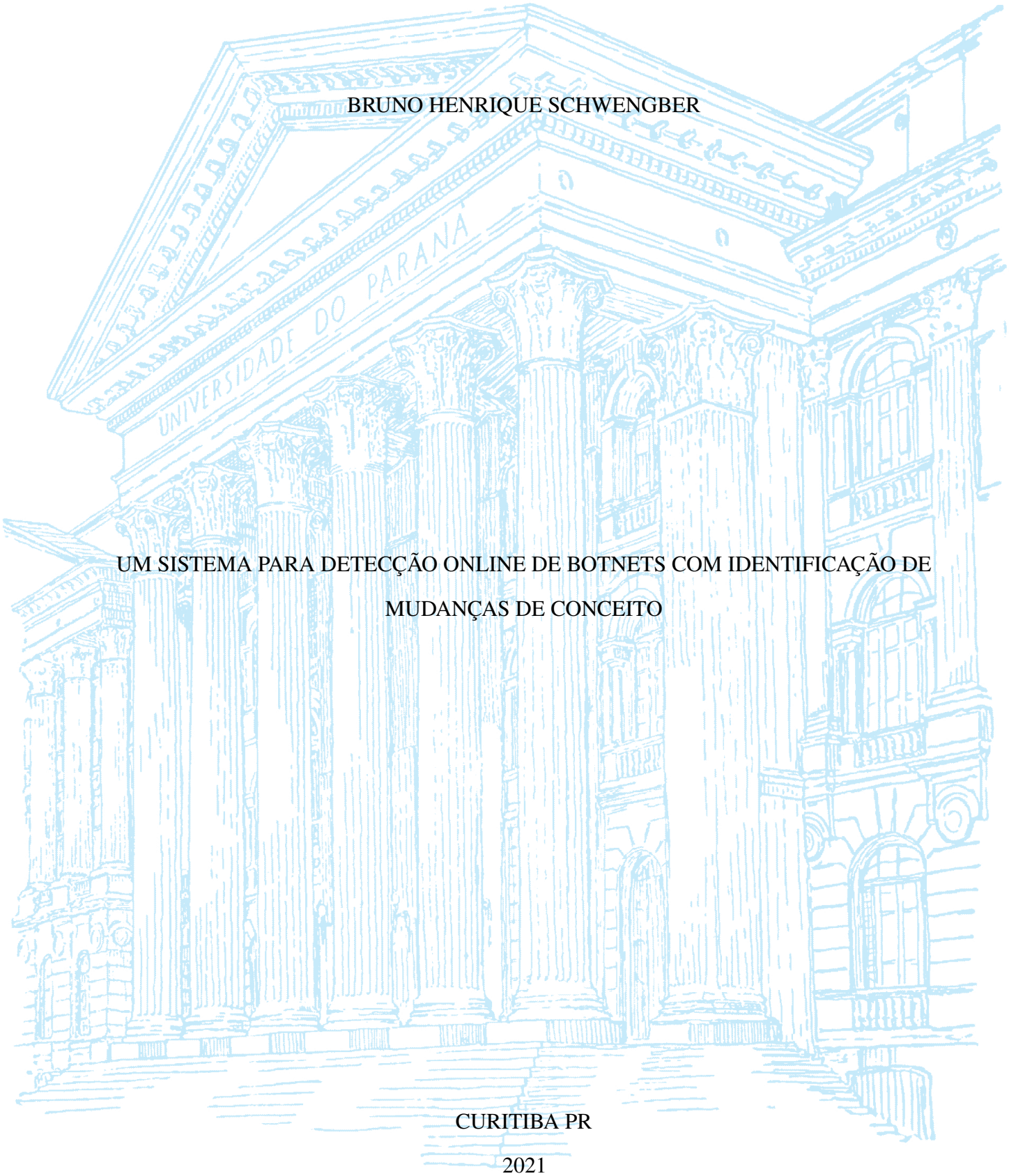
UNIVERSIDADE FEDERAL DO PARANÁ

BRUNO HENRIQUE SCHWENGBER



UM SISTEMA PARA DETECÇÃO ONLINE DE BOTNETS COM IDENTIFICAÇÃO DE MUDANÇAS DE CONCEITO

CURITIBA PR

2021

BRUNO HENRIQUE SCHWENGBER

UM SISTEMA PARA DETECÇÃO ONLINE DE BOTNETS COM IDENTIFICAÇÃO DE MUDANÇAS DE CONCEITO

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.

Orientador: Profa. Dra. Michele Nogueira Lima.

CURITIBA PR

2021

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

S414s Schwengber, Bruno Henrique
Um sistema para detecção *online* de *botnets* com identificação de mudanças de conceito [recurso eletrônico] / Bruno Henrique Schwengber – Curitiba, 2021.

Dissertação - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-graduação em Informática.

Orientadora: Profa. Dra. Michele Nogueira Lima

1. Tecnologia da informação – Sistemas de segurança. 2. Redes de computação – Segurança. I. Universidade Federal do Paraná. II. Lima, Michele Nogueira. III. Título.

