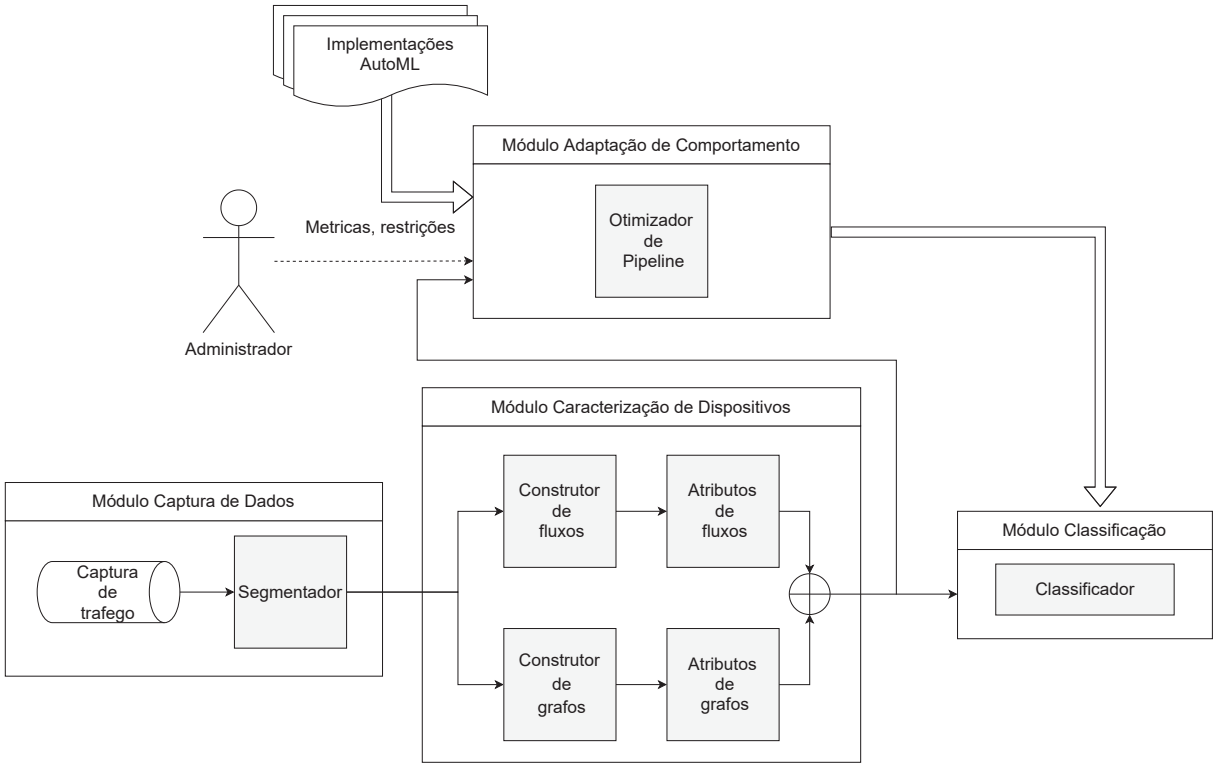


Figura 5.3: Arquitetura do sistema ANTE para a escolha autônoma de um classificador

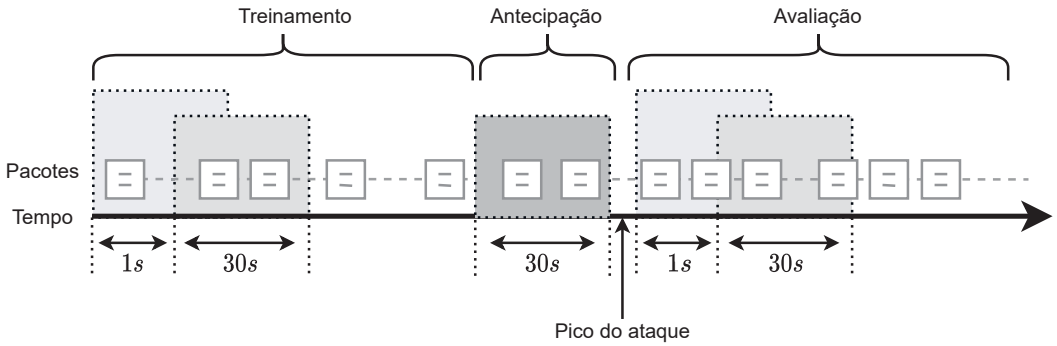


Figura 5.4: Segmentação de tráfego com janelas deslizantes do mesmo tamanho.

5.4.2 Resultados

Esta Subseção apresenta uma avaliação da implementação do sistema ANTE descrita na Subseção 5.4.1. Para esta avaliação foram utilizados os cenários de 2, 5, 6 e 9. Estes cenários foram escolhidos por terem se mostrado mais desafiadores de acordo com os resultados da Seção 5.3.2. Além disso, a escolha desses cenários visam testar a implementação do sistema usando *botnets* com comportamentos heterogêneos, variando desde o protocolo até as classes de dispositivos.

Cenário 2: Como pode ser observado na Tabela 5.6, o componente ANTE-H2 e o ANTE-AG conseguiram classificar corretamente todos os dispositivos nas janelas de antecipação e testes. É possível observar novamente que os classificadores criados com parâmetros padrões exibem um desempenho menor. É importante ressaltar que geralmente uma acurácia de 100% é um sinal de *overfitting*, porém nesse caso a alta acurácia é causada pela facilidade de separar os dispositivos comuns dos *bots*.

Técnica	Janela	Acurácia	Precisão	*Recall*	*F1-Score*
ANTE-H2	Antecipação	100.0%	100.0%	100.0%	100.0%
	Testes	100.0%	100.0%	100.0%	100.0%
ANTE-AG	Antecipação	100.0%	100.0%	100.0%	100.0%
	Testes	100.0%	100.0%	100.0%	100.0%
SVM	Antecipação	92.31%	50.0%	100.0%	66.67%
	Testes	73.91%	68.97%	100.0%	81.63%
Naïve Bayes	Antecipação	92.31%	0.0%	0.0%	0.0%
	Testes	100.0%	100.0%	100.0%	100.0%
MLP	Antecipação	100.0%	100.0%	100.0%	100.0%
	Testes	97.1%	95.24%	100.0%	97.56%

Tabela 5.6: Resultados para o CTU-13

Cenário 5: Como visto na Tabela 5.7, os resultados obtidos no CICDDoS são condizentes com os testes anteriores. As duas implementações do COP classificaram corretamente todas as instâncias de *bots*, enquanto os classificadores criados com os parâmetros padrões apresentam uma performance menor.