CDD: 005.8

Bibliotecária: Roseny Rivelini Morciani CRB-9/1585

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **BRUNO HENRIQUE SCHWENGBER** intitulada: **Um Sistema para Detecção Online de Botnets com Identificação de Mudanças de Conceito**, 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, 26 de Março de 2021.

Assinatura Eletrônica
26/03/2021 17:34:54.0
MICHELE NOGUEIRA LIMA
Presidente da Banca Examinadora (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
26/03/2021 16:06:51.0
ALDRI LUIZ DOS SANTOS
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
26/03/2021 18:58:23.0
DANIEL MACEDO BATISTA
Avaliador Externo (UNIVERSIDADE DE SÃO PAULO - USP)

"Seja a mudança que você quer ver no mundo" - Mahatma Gandhi

AGRADECIMENTOS

A minha família Claudino Roque Schwengber, Salete Miriam Flach Schwengber e Rafael Augusto Schwengber pelo amor, incentivo e apoio incondicional. A minha orientadora Dra. Michele Nogueira Lima pelo suporte, correções, apontamentos e incentivos para a produção e formação acadêmica. A minha banca avaliadora pelas correções e dicas para melhorar o desenvolvimento desta dissertação. Aos meus amigos pela compreensão e paciência em todas as nossas conversas, principalmente Nelson Gonçalves Prates Junior e Andressa Vergütz. A universidade e seu corpo docente, que oportunizaram meu aprendizado. E a todos que direta ou indiretamente fizeram parte da minha formação, meu muito obrigado.

RESUMO

A Internet disponibiliza serviços de rede e seu crescimento vem sendo intensificado pela ascensão do uso de dispositivos da IoT. O aperfeiçoamento na infraestrutura de comunicação e a heterogeneidade dos dispositivos têm potencializado os problemas de segurança em redes. Um dos problemas consiste nas *botnets* (i.e., redes de dispositivos infectados), as quais possuem a capacidade de gerar ataques maciços por meio de dispositivos infectados (*bots*). Os ataques gerados por *botnet* são variados, como negação de serviços, *phishing* e *spam*. De forma geral, os ataques afetam os usuários, os serviços ou as infraestruturas de rede. Vários trabalhos na literatura tratam a detecção de *botnets*. Uma vertente desses trabalhos detecta *botnets* por meio do uso de aprendizado de máquina, onde classificadores são treinados com o comportamento considerado normal da rede e dessa forma conseguem identificar anomalias no comportamento do tráfego. O treinamento desses classificadores assume um estado pré-determinado da rede sem dispositivos infectados. Isto torna inviável o uso dessas abordagens dependendo do tamanho da rede e do comportamento geral dos dispositivos, como a entrada e a saída constante de dispositivos. Outra vertente utiliza o cálculo de entropia ou mesmo de coeficientes de correlação para diferenciar o tráfego de ataque e o tráfego normal. Este trabalho apresenta TRUSTED, um sistema para a detecção *online* não supervisionada de *botnets*, que identifica os dispositivos infectados de forma antecipada ou no momento do ataque. Através da detecção de mudança de conceitos e um algoritmo de *clustering*, o sistema provê um aprendizado não supervisionado e identifica dispositivos infectados que estão trafegando na rede. Isso implica em dizer que o sistema não necessita de conhecimento prévio das características do fluxo da rede, de assinaturas de ataque ou de treinamento prévio de modelos de classificação para identificação das *botnets*. A avaliação do sistema TRUSTED segue uma abordagem orientada a traços, utilizando as bases de dados Botnet 2014 e CTU-13. Ambas as bases contêm traços de rede com tráfego benigno e tráfego maligno rotulados. Dessa forma, a partir dessas bases são extraídos valores das características dos fluxos de redes para as avaliações *offline* e *online*. A partir do conjunto de valores extraídos é feita a construção de um bloco de dados, que contempla o conjunto de valores mais recentes obtidos na rede para análise. Este serve de entrada para a identificação de mudanças de conceito e a detecção de dispositivos infectados. Os resultados mostram a viabilidade do sistema por meio da avaliação empírica das variáveis do sistema e da comparação com as estratégias FIXA e ARF (supervisionada). Além disso, entre todos os cenários avaliados, os resultados de acurácia variam entre 75% e 98%. Comparando estes resultados com as estratégias FIXA e ARF, o sistema TRUSTED superou a estratégia FIXA em todos os resultados, contudo a estratégia ARF obteve melhores resultados na avaliação *offline*. Na avaliação *online* o sistema TRUSTED também superou a estratégia ARF na maioria dos cenários.

Palavras-chave: Mudança de conceito. Detecção de botnet. Ataques. Detecção *online*.

ABSTRACT

The Internet provides network services and its growth is intensified by the rise in the use of IoT devices. The improvement in the communication infrastructure and the heterogeneity of the devices have potencialized network security issues. One of the greatest issue lies in botnets, which can generate massive attacks using a network of infected devices. The attacks generated by botnets are diverse, such as the denial of service, phishing, and spam. In general, attacks affect users, services or network infrastructure. Several works in the literature deal with botnet detection. Some of them detect botnets through the use of machine learning, where classifiers are trained with normal behavior of the network and thus can identify anomalies in traffic behavior. The training of these classifiers lies in the main assumption that the state of the network is without infected devices. Hence, these approaches are not feasible depending on the size of the network and the general behavior of devices, such as constant input and output of devices. Other approaches use entropy calculation or even correlation coefficients to differentiate between attack and normal traffic. Thus, this work presents TRUSTED, a system for unsupervised online botnet detection, which identifies infected devices from early or during the time of the attack. Using concept drift and a clustering algorithm, the system provides unsupervised learning and identifies infected devices in the network. This implies that the system does not require prior knowledge of the network flow characteristics, attack signatures or prior training of classification models to identify the botnets. The evaluation of the TRUSTED system follows a trace-driven approach, using the Botnet 2014 and CTU-13 datasets. Both datasets contain network traces with labeled benign and malignant traffic. Thus, from these datasets, the values of network flow features are extracted for the offline and online evaluations. From the set of extracted values, the system constructs a data block, which comprises the most recent set of values obtained in the network for analysis. This block works as input for the concept drift identification and the detection of infected devices. Results show the feasibility of the system through the empirical evaluation of the system variables and the comparison with the FIXED and ARF (supervised) strategies. Also, the accuracy results vary between 75 % and 98 % among all the scenarios evaluated. Comparing these results with the FIXED and ARF strategies the TRUSTED system surpassed the FIXED strategy in all results. However, the ARF strategy has achieved better results in the offline evaluation. In online evaluation, the TRUSTED system has also surpassed the ARF strategy in most scenarios.

Keywords: Concept drift. Botnet detection. Attacks. Online detection

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

ADWIN	*Adaptive Windowing* Janelamento Dinâmico
ARF	*Adaptive Random Forest* Árvores Aleatórias Adaptativas
AUC	*Area Under Curve* Área Abaixo da Curva
C&C	*Command and Control* Comando e Controle
CTU	*Czech Technical University* Universidade Técnica Tcheca
D3	*Discriminative Drift Detector* Método de Detecção de Mudança de Conceito Discriminativa
D3N	*Non-linear Discriminative Drift Detector* Método de Detecção de Mudança de Conceito Discriminativa Não Linear
DDM	*Drift Detector Method* Método de Detecção de Mudança de Conceito
DDoS	*Distributed Denial of Service* Negação de Serviço Distribuída
DNS	*Domain Name Service* Serviço de Nomes e Domínios
DoS	*Denial of Service* Negação de Serviço
EDDM	*Early Drift Detector Method* Método de Deteção de Mudança de Conceito Antecipada
FN	Falso Negativo
FP	Falso Positivo
FIXWIN	*Fixed Windowing* Janelamento Fixo
HAC	*Hierarchical Agglomerative Clustering* Clusterizador Hierárquico Aglomerativo
HDDDM	*Hellinger Distance Drift Detection Methodology* Metodologia de Detecção de Mudanças de Conceito por Distância de Hellinger
HT	*Hoeffding Tree*

	Árvore de Hoeffding
HTTP	*Hypertext Transfer Protocol* Protocolo de Transferência de Hyper Texto
IDS	*Intrusion Detection System* Sistema de Detecção de Intrusos
IoT	*Internet of Things* Internet das Coisas
IP	*Internet Protocol* Protocolo de Internet
MD3	*Margin Density Drift Detection* Detecção de Mudanças de Conceito por Densidade de Margem
PHT	*Page-Hinckley Teste* Teste de Page-Hinckley
P2P	*Peer-to-Peer* Dispositivo para Dispositivo
PCA	*Principal Component Analysis* Análise de Componentes Principais
TRUSTED	*an on-line sysTem foR UnSupervised boTnEt Detection* Um Sistema Online Não Supervisionado para a Detecção de Botnets
SVM	*Support Vector Machine* Máquina de Vetores de Suporte
VN	Verdadeiro Negativo
VP	Verdadeiro Positivo

LISTA DE SÍMBOLOS

β	beta, limiar de desempenho para detecção da mudança de conceito
δ	delta, quantidade mínima de instâncias para detecção de mudança de conceito
σ	sigma, quantidade de novas instâncias para detecção de mudança de conceito
γ	gama, distância máxima entre duas instâncias para agrupar
p	precisão
r	*recall*

SUMÁRIO

1 INTRODUÇÃO

O aperfeiçoamento na infraestrutura de comunicação, o crescente uso das aplicações em redes de computadores e a heterogeneidade dos dispositivos conectados têm gerado um volume maior de vulnerabilidades de segurança de redes. Com a expansão do uso da Internet, a complexidade de configuração e de gerência dos recursos aumenta (Ammar et al., 2018). Além disso, o uso da Internet está sendo impulsionado com o uso dos dispositivos da Internet das Coisas (do inglês, *Internet of Things - IoT*). Estes dispositivos possuem diversas vulnerabilidades de segurança, como falhas no acesso e na autenticação, facilidade de descoberta na rede e baixo poder de processamento para detecção de intrusos (Neshenko et al., 2019). Assim, com a grande quantidade de dispositivos, as consequências dos ataques são intensificadas (EVEO, 2018).