Técnica	Janela	Acurácia	Precisão	*Recall*	*F1-Score*
ANTE-H2	Antecipação	100.0%	100.0%	100.0%	100.0%
	Testes	100.0%	100.0%	100.0%	100.0%
ANTE-AG	Antecipação	100.0%	100.0%	100.0%	100.0%
	Testes	100.0%	100.0%	100.0%	100.0%
SVM	Antecipação	97.87%	0.0%	0.0%	0.0%
	Testes	98.75%	0.0%	0.0%	0.0%
Naïve Bayes	Antecipação	97.87%	50.0%	100.0%	66.67%
	Testes	100.0%	100.0%	100.0%	100.0%
MLP	Anticipation	93.62%	25.0%	100.0%	40.0%
	Testes	99.07%	57.14%	100.0%	72.73%

Tabela 5.7: Resultados para o CICDDoS

Cenário 6: A Tabela 5.8 mostra os resultados obtidos a partir das escolha automática dos classificadores para o cenário ISOT. O componente ANTE-H2 foi capaz de classificar corretamente todos os dispositivos antes do pico do ataque. Porém, durante a janela de testes, o componente ANTE-H2 classificou incorretamente um *bot* como dispositivo normal. O componente ANTE-AG foi capaz de obter uma acurácia de 96% antes do pico do ataque, e 97.69% depois do pico. Na Tabela 5.8 também é possível observar a dificuldade do SVM, NV e MLP em classificar os dispositivos corretamente, essa diferença demonstra a habilidade do COP em escolher um conjunto de parâmetros bem ajustado.

Cenário 9: Conforme ilustrado na Tabela 5.9, o componente ANTE-H2 obteve os melhores resultados, classificando corretamente todos os dispositivos na janela de antecipação e obtendo 88.02% de precisão na janela de testes. O componente ANTE-AG também foi capaz de classificar corretamente todos os dispositivos na janela de antecipação, porém o ANTE-H2 teve uma performance melhor na janela de testes.

Técnica	Janela	Acurácia	Precisão	*Recall*	*F1-Score*
ANTE-H2	Antecipação	100.0%	100.0%	100.0%	100.0%
	Testes	98.46%	100.0%	91.67%	95.65%
ANTE-AG	Antecipação	96.0%	100.0%	85.71%	92.31%
	Testes	97.69%	100.0%	87.5%	93.33%
SVM	Antecipação	88.0%	100.0%	57.14%	72.73%
	Testes	90.77%	100.0%	50.0%	66.67%
Naïve Bayes	Antecipação	72.0%	50.0%	100.0%	66.67%
	Testes	73.85%	40.0%	83.33%	54.05%
MLP	Antecipação	100.0%	100.0%	100.0%	100.0%
	Testes	95.38%	90.91%	83.33%	86.96%

Tabela 5.8: Resultados para o ISOT

Técnica	Janela	Acurácia	Precisão	*Recall*	*F1-Score*
ANTE-H2	Antecipação	100.0%	100.0%	100.0%	100.0%
	Testes	96.69%	88.02%	100.0%	93.63%
ANTE-AG	Antecipação	100.0%	100.0%	100.0%	100.0%
	Testes	79.2%	53.92%	100.0%	70.06%
SVM	Antecipação	77.13%	54.4%	100.0%	70.46%
	Testes	31.72%	26.28%	100.0%	41.62%
Naïve Bayes	Antecipação	64.05%	41.98%	83.33%	55.84%
	Testes	62.63%	38.14%	86.04%	52.85%
MLP	Antecipação	98.48%	100.0%	94.44%	97.14%
	Testes	70.55%	43.82%	74.39%	55.15%

Tabela 5.9: Resultados para o BoT-IoT

Cenário	Pré-processamento			Classificador
	Dados		Características	
	Imputação	Normalização		
ISOT	Mean	RobustScaler	SelectPercentile	AdaBoost
CTU-13	Mean	Standardize		RandomForest
CICDDoS	MostFrequent	MinMax	PolynomialFeature	RandomForest
BoT-IoT	Mean	MinMax	FastICA	SVM

Tabela 5.10: *Pipeline* de classificação escolhido pelo ANTE-AS para diferentes cenários de teste

5.5 AVALIAÇÃO DA OTIMIZAÇÃO AUTÔNOMA DE UM PIPELINE DE CLASSIFICAÇÃO

Conforme pode ser observado na Figura 5.5, nesta avaliação o componente de otimização de pipeline é incrementado com a habilidade de selecionar, além do classificador, a melhor técnica de pré-processamento de dados e pré-processamento de características. Esta implementação do COP foi realizada tendo como base o Auto-Sklearn (Feurer et al., 2019), sendo denominado de ANTE-AS. O Auto-Sklearn é capaz de realizar uma busca eficiente entre 15 algoritmos de classificação, 14 métodos de pré-processamento de dados e 4 técnicas de pré-processamento de características.

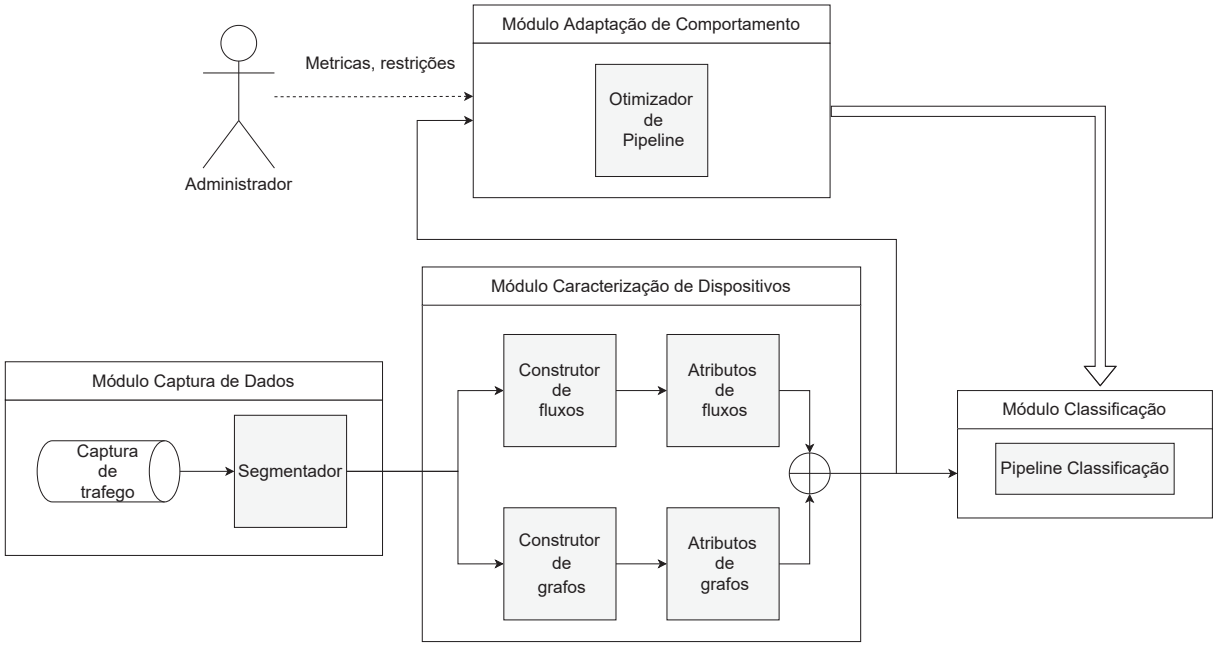


Figura 5.5: Arquitetura do sistema ANTE para a otimização autônoma de um pipeline de classificação