Existem várias formas de ataques, por exemplo, ataques de negação de serviços distribuídos (do inglês, *Distributed Denial of Service* - DDoS), *trojan horse*, *phishing* e *worms* (SecurityTrails, 2018). Estes ataques são considerados agressivos por utilizar milhares de equipamentos para atacar uma única vítima (ex., um usuário, servidor ou infraestrutura de rede). Para alcançar o controle desses milhares de equipamentos são criados *softwares* (ex., vírus) que buscam por vulnerabilidades nos equipamentos a fim de transformá-los em *bots*. Os *bots* são equipamentos infectados que executam tarefas pré-configuradas e sob controle do atacante (Mirkovic e Reiher, 2004).

As *botnets* são redes constituídas por dispositivos infectados conectados à Internet, chamados *bots*. Essas *botnets* são principalmente utilizadas para a realização de ataques DDoS, que causam prejuízos pela negação do serviço ou infraestrutura (Silva et al., 2013b). O impacto dos ataques DDoS se potencializou pela utilização de técnicas mais sofisticadas dos atacantes. Além disso, o tempo de duração dos ataques DDoS aumentou (ABRANET, 2019; Kaspersky, 2019). Ainda, no Brasil a quantidade de notificações de incidentes de ataques reportados aumentou em 29% e a quantidade de equipamentos reportados como participantes de ataques aumentou em 90%, isso, em 2019 na comparação com 2018 (CERT.br, 2020). Com relação aos prejuízos, os ataques de negação de serviços (do inglês, *Denial of Service* - DoS) e ataques DDoS, em 2018, geraram um prejuízo de 45 bilhões de dólares no âmbito global (Vieira, 2019).

1.1 MOTIVAÇÃO

Devido às consequências causadas pelos ataques de *botnets*, por exemplo, a negação de serviços, divulgação de dados pessoais, perda de recursos computacionais e os prejuízos monetários, o desenvolvimento de estratégias para a detecção e a identificação de *botnets* torna-se indispensável (Kaspersky, 2019). Com o objetivo de conter os prejuízos, o investimento em segurança no Brasil cresceu de 30% a 40% anualmente, alcançando 8 bilhões de dólares em 2018 (TIINSIDE, 2019). Para a detecção de *botnets* existem diversos meios, contudo, ressalta-se a importância da detecção *online* para reduzir o tempo de resposta do sistema, bem como otimizar recursos de processamento e memória (Casas et al., 2019). A detecção *online* utiliza-se de um pequeno conjunto de instâncias para realizar a classificação, descartando as instâncias desnecessárias (Silva et al., 2013a).

No contexto de redes de computadores, observa-se um aumento substancial no volume de dados e na velocidade do tráfego de rede (Souza, 2020). Dessa forma, considerando o aumento no volume de dados e a velocidade do tráfego, é necessário processar os *streams* de forma *online*, isto é, coletar e processar os dados conforme eles chegam para o sistema, e descartar.

Um *stream* refere-se ao envio e recebimento constante de dados pela Internet. A vista disso, deve-se considerar o cenário de redes como dinâmico, podendo conter variações nas distribuições estatísticas dos dados. Dessa maneira, a obtenção de características e identificação de *botnets* em *streams* é um problema. Para tratar este problema é necessário processar o *stream* de dados de forma *online*. Além disso, é necessário lidar com as limitações de memória e considerar os dados contínuos não estacionários que acarretam na ocorrência da mudança de conceito (do inglês, *concept drift*) no comportamento dos dados, em que as características estatísticas dos dados variam no tempo (Ramírez-Gallego et al., 2017).

Comumente, a detecção de *botnets* é realizada utilizando comportamento benigno para o treinamento dos métodos com aprendizagem de máquina, isto é, necessita-se de dados rotulados para que o algoritmo possa aprender o padrão de comportamento dos dados e identificar novas ocorrências. Neste contexto, a obtenção de dados rotulados para o treinamento é difícil, pois necessita interação humana para a confirmação dos rótulos e necessita armazenamento (Silva et al., 2013b; Al Shorman et al., 2020). Além disso, para a detecção *online* esses métodos exigem o treinamento contínuo sobre comportamento da rede para que possam ser eficazes (Casas et al., 2019). Eles necessitam dos rótulos agregados às instâncias, o que prejudica o desempenho de processamento e armazenamento. Dessa forma, é imprescindível o desenvolvimento de novas soluções para atuar de forma contínua, *online* e sem rótulos para detectar e identificar *botnets*.