Para a realização de testes foram selecionados os cenários ISOT, CTU-13, CICDDoS, e BOT-IOT. A segmentação das capturas ocorrem como ilustrado na Figura 5.4. Na Tabela 5.10 é ilustrada a configuração de *pipeline* escolhida pelo ANTE-AS. É possível notar que existe uma diversidade de técnicas entre os componentes do *pipeline*. O componente ANTE-AS foi capaz de escolher técnicas de redução de dimensionalidade diferentes para cenários distintos, essa diversidade também pode ser observada na escolha de um classificador. As métricas de performance de cada *pipeline* são exibidas na Tabela 5.11.

5.6 AVALIAÇÃO DA ADAPTABILIDADE DO SISTEMA ANTE

O objetivo desse experimento é avaliar se o sistema é capaz de detectar *drifts* no comportamento no tráfego da rede, principalmente se ocorrerem *drifts* próximos ao início do ataque. É esperado que ao identificar *drifts* no comportamento da rede o sistema possa evitar a degradação do desempenho em relação a correta classificação dos dispositivos através da seleção de uma nova técnica de aprendizado de máquina. A Seção 5.6.1 detalha os testes realizados e a Seção 5.6.4 apresenta os resultados obtidos.

5.6.1 Metodologia

A avaliação do sistema ocorreu a partir da execução do sistema ANTE na captura 51 da base de dados CTU-13. Esta base foi selecionada por sua abrangência de rótulos, onde é possível encontrar os momentos exatos do começo e fim de um ataque e também pela existência de um conjunto diversificado de ataques. Para testar o sistema com diferentes níveis de disponibilidade de rótulos o sistema foi executado no modo semi-supervisionado e supervisionado. Para ambos os experimentos este trabalho utiliza a abordagem *Test-Then-Train* (Gama et al., 2014). Onde sempre que uma nova amostra é obtida esta é usada primeiro para o teste do *pipeline* de classificação, e posteriormente é inserida no *buffer* de treinamento. O conjunto de parâmetros utilizados pelo algoritmo de detecção de *drift* foi escolhido com base na recomendação de (Gözüaçık e Can, 2021), sendo utilizado um *buffer* de quatro minutos e um valor de p igual a 0.4. Como a janela de segmentação tem tamanho de um segundo, o *buffer* do módulo de detecção de *drift* é composto por 240 amostras de atributos de rede. O tamanho máximo do *buffer* de treinamento presente no módulo de otimização de *pipelines* também foi definido como quatro minutos.

5.6.2 Experimento 1

O Experimento 1 consiste em testar o sistema no modo semi-supervisionado. Durante a execução do sistema, após cada ocorrência de *drift* os dados presentes no *buffer* de treinamento são enriquecidos com rótulos de dispositivos normais. Dentre todos os dispositivos normais ativos durante a fase de construção do *pipeline* 50% foram rotulados. A Figura 5.6 apresenta os resultados obtidos neste experimento. Os quadros em cinza (Quadros A, B, C, D e E) indicam os momentos em que o sistema detectou *drifts* nos dados do tráfego de rede. E os quadros pontilhados em vermelho (Quadros 1, 2 e 3) indicam o período (o início e o fim) dos ataques. O Quadro A indica um *drift* no início da captura, pois o sistema está inicializando. É oportuno notar que no início da base, após o primeiro treinamento, os valores da precisão e do recall para os dados normais estão variando entre 90% e 100%. Isso significa que o sistema utilizando o *pipeline* inicial classifica corretamente como dispositivos normais quase todos os dispositivos normais. Como os *bots* ainda não estão ativos, ou seja, ainda não estão realizando o ataque, não é possível calcular a precisão e o *recall* para a classe dos *bots* nos momentos iniciais. Próximo do segundo 1000, o sistema detecta um *drift* (Quadro B). Neste momento, a solução constrói um novo *pipeline*. Após o retreinamento, os resultados da classificação dos dispositivos normais continuam próximos de 100%. O problema do *drift* no Quadro B é que a documentação da base não indica um ataque neste momento. Deste modo, este trabalho trata o *drift* no Quadro B como um falso positivo.

A Figura 5.6 apresenta que, próximo do segundo 6000, um ataque é iniciado (Quadro 1). É importante ressaltar que o *pipeline* construído após o segundo *drift* é ineficaz para detectar os *bots* no começo do ataque. Isso porque os resultados da precisão e do *recall* para a classe dos *bots* após o início do ataque ainda estão zerados. Poucos segundo depois do início do ataque o

sistema identifica uma mudança nos dados do tráfego de rede e constrói um novo *pipeline*. A partir do retreinamento, o sistema continua classificando os não *bots* com altas taxas de acertos. Porém o mais relevante é que o sistema começa a classificar corretamente os *bots*. Visto que, a precisão e o *recall* para a classe dos *bots*, linhas amarelas e marrom respectivamente, estão visíveis no gráfico. Após o *drift* do Quadro C, o *recall* para classe dos *bots* fica praticamente constante em 100%. Isso significa que o sistema consegue identificar todos os *bots*. Porém a precisão varia em níveis baixos, de 40% a 80%. Isso significa que para o sistema acertar os *bots*, ele classifica alguns dispositivos normais como *bots* também. Apesar de estar ativo por menos tempo que o ataque anterior, é possível verificar que logo após o início do ataque não existem resultados para a precisão e para o *recall* da classe dos *bots*. Após detectar o *drift*, o sistema consegue identificar os *bots*, porém com alta taxa de FPs. No Quadro 3 temos o início do terceiro ataque. Neste caso os resultados são ligeiramente diferentes. Durante o início do ataque, antes da detecção do *drift*, o sistema não consegue classificar os *bots* e a precisão para a classe dos dispositivos normais, linha azul, cai bastante em comparação com os resultados dos momentos anteriores. Porém, após a detecção do *drift* e o retreinamento, o sistema volta a classificar corretamente os dispositivos normais. Onde a precisão e o *recall* dos dispositivos normais é próximo de 100%, e a precisão dos *bots* varia entre 35% e 90%.