1.2 PROBLEMA

Uma *botnet* é um conjunto de dispositivos infectados conectados à Internet (*bots*) com ações pré-programadas e um intermediador (*master-bot*) conectado aos *bots* para ordenar a execução de ataques (Feily et al., 2009). A detecção e a identificação de *botnets* são importantes pelo dano potencial causado às vítimas. Dessa forma, são necessárias soluções capazes de detectar *botnets*, ou seja, identificar o conjunto de dispositivos infectados (*bots*). Ainda, para que possam ser tomadas medidas antes que o ataque cause danos a vítima a detecção deve ser feita de forma *online*, isto é, analisando o tráfego assim que ele é gerado pelos dispositivos (Wu et al., 2018). A detecção *online* permite a análise por meio de um bloco de dados, mantendo somente os dados mais novos, contudo apresenta limitações pois necessita processar os dados assim que são recebidos pelo sistema (Casas et al., 2019). No processamento dos dados para detectar *botnets* é necessário distinguir o tráfego benigno do maligno. Contudo, a distinção entre o tráfego benigno e o tráfego *botnet* é dificultado pela sofisticação dos *malwares* para parecer com o tráfego benigno. A detecção de *botnets* consiste na identificação de *bots* e a análise de atividades coordenadas entre dispositivos na rede (Silva et al., 2013b; Wu et al., 2018).

Dentre os trabalhos da literatura existem vários meios para a detecção de *botnets*, porém, é predominante o uso de estratégias de aprendizado de máquina supervisionado (Stevanovic e Pedersen, 2014). Contudo, outras estratégias semi-supervisionadas e não supervisionadas têm ganhado atenção pelo fato de não precisarem de todos os rótulos para treinamento (Yahyazadeh e Abadi, 2015; Wu et al., 2018). As abordagens existentes de aprendizagem de máquina propõem soluções com algoritmos de aprendizagem supervisionada, como em Casas et al. (2019), que estão limitados a atuar com dados rotulados para treinamento do algoritmo, e outros com algoritmos de aprendizagem não supervisionada que comumente possuem limitações com relação à detecção *online* como Wu et al. (2018). Por outro lado, Yahyazadeh e Abadi (2012) apresentam uma solução com aprendizagem não supervisionada *online*, porém, não consideram a ocorrência da mudança do comportamento da rede, também chamado de mudança de conceito. A mudança de conceito pode influenciar de forma negativa na abordagem de classificação dos *bots* seja de forma

supervisionada ou não-supervisionada. Assim torna-se indispensável levar em consideração a mudança de conceito dentro das novas soluções para a detecção e a identificação de *botnets*.

A partir da definição do problema foram construídas duas questões de pesquisa para guiar os objetivos, contribuições e avaliações sobre a proposta dessa dissertação. A primeira questão é *(i)* qual o impacto da detecção de mudanças de conceito na detecção de *botnets*? Com essa questão visa-se avaliar e definir qual o impacto das mudanças de conceito no cenário de classificação de tráfego em redes. A segunda questão é *(ii)* qual a eficácia de um sistema de detecção *online* e não supervisionado de *botnets* levando em consideração mudanças de conceito no tráfego? Por meio dessa questão pretende-se construir o arcabouço de evidências para defender e comparar o sistema proposto com estratégias apresentadas na literatura.

1.3 OBJETIVOS E CONTRIBUIÇÕES

Esta dissertação tem como objetivo contribuir com os estudos relacionados à segurança de redes de computadores por meio do desenvolvimento de um sistema de detecção e identificação de *botnets* de forma *online*. O sistema é independente de treinamento, independente de conhecimento de assinaturas dos ataques e considera possíveis mudanças de conceito da rede. O sistema, denominado TRUSTED (do inglês, *an on-line sysTem foR UnSupervied boTnEt Detection*), consiste em três etapas: *(i)* extração dos dados, *(ii)* janelamento deslizante e detecção de mudança de conceito, e *(iii)* agrupamento, análise da similaridade intra-*cluster* e identificação dos *bots*. A etapa de extração dos dados monitora a rede e separa os pacotes em fluxos de rede por meio da tupla ⟨IP de origem, IP de destino, Porta de origem, Porta de destino e protocolo de comunicação⟩. A etapa de janelamento deslizante e detecção de mudança de conceito constrói o bloco de dados e detecta possíveis mudanças de conceito nos dados. A etapa de agrupamento, análise de similaridade e identificação dos *bots* analisa o bloco de dados construído na etapa dois agrupando as instâncias e calculando a similaridade destas para identificar quais são maliciosas. A avaliação do sistema segue uma abordagem orientada a traços e ressalta a eficácia na detecção e identificação de *botnets* de forma *online* por meio das métricas de acurácia, precisão, *recall* e *F1-Score*. Assim, a principal contribuição dessa dissertação é a apresentação de um sistema capaz de detectar de forma *online* e identificar *botnets* utilizando aprendizado de máquina não-supervisionado considerando as mudanças de conceito.