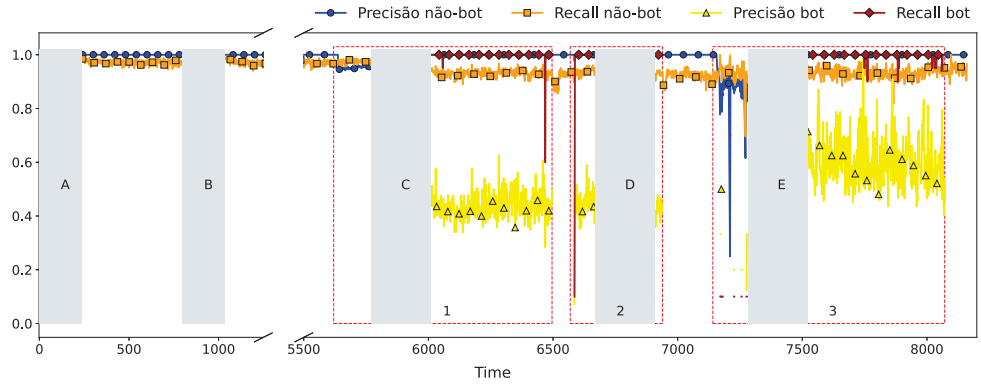


Figura 5.6: Precisão e *recall* ao longo do tempo durante o Experimento 1

5.6.3 Experimento 2

A execução do segundo experimento consiste em testar o poder do sistema ANTE para automatizar a criação de um *pipeline* de classificação utilizando aprendizado de máquina supervisionado para o atual cenário da rede. Como as técnicas supervisionadas necessitam de rótulos das duas classes para o treinamento, este trabalho selecionou aleatoriamente 50% dos rótulos dos *bots* e dos dispositivos normais. Escolhemos 50% dos rótulos para cada classe pois, em cenários reais, a obtenção de dados rotulados é custosa.

A Figura 5.7 apresenta os resultados obtidos durante a execução do Experimento 2. Como o módulo de detecção de *drift* é o mesmo do executado no Experimento 1, o sistema detectou os *drifts* no mesmos momentos identificados no experimento anterior (Quadros A, B, C, D e E). Como o sistema foi projetado para utilizar técnicas supervisionadas apenas quando o usuário informar rótulos de *bots*, o começo da Figura 5.7, entre o Quadro A e o quadro C, não apresenta resultados. Isso acontece pois os resultados seriam iguais aos da Figura 5.6. Como o objetivo deste experimento é avaliar o aprendizado supervisionado e para simplificar a apresentação dos resultados, este trabalho optou por apresentar as métricas apenas após o início do primeiro ataque, pois é onde as diferenças começam.

No começo do primeiro ataque, representado pelo Quadro 1 da Figura 5.7, o sistema faz a construção do primeiro *pipeline* de classificação. Já que esse é o primeiro momento onde é possível obter rótulos de ambas as classes. Assim que o sistema detecta *drift* (Quadro C) o sistema supervisionado começa a acertar quase todas as classificações de ambas as classes. Neste momento, o sistema construiu um novo *pipeline* de classificação. Durante o início do segundo ataque (Quadro 2) o sistema utilizando o *pipeline* começa acertando a maioria das classificações. Porém, conforme o ataque evolui e a mudança do comportamento da rede se aproxima, o sistema apresenta uma queda brusca na precisão dos *bots*. Esse desempenho se mantém mesmo depois do sistema identificar o quarto *drift* (Quadro D), e construir outro *pipeline*. Após o início do do terceiro ataque (Quadro 3) o sistema apresenta instabilidade no desempenho. Isso é demonstrado logo após o início do ataque, onde a precisão para a classe de *bots* aproxima-se dos 40%. Em seguida o sistema identifica um novo *drift* (Quadro E). Após o sistema detectar o quinto *drift* as métricas do sistema ficam próximas de 100%. A precisão para os *bots* é a que mais oscila neste período. Isso significa que o sistema classifica alguns dispositivos normais como *bots*.

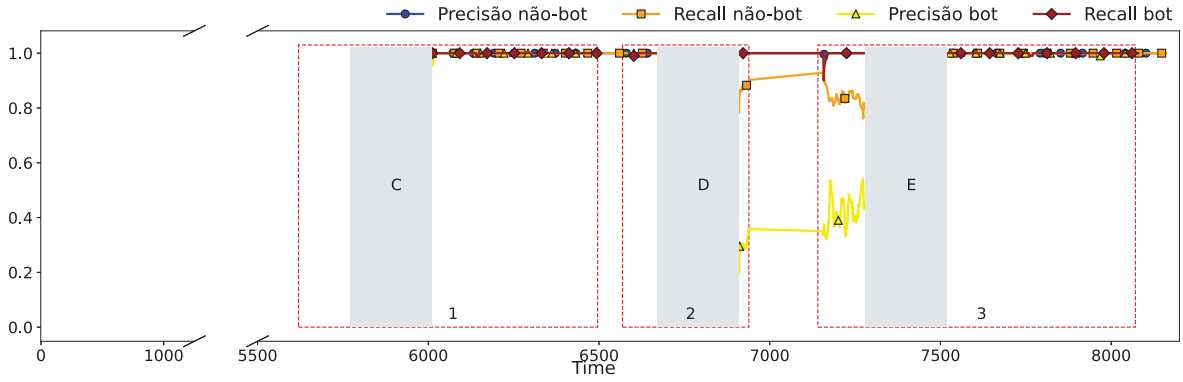


Figura 5.7: Precisão e *recall* ao longo do tempo para durante o Experimento 2

5.6.4 Resultados

Os resultados do Experimento 1 indicam que, em geral, o sistema ANTE utilizando o aprendizado semi-supervisionado é capaz de classificar os dispositivos na rede. Ao longo de toda a base de dados, o sistema consegue classificar corretamente quase todos os dispositivos normais, aqueles que não estão fazem parte da *botnet*. Durante o início dos ataques, o sistema não consegue classificar os *bots*, porém após o retreinamento o sistema consegue fazer esta identificação. Porém para o sistema identificar corretamente os *bots* o sistema classifica alguns dispositivos normais também como *bots*, isso faz com que a precisão fique baixa. O modo supervisionado é superior ao modo semi-supervisionado. Isso ocorre principalmente nos momentos onde ocorrem os ataques. Isso é indicado por uma precisão e *recall* sempre próximas de 100%. Para atingir esse resultado, o sistema seleciona a melhor técnica de aprendizado supervisionado considerando o comportamento da rede. Por fim, é importante ressaltar que geralmente uma precisão e *recall* de 100% é um sinal de *overfitting*, porém durante os testes foram utilizadas duas estrategias distintas para combater o *overfitting*. Além de utilizado a abordagem *Test-Then-Train*, nenhum dispositivo que aparece no treinamento foi utilizado durante os testes. Desse modo, os testes foram realizados com dispositivos completamente desconhecidos. Dessa maneira, o desempenho é resultado da otimização de parâmetros e escolha de uma técnica adequada

5.7 RESUMO

Este capítulo apresentou a avaliação do sistema ANTE, detalhando os cenários de testes, métricas e metodologias. Os resultados das avaliações comprovaram a assertividade do sistema ANTE em detectar dispositivos integrantes de um ataque de *botnet*. Além disso, foi possível observar que a implementação preliminar do módulo de adaptação proposto foi capaz de selecionar técnicas de detecção diferentes para cenários distintos. Isso indica que o sistema ANTE foi capaz de considerar diferentes aspectos de um ataque na escolha da melhor forma de realizar a detecção.