1.4 ESTRUTURA DA DISSERTAÇÃO

O restante da dissertação está organizado da seguinte forma. O Capítulo 2 expõe os fundamentos. O Capítulo 3 apresenta os trabalhos relacionados. O Capítulo 4 descreve o sistema de detecção e identificação de *botnets*. O Capítulo 5 explica a metodologia de avaliação adotada para os experimentos, bem como apresenta e discute os resultados. O Capítulo 6 conclui esta dissertação e aponta direções futuras.

2 FUNDAMENTOS

Este capítulo descreve os principais conceitos relacionados a esta dissertação. A Seção 2.1 aborda os fundamentos relacionados às *botnets*, como suas arquiteturas, seus estágios de vida, suas características e ataques. A Seção 2.2 apresenta os conceitos, as características e as classificações da aprendizagem de máquina, bem como apresenta os conceitos de mudança de conceito e como ocorre a detecção.

2.1 BOTNETS

As *botnets* são redes constituídas por dispositivos infectados conectados à Internet, chamados *bots*. Os *bots* são controlados por um ou mais atacantes, chamados *bot masters*. Em outros termos, os *bots* são códigos maliciosos executados em dispositivos que permitem o *bot master* coordenar atividades de todo o conjunto de *bots* (*botnet*) de forma remota. O principal objetivo de uma *botnet* é executar atividades maliciosas de forma coordenada e remota (Silva et al., 2013b). Além disso, as redes *botnet*, possuem características de estágios de vida e podem ser classificadas em diferentes arquiteturas, como centralizada, dispositivo para dispositivo (do inglês, *Peer-to-Peer* - P2P), híbridas ou aleatórias (Cooke et al., 2005; Wang et al., 2008). As subseções seguintes apresentam as arquiteturas e estágios de vida das *botnets*.

2.1.1 Arquiteturas

Entre as arquiteturas existentes, a centralizada e a P2P se destacam pois são utilizadas como base para as arquiteturas híbrida e aleatória (Liu et al., 2014). A arquitetura centralizada apresentada na Figura 2.1 possui o atacante e um *bot-master* conectado aos *bots* (Silva et al., 2013b). Este modelo de *botnet* era seguido por todas as *botnets* criadas antes de 2006 (Goodman, 2017), por exemplo a *EarthLink Spammer* (2000), que foi a primeira *botnet* reconhecida publicamente (Srivastava, 2020). A arquitetura centralizada funciona de forma similar ao modelo clássico de rede cliente-servidor, isto é, o *botmaster* possui uma conexão direta a todos os dispositivos da rede. Em uma estrutura centralizada são mantidos alguns centros de comando e controle (do inglês, *Command and Control* - C&C), também chamados de *bot-master* conectados aos *bots* da rede. Estes centros de comando recebem a ordem de ataque por meio de terminais remotos do atacante e encaminham mensagens para todos os *bots* da *botnet* utilizando protocolos (ex., HTTP e IRC) de configurações definidos na arquitetura da *botnet*. As vantagens do uso de *botnets* centralizadas são: *(i)* a rapidez no tempo de reação entre o envio do ataque e *(ii)* a ocorrência do ataque e a facilidade de coordenação devido aos pontos centralizados de C&C. Porém, a facilidade provida pelo uso de pontos centralizados de C&C também apresentam pontos únicos de falha, onde na ocorrência de problemas isolados a *botnet* fica inutilizável (Silva et al., 2013b; Goodman, 2017).

As *botnets* modernas necessitam de maior flexibilidade e robustez para lidar com todos os *bots* e para maximizar os lucros, como a implementação de suporte a alteração de assinatura da *botnet* sem ter a necessidade de infectar todos os *bots* novamente (Goodman, 2017). As *botnets* com arquitetura descentralizada são mais complexas e por isso são mais difíceis de descobrir (Silva et al., 2013b). Este modelo foi detectado pela primeira vez em 2006, com a *botnet Nugache*. Depois, várias outras foram identificadas como: a *Storm*, a *Smoke Bot*, a *Zero*