Cenário	Solução	Janela	Acurácia	Precisão	Recall	F1-Score
ISOT	ANTE-AS	Antecipação	100.0%	100.0%	100.0%	100.0%
		Teste	98.46%	95.83%	95.83%	95.83%
	SVM	Antecipação	88.0%	100.0%	57.14%	72.73%
		Teste	90.77%	100.0%	50.0%	66.67%
	Naïve Bayes	Antecipação	72.0%	50.0%	100.0%	66.67%
		Teste	73.85%	40.0%	83.33%	54.05%
	MLP	Antecipação	100.0%	100.0%	100.0%	100.0%
		Teste	95.38%	90.91%	83.33%	86.96%
CTU-13	ANTE-AS	Antecipação	100.0%	100.0%	100.0%	100.0%
		Teste	100.0%	100.0%	100.0%	100.0%
	SVM	Antecipação	92.31%	50.0%	100.0%	66.67%
		Teste	73.91%	68.97%	100.0%	81.63%
	Naïve Bayes	Antecipação	92.31%	0.0%	0.0%	0.0%
		Teste	100.0%	100.0%	100.0%	100.0%
	MLP	Antecipação	100.0%	100.0%	100.0%	100.0%
		Teste	97.1%	95.24%	100.0%	97.56%
CICDDoS	ANTE-AS	Antecipação	100.0%	100.0%	100.0%	100.0%
		Teste	100.0%	100.0%	100.0%	100.0%
	SVM	Antecipação	97.87%	0.0%	0.0%	0.0%
		Teste	98.75%	0.0%	0.0%	0.0%
	Naïve Bayes	Antecipação	97.87%	50.0%	100.0%	66.67%
		Teste	100.0%	100.0%	100.0%	100.0%
	MLP	Antecipação	93.62%	25.0%	100.0%	40.0%
		Teste	99.07%	57.14%	100.0%	72.73%
BoT-IoT	ANTE-AS	Antecipação	100.0%	100.0%	100.0%	100.0%
		Teste	81.27%	56.77%	96.77%	71.56%
	SVM	Antecipação	77.13%	54.4%	100.0%	70.46%
		Teste	31.72%	26.28%	100.0%	41.62%
	Naïve Bayes	Antecipação	64.05%	41.98%	83.33%	55.84%
		Teste	62.63%	38.14%	86.04%	52.85%
	MLP	Antecipação	98.48%	100.0%	94.44%	97.14%
		Teste	70.55%	43.82%	74.39%	55.15%

Tabela 5.11: Avaliação dos *pipelines* de classificação escolhidos pelo ANTE-AS para diferentes cenários de teste.

6 CONSIDERAÇÕES FINAIS

Os ataques realizados por *botnets* têm se destacado, dado à sua diversidade e o potencial para causar danos econômicos, sociais, institucionais e operacionais. No entanto, a detecção de *botnets* é um desafio complexo, devido à grande diversidade de *botnets* existentes e ao constante desenvolvimento de novas variantes mais sofisticadas e evasivas. Além disso, os *botmasters* estão aprimorando suas técnicas de invasão, tornando ainda mais difícil a detecção de dispositivos infectados.

Apesar dessas constantes mudanças e evoluções, as técnicas de detecção de *botnets* ainda não conseguiram acompanhar esse ritmo dinâmico. Isso se deve, em grande parte, ao fato de que essas soluções ainda se baseiam em um contexto estático, onde é assumido que um único tipo de ataque será encontrado ou que o ataque apresenta um comportamento padrão. Esse tipo de abordagem se torna cada vez mais problemático diante da variedade de técnicas de ataque possíveis. Como resultado, muitas vezes ocorre uma discrepância entre a distribuição esperada pela solução e a distribuição de dados observada, o que compromete a eficácia da detecção de *botnets*. Logo, um sistema para a detecção de *botnets* deve ser capaz de tratar ameaças desconhecidas com comportamento dinâmico.

Desta forma foi proposto o ANTE, um sistema com o objetivo de detectar mudanças no comportamento de rede e escolher a melhor técnica de detecção em tempo real. O sistema ANTE é composto por um conjunto de módulos independentes que interagem entre si para formar um sistema integrado capaz de se adaptar a mudanças no comportamento da rede. O uso de diferentes técnicas de aprendizado de máquina na implementação desses módulos habilita o sistema ANTE a construir automaticamente a melhor forma de detecção para diferentes tipos de *botnets* em tempo de execução sem a necessidade de intervenção humana.

Para avaliar a eficácia do sistema na detecção de tipos de *botnets* distintas, foi utilizado um conjunto diversificado de cenários de ataques. Além disso, as avaliações conduzidas consideram cenários dinâmicos, com mudanças no comportamento da rede. Os resultados das avaliações comprovaram a eficácia do ANTE em detectar *botnets*. O sistema foi capaz de manter uma média de precisão e *recall* acima de 90% durante o seu período de execução. Além disso, foi possível observar que a implementação do módulo de adaptação proposto foi capaz de selecionar técnicas de detecção diferentes para cenários distintos. Isso indica que o sistema ANTE foi capaz de considerar diferentes aspectos de um ataque na escolha da melhor forma de realizar a detecção.

6.1 DIREÇÕES FUTURAS

Um aspecto importante para lidar com novas ameaças é a necessidade de manter um conjunto expressivo de técnicas de detecção. Para realizar a busca e instanciação dessas técnicas de maneira eficiente, pretende-se equipar o sistema ANTE com uma base de conhecimento formada pela combinação de contextos encontrados durante o seu ciclo de vida e soluções empregadas. O objetivo por trás dessa decisão é criar um mecanismo capaz de aprender como aplicar a melhor técnica de detecção baseada em memórias passadas.

6.2 PUBLICAÇÕES

Esta seção apresenta a lista de publicações ou submissões realizadas.

- NEIRA, Anderson Bergamini; MEDEIROS, Alex ; NOGUEIRA, Michele. **Identificação Antecipada de Botnets por Aprendizagem de Máquina**. Em: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS (SBRC), 2020, Rio de Janeiro.

- NEIRA, Anderson Bergamini; MEDEIROS, Alex ; NOGUEIRA, Michele. **Early botnet detection for theinternet and the internet of things by autonomous machine learning**. Em 2020 16th International Conference on Mobility, Sensing and Networking (MSN). IEEE.