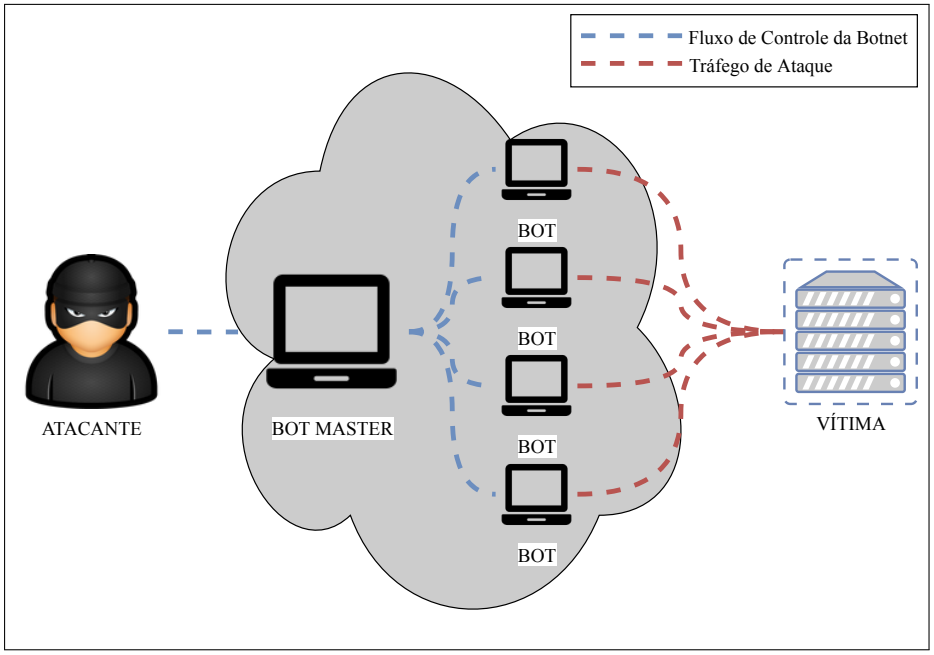


Figura 2.1: Arquitetura de uma botnet centralizada

Acess, a *Zeus* e a *Mirai*. Em 2016, a *botnet Mirai* foi identificada quando realizou um dos maiores ataques da história (Goodman, 2017).

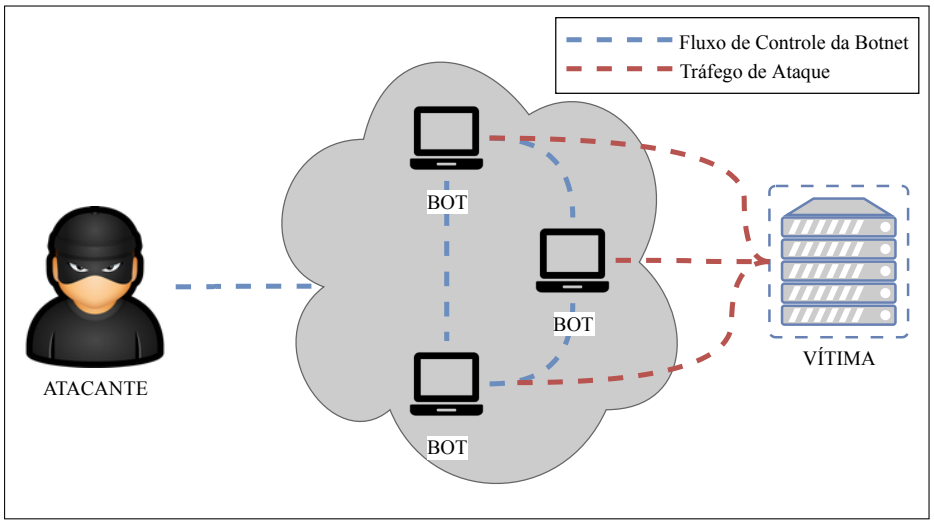


Figura 2.2: Arquitetura de uma *botnet* P2P

A Figura 2.2 apresenta uma representação gráfica da arquitetura descentralizada, onde o atacante envia o comando de ataque por meio de um terminal remoto para alguns dispositivos da *botnet* e estes dispositivos disseminam o comando até alcançar todos os dispositivos conectados a eles até que se alcance toda a *botnet*. Nesta arquitetura a detecção de nós isolados não acarreta na desarticulação da rede *botnet* pois não existe um ponto C&C para ser descoberto e desabilitado. As redes *botnet* descentralizadas comumente utilizam protocolos P2P e podem ser classificadas como *(i)* rede P2P desestruturada, *(ii)* rede P2P estruturada e *(iii)* rede P2P com super nó (do inglês, *superpeer*) (Jelasity et al., 2009). As redes P2P desestruturadas consistem em redes com topologias de diferentes graus de distribuição, ou seja, não são mapeadas e não possuem um padrão de conexões, elas não oferecem suporte a roteamento nem chaves de procura (do inglês,

key lookup). As redes P2P estruturadas são mapeadas. Elas utilizam uma tabela distribuída entre os nós para o mapeamento e roteamento da rede (Risse et al., 2004). As redes com super nós se assemelham a redes centralizadas, pois possuem super nós que gerenciam um grupo de nós comuns. Estas redes também mantém uma tabela distribuída para mapear e realizar o roteamento (Zhang et al., 2010).

As *botnets* híbridas e aleatórias são robustas, porém não são comumente utilizadas por ter menor alcance de recrutamento de dispositivos (Liu et al., 2014). As redes *botnets* híbridas implementam características das redes centralizadas e redes P2P, um exemplo de rede híbrida são redes P2P com super nós, onde existem pontos de C&C que coordenam grupos de nós comuns. Cooke et al. (2005), introduziram o modelo de redes aleatórias. No modelo, os *bots* não se comunicam com o *botmaster*, somente aguardam uma tentativa de comunicação do *botmaster*. Esse modelo é facilmente implementado e possui alta resiliência, no entanto possui problemas de escalabilidade devido à necessidade de recrutar *bots* na rede. Além disso, o modelo aleatório é considerado apenas teórico (Silva et al., 2013b).