- MEDEIROS, Alex; NEIRA, Anderson Bergamini; NOGUEIRA, Michele. **Autonomous Machine Learning for EarlyBot Detection in the Internet of Things**. Em Digital Communications and Networks (DCN)

- ARAUJO, ALEX MEDEIROS ; NEIRA, ANDERSON BERGAMINI; NOGUEIRA, MICHELE . **Lifelong Autonomous Botnet Detection**. Globecom, 2022

REFERÊNCIAS

Alazzam, H., Alsmady, A. e Shorman, A. A. (2019). Supervised detection of IoT botnet attacks. Em *Proceedings of the Second International Conference on Data Science, E-Learning and Information Systems*, páginas 1–6.

Alenazi, A., Traore, I., Ganame, K. e Woungang, I. (2017). Holistic model for HTTP botnet detection based on DNS traffic analysis. Em *Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*, páginas 1–18. Springer International Publishing.

Almeida, P. R., Oliveira, L. S., Britto Jr, A. S. e Sabourin, R. (2018). Adapting dynamic classifier selection for concept drift. *Expert Systems with Applications*, 104:67–85.

Amini, P., Araghizadeh, M. A. e Azmi, R. (2015). A survey on botnet: Classification, detection and defense. Em *2015 International Electronics Symposium (IES)*, páginas 233–238.

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J. A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K. e Zhou, Y. (2017). Understanding the mirai botnet. Em *USENIX Security Symposium (USENIX Security 17)*, páginas 1093–1110, Vancouver, BC. USENIX Association.

Arnaldo, I. e Veeramachaneni, K. (2019). The holy grail of "systems for machine learning". *ACM SIGKDD Explorations Newsletter*, 21(2):39–47.

Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L. e Rieck, K. (2020). Dos and don'ts of machine learning in computer security.

Asha, S., Harsha, T. e Soniya, B. (2016). Analysis on botnet detection techniques. Em *2016 International Conference on Research Advances in Integrated Navigation Systems (RAINS)*, páginas 1–4.

Bace, R. e Mell, P. (2001). Intrusion detection systems.

Bagnall, A. e Cawley, G. C. (2017). On the use of default parameter settings in the empirical evaluation of classification algorithms.

Bansal, A. e Mahapatra, S. (2017). A comparative analysis of machine learning techniques for botnet detection. Em *Proceedings of the International Conference on Security of Information and Networks*, página 91–98, New York, NY, USA. Association for Computing Machinery.

Barbier, J., Buckalew, L., Loucks, J., Moriarty, R., O'Connell, K. e Riegel, M. (2016). Cybersecurity as a growth advantage. `https://www.newhorizons.com/Portals/278/Downloads/Cybersecurity-as-a-Growth-Advantage-Cisco.pdf`. Último acesso em Nov/2021.

Barford, P. e Yegneswaran, V. (2007). An inside look at botnets. Em *Advances in Information Security*, páginas 171–191. Springer US.

Bergstra, J. e Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2).

Bertino, E. e Islam, N. (2017). Botnets and Internet of Things security. *Computer*, 50(2):76–79.

Bick, A., Blandin, A. e Mertens, K. (2021). Work from home before and after the COVID-19 outbreak.

Bischl, B., Kerschke, P., Kotthoff, L., Lindauer, M., Malitsky, Y., Fréchette, A., Hoos, H., Hutter, F., Leyton-Brown, K., Tierney, K. e Vanschoren, J. (2016). Aslib: A benchmark library for algorithm selection. *Artificial Intelligence*, 237:41–58.

Bottazzi, G. e Me, G. (2014). The botnet revenue model. Em *Proceedings of the International Conference on Security of Information and Networks*, páginas 459–465.

Brunt, R., Pandey, P. e McCoy, D. (2017). Booted: An analysis of a payment intervention on a DDoS-for-hire service. Em *Workshop on the Economics of Information Security*, páginas 06–26.

Celik, B. e Vanschoren, J. (2021). Adaptation strategies for automated machine learning on evolving data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9):3067–3078.

Chang, W., Mohaisen, A., Wang, A. e Chen, S. (2018). Understanding adversarial strategies from bot recruitment to scheduling. Em *Security and Privacy in Communication Networks*, páginas 397–417, Cham. Springer International Publishing.

Chowdhury, S., Khanzadeh, M., Akula, R., Zhang, F., Zhang, S., Medal, H., Marufuzzaman, M. e Bian, L. (2017). Botnet detection using graph-based feature clustering. *Journal of Big Data*, 4(1):1–23.

Cid-Fuentes, J. Á., Szabo, C. e Falkner, K. (2018). An adaptive framework for the detection of novel botnets. *Computers & Security*, 79:148–161.

Claesen, M. e Moor, B. D. (2015). Hyperparameter search in machine learning.

Cucinotta, D. e Vanelli, M. (2020). Who declares COVID-19 a pandemic. *Acta Bio Medica: Atenei Parmensis*, 91(1):157.

Daya, A. A., Salahuddin, M. A., Limam, N. e Boutaba, R. (2019). A graph-based machine learning approach for bot detection. Em *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, páginas 144–152.

Derakhshan, F. e Ashrafnejad, M. (2020). *The Risk of Botnets in Cyber Physical Systems*. Springer International Publishing, Cham.

Ebit (2021). 44ª ed. webshoppers (versão free). `https://company.ebit.com.br/webshoppers/webshoppersfree`. Último acesso em Nov/2021.

Elkan, C. e Noto, K. (2008). Learning classifiers from only positive and unlabeled data. Em *SIGKDD*, páginas 213–220.

Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M. e Smola, A. (2020). Autogluon-tabular: Robust and accurate AutoML for structured data. *arXiv preprint arXiv:2003.06505*.

F-Secure (2012). Zeroaccess the most profitable botnet malware in the wild. `https://archive.f-secure.com/weblog/archives/ZeroAccess.pdf`. Último acesso em Nov/2021.

Fedynyshyn, G., Chuah, M. C. e Tan, G. (2011). Detection and classification of different botnet C&C channels. Em *International Conference on Autonomic and Trusted Computing*, páginas 228–242. Springer.

Feldmann, A., Gasser, O., Lichtblau, F., Pujol, E., Poese, I., Dietzel, C., Wagner, D., Wichtlhuber, M., Tapiador, J., Vallina-Rodriguez, N., Hohlfeld, O. e Smaragdakis, G. (2021). Implications of the COVID-19 pandemic on the internet traffic. Em *Broadband Coverage in Germany; 15th ITG-Symposium*, páginas 1–5.

Feurer, M. e Hutter, F. (2019). *Hyperparameter Optimization*, páginas 3–33. Springer International Publishing.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M. e Hutter, F. (2019). Auto-sklearn: efficient and robust automated machine learning. Em *Automated Machine Learning*, páginas 113–134. Springer, Cham.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M. e Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37.

García, S., Grill, M., Stiborek, J. e Zunino, A. (2014). An empirical comparison of botnet detection methods. *Computers & Security*, 45:100 – 123.

Goebel, J. e Holz, T. (2007). Rishi: Identify bot contaminated hosts by IRC nickname evaluation. *HotBots*, 7(8-8):192.

Gözüaçık, Ö., Büyükçakır, A., Bonab, H. e Can, F. (2019). Unsupervised concept drift detection with a discriminative classifier. Em *Proceedings of the ACM International Conference on Information and Knowledge Management*, páginas 2365–2368.

Gözüaçık, Ö. e Can, F. (2021). Concept learning using one-class classifiers for implicit drift detection in evolving data streams. *Artificial Intelligence Review*, 54(5):3725–3747.

Gu, G., Porras, P. A., Yegneswaran, V., Fong, M. W. e Lee, W. (2007). Bothunter: Detecting malware infection through IDS-driven dialog correlation. Em *USENIX Security Symposium*, volume 7, páginas 1–16.

Gu, G., Yegneswaran, V., Porras, P., Stoll, J. e Lee, W. (2009). Active botnet probing to identify obscure command and control channels. Em *2009 Annual Computer Security Applications Conference*, páginas 241–253.

Gupta, B. e Badve, O. P. (2017). Taxonomy of DoS and DDoS attacks and desirable defense mechanism in a cloud computing environment. *Neural Comp.and Appl.*, 28(12):3655–3682.

Hassija, V., Chamola, V., Saxena, V., Jain, D., Goyal, P. e Sikdar, B. (2019). A survey on IoT security: Application areas, security threats, and solution architectures. *IEEE Access*, 7:82721–82743.

He, X., Zhao, K. e Chu, X. (2021). AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212.

He, Z., Zhang, T. e Lee, R. B. (2017). Machine learning based DDoS attack detection from source side in cloud. Em *IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)*, páginas 114–120, New York, NY, USA. IEEE.

Heydari, B., Yajam, H., Akhaee, M. A. e Salehkalaibar, S. (2017). Utilizing features of aggregated flows to identify botnet network traffic. Em *International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC)*, páginas 25–30. IEEE.

Holt, T. J. e Bossler, A. M. (2020). *The Palgrave Handbook of International Cybercrime and Cyberdeviance*. Springer.

Hoos, H. H. (2012). *Automated Algorithm Configuration and Parameter Tuning*, páginas 37–71. Springer Berlin Heidelberg.

Imperva (2021). Bad bot report 2021. `https://www.imperva.com/blog/bad-bot-report-2021-the-pandemic-of-the-internet/`. Último acesso em Nov/2021.

Indre, I. e Lemnaru, C. (2016). Detection and prevention system against cyber attacks and botnet malware for information systems and Internet of Things. Em *IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, páginas 175–182. IEEE.

Jyoti, N. e Behal, S. (2021). A meta-evaluation of machine learning techniques for detection of DDoS attacks. Em *International Conference on Computing for Sustainable Global Development (INDIACom)*, páginas 522–526, New Delhi, India. IEEE.

Kalige, E., Burkey, D. e Director, I. (2012). A case study of eurograbber: How 36 million euros was stolen via malware. *Versafe (White paper)*, 35.

Kambourakis, G., Anagnostopoulos, M., Meng, W. e Zhou, P. (2019). *Botnets: Architectures, Countermeasures, and Challenges*. CRC Press.

Kambourakis, G., Kolias, C. e Stavrou, A. (2017). The mirai botnet and the IoT zombie armies. Em *Proceedings of the IEEE Military Communications Conference (MILCOM)*, páginas 267–272.

Kedziora, D. J., Musial, K. e Gabrys, B. (2020). Autonoml: Towards an integrated framework for autonomous machine learning. *arXiv e-prints*, páginas arXiv–2012.

Khan, W. Z., Khan, M. K., Muhaya, F. T. B., Aalsalem, M. Y. e Chao, H.-C. (2015). A comprehensive study of email spam botnet detection. *IEEE Communications Surveys & Tutorials*, 17(4):2271–2295.

Khattak, S., Ramay, N. R., Khan, K. R., Syed, A. A. e Khayam, S. A. (2014). A taxonomy of botnet behavior, detection, and defense. *IEEE Communications Surveys Tutorials*, 16(2):898–924.

Klaar, H. (2013). *Botnets*. Springer, London New York.

Komer, B., Bergstra, J. e Eliasmith, C. (2019). *Hyperopt-Sklearn*, páginas 97–111. Springer International Publishing, Cham.

Kondo, S. e Sato, N. (2007). Botnet traffic detection techniques by C&C session classification using SVM. Em *Advances in Information and Computer Security*, páginas 91–104. Springer Berlin Heidelberg.

Koroniotis, N., Moustafa, N. e Sitnikova, E. (2019). Forensics and deep learning mechanisms for botnets in Internet of Things: A survey of challenges and solutions. *IEEE Access*, 7:61764–61785.

Kozik, R. e Choraś, M. (2017). Pattern extraction algorithm for netflow-based botnet activities detection. *Security and Communication Networks*, 2017.

LeDell, E. e Poirier, S. (2020). H2o AutoML: Scalable automatic machine learning. Em *Proceedings of the AutoML Workshop at ICML*, volume 2020.

Lin, W. e Lee, D. (2012). Traceback attacks in cloud – pebbletrace botnet. Em *2012 32nd International Conference on Distributed Computing Systems Workshops*, páginas 417–426.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J. e Zhang, G. (2018). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363.

Lu, W., Tavallaee, M. e Ghorbani, A. A. (2009). Automatic discovery of botnet communities on large-scale communication networks. Em *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, página 1–10. Association for Computing Machinery.

Luo, G. (2016). A review of automatic selection methods for machine learning algorithms and hyper-parameter values. *Network Modeling Analysis in Health Informatics and Bioinformatics*, 5(1):1–16.

McLynn, F. (2013). *Crime and punishment in eighteenth century England*. Routledge.

Miller, S. e Busby-Earle, C. (2016). The role of machine learning in botnet detection. Em *2016 11th international conference for Internet technology and secured transactions (icitst)*, páginas 359–364. IEEE.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill, New York.

Mizoguchi, S., Takemori, K., Miyake, Y., Hori, Y. e Sakurai, K. (2011). Traceback framework against botmaster by sharing network communication pattern information. Em *2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, páginas 639–644.

Monteiro, S. D. (2007). O ciberespaço: o termo, a definição e o conceito. *DataGramaZero-Revista de Ciência da Informação*, 8(3):1–21.

Moreira, D. A. B., Marques, H. P., Costa, W. L., Celestino, J., Gomes, R. L. e Nogueira, M. (2021). Anomaly detection in smart environments using AI over fog and cloud computing. Em *2021 IEEE 18th Annual Consumer Communications Networking Conference (CCNC)*, páginas 1–2.

Murphy, K. P. (2022). *Probabilistic Machine Learning: An introduction*. MIT Press.

NIST (2018). Framework for improving critical infrastructure cybersecurity. Relatório técnico, U.S. Department of Commerce.

Nooribakhsh, M. e Mollamotalebi, M. (2020). A review on statistical approaches for anomaly detection in DDoS attacks. *Information Security Journal: A Global Perspective*, 29(3):118–133.

Paxson, V. (1999). Bro: A system for detecting network intruders in real-time. *Computer networks*, 31(23-24):2435–2463.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Pelloso, M., Vergutz, A., Santos, A. e Nogueira, M. (2018). A self-adaptable system for DDoS attack prediction based on the metastability theory. Em *2018 IEEE Global Communications Conference (GLOBECOM)*, páginas 1–6.

Putman, C., Nieuwenhuis, L. J. et al. (2018). Business model of a botnet. Em *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, páginas 441–445. IEEE.

PwC (2021). Power shifts: Altering the dynamics of the e&m industry. `https://www.pwc.com/gx/en/entertainment-media/outlook-2021/perspectives-2021-2025.pdf`. Último acesso em Nov/2021.

Radain, D., Almalki, S., Alsaadi, H. e Salama, S. (2021). A review on defense mechanisms against distributed denial of service (DDoS) attacks on cloud computing. Em *2021 International Conference of Women in Data Science at Taif University (WiDSTaif)*, páginas 1–6. IEEE.

Raghava, N., Sahgal, D. e Chandna, S. (2012). Classification of botnet detection based on botnet architechture. Em *2012 International Conference on Communication Systems and Network Technologies*, páginas 569–572. IEEE.

Rahal, B. M., Santos, A. e Nogueira, M. (2020). A distributed architecture for DDoS prediction and bot detection. *IEEE Access*, 8:159756–159772.

Ramsbrock, D., Wang, X. e Jiang, X. (2008). A first step towards live botmaster traceback. Em *Recent Advances in Intrusion Detection*, páginas 59–77. Springer Berlin Heidelberg.

Rice, J. R. (1976). The algorithm selection problem. Em *Advances in computers*, volume 15, páginas 65–118. Elsevier.

Rodríguez-Gómez, R., Maciá-Fernández, G. e Garcia-Teodoro, P. (2011). Analysis of botnets through life-cycle. Em *Proceedings of the International Conference on Security and Cryptography*, páginas 257–262. IEEE.

Roesch, M. et al. (1999). Snort: Lightweight intrusion detection for networks. Em *Lisa*, volume 99, páginas 229–238.

Russell, S. e Norvig, P. (2020). *Artificial Intelligence: A Modern Approach (Pearson Series in Artifical Intelligence)*. Pearson.

Santos, A. A., Nogueira, M. e Moura, J. M. F. (2017). A stochastic adaptive model to explore mobile botnet dynamics. *IEEE Communications Letters*, 21(4):753–756.

Schwab, K. (2016). *The fourth industrial revolution*. World Economic Forum, Geneva, Switzerland.

Schwengber, B. H., Vergütz, A., Prates, N. G. e Nogueira, M. (2020). A method aware of concept drift for online botnet detection. Em *GLOBECOM 2020-2020 IEEE Global Communications Conference*, páginas 1–6. IEEE.

Sharafaldin, I., Lashkari, A. H., Hakak, S. e Ghorbani, A. A. (2019). Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. Em *2019 International Carnahan Conference on Security Technology (ICCST)*, páginas 1–8.

Shinan, K., Alsubhi, K., Alzahrani, A. e Ashraf, M. U. (2021). Machine learning-based botnet detection in software-defined network: A systematic review. *Symmetry*, 13(5).

Silva, S. S., Silva, R. M., Pinto, R. C. e Salles, R. M. (2013). Botnets: A survey. *Computer Networks*, 57(2):378–403.

Somani, G., Gaur, M. S., Sanghi, D., Conti, M. e Buyya, R. (2017). DDoS attacks in cloud computing: Issues, taxonomy, and future directions. *Computer Communications*, 107:30–48.

Srinivasan, K., Mubarakali, A., Alqahtani, A. S. e Kumar, A. D. (2019). A survey on the impact of DDoS attacks in cloud computing: Prevention, detection and mitigation techniques. Em *Intelligent Communication Technologies and Virtual Mobile Networks*, páginas 252–270. Springer.

Stevanovic, M. e Pedersen, J. M. (2015). An analysis of network traffic classification for botnet detection. Em *2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, páginas 1–8. IEEE.

Stevanovic, M. e Pedersen, J. M. (2016). On the use of machine learning for identifying botnet network traffic. *Journal of Cyber Security and Mobility*, 4(2 & 3).

Thornton, C., Hutter, F., Hoos, H. H. e Leyton-Brown, K. (2013). Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. Em *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, página 847–855. Association for Computing Machinery.

Tsymbal, A. (2004). The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106(2):58.

Tuan, T. A., Long, H. V., Son, L. H., Kumar, R., Priyadarshini, I. e Son, N. T. K. (2020). Performance evaluation of botnet DDoS attack detection using machine learning. *Evolutionary Intelligence*, 13(2):283–294.

Villamarin-Salomon, R. e Brustoloni, J. C. (2008). Identifying botnets using anomaly detection techniques applied to DNS traffic. Em *2008 5th IEEE Consumer Communications and Networking Conference*, páginas 476–481.

Vu, S. N. T., Stege, M., El-Habr, P. I., Bang, J. e Dragoni, N. (2021). A survey on botnets: Incentives, evolution, detection and current trends. *Future Internet*, 13(8):198.

Vuong, S. T. e Alam, M. S. (2011). Advanced methods for botnet intrusion detection systems.

Wang, Z., Tian, M. e Jia, C. (2017). An active and dynamic botnet detection approach to track hidden concept drift. Em *International Conference on Information and Communications Security*, páginas 646–660. Springer.

Waring, J., Lindvall, C. e Umeton, R. (2020). Automated machine learning: Review of the state-of-the-art and opportunities for healthcare. *Artificial Intelligence in Medicine*, 104.

Yang, L. e Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316.

Zeidanloo, H. R., Shooshtari, M. J. Z., Amoli, P. V., Safari, M. e Zamani, M. (2010). A taxonomy of botnet detection techniques. Em *2010 3rd International Conference on Computer Science and Information Technology*, volume 2, páginas 158–162.

Zhang, J., Perdisci, R., Lee, W., Sarfraz, U. e Luo, X. (2011). Detecting stealthy P2P botnets using statistical traffic fingerprints. Em *2011 IEEE/IFIP 41st International Conference on Dependable Systems Networks (DSN)*, páginas 121–132.

Zhao, D., Traore, I., Sayed, B., Lu, W., Saad, S., Ghorbani, A. e Garant, D. (2013). Botnet detection based on traffic behavior analysis and flow intervals. *computers & security*, 39:2–16.