2.1.2 Estágios de Vida de Uma Botnet

De forma geral, as *botnets* seguem passos similares entre si para criar a rede, recrutar membros e manter a *botnet* (Alieyan et al., 2017). Segundo Wainwright e Kettani (2019), os estágios de vida de uma *botnet* podem ser classificados como *(i)* concepção, *(ii)* recrutamento, *(iii)* interação, *(iv)* propaganda (*marketing*) e *(v)* execução de ataque. Assim, é possível identificar a fase inicial da *botnet* compreendida pelos itens (*i, ii, iii, iv*), e viabilizar a detecção antecipada a fim de evitar qualquer dano exequível pelo atacante. Cada uma das fases é explicada a seguir.

- Na primeira fase, a concepção da *botnet* refere-se às motivações para projetar e implementar uma *botnet*. As motivações incluem ganho financeiro, entretenimento, ego, causa, entrada para grupos sociais e *status* (Rodríguez-Gómez et al., 2013; Wainwright e Kettani, 2019). A motivação para a criação da *botnet* influência na decisão do *design* de implementação que pode variar entre as arquiteturas apresentadas na Subseção 2.1.1 dependendo das necessidades da *botnet* (Rodríguez-Gómez et al., 2013). Uma defesa para a concepção é eliminar a motivação por meio de repercussões legais severas (Wainwright e Kettani, 2019).

- Na segunda fase, o recrutamento, o objetivo é maximizar o número de dispositivos na *botnet* (Rodríguez-Gómez et al., 2013; Wainwright e Kettani, 2019). Dessa forma, a propagação da *botnet* é realizada explorando vulnerabilidades em dispositivos e usuários através de vírus (ex., *trojan horses em websites*). Em teoria, se houvesse uma forma de prevenir o recrutamento, *botnets* não poderiam ser construídas. A fim de prevenir o recrutamento pode-se procurar identificar a comunicação do *botmaster* enquanto ele busca por novos dispositivos, incrementar a segurança dos dispositivos e manter os sistemas operacionais atualizados (Wainwright e Kettani, 2019).

- A terceira fase, a interação, envolve a comunicação entre os dispositivos da *botnet*, bem como comunicações com dispositivos externos para explorar potenciais dispositivos vulneráveis (Rodríguez-Gómez et al., 2013; Wainwright e Kettani, 2019). Assim como no recrutamento, uma solução para detectar os *bots* é por meio da identificação de tráfego anômalo na rede (Wainwright e Kettani, 2019). Nesta fase, as comunicações internas (entre *bots*) são realizadas por meio de canais C&C e protocolos do tipo *Internet Relay Chat* (IRC) e *Hypertext Transfer Protocol* (HTTP). Ainda, para atingir dispositivos externos são utilizados serviços de rede disponibilizados *online* como Sistema de Nomes

de Domínio (do inglês, *Domain Name Service* - DNS) e serviços de redes *Peer-to-Peer* (P2P) (Rodríguez-Gómez et al., 2013).

- Na quarta fase, a propaganda, a finalidade é relacionada com a concepção e consiste no anúncio da venda do código ou aluguel da rede *botnet*. A ideia é tornar a rede *botnet* rentável. Para esta fase, uma potencial defesa é identificar o criador por meio de anúncios na *dark web* (Wainwright e Kettani, 2019).

- Na quinta fase, o objetivo é executar ataques (ex., DDoS, *spam, phishing*) (Rodríguez-Gómez et al., 2013; Wainwright e Kettani, 2019). Defesas contra-ataques são desenvolvidas, com o intuito de identificar o tráfego malicioso antes do ataque atingir seu alvo (Wainwright e Kettani, 2019).

2.1.3 Ataques de Negação de Serviços

Em geral, as *botnets* são utilizadas para realizar ataques DDoS (Mahmoud et al., 2015). Estes ataques visam negar uma infraestrutura de rede ou um serviço aos usuários legítimos, de forma a prejudicar diretamente todos os envolvidos. Uma das formas mais frequentes para causar a negação de serviços é enviar um grande conjunto de pacotes para sobrecarregar os recursos disponíveis pela vítima e tornar o serviço indisponível aos usuários autênticos (Mirkovic e Reiher, 2004). Com o aumento da largura de banda da Internet, comumente os ataques atingem elevadas taxas de envios tornando-os potentes o suficiente para superar as capacidades de qualquer alvo (Stallings e Brown, 2014). Esse tipo de ataque engloba três entidades. O atacante, o meio do ataque e a vítima (ex., infraestrutura de rede, servidor). O atacante é a entidade que executa os comandos para a efetivação do ataque. O meio do ataque consiste de dispositivos infectados utilizados para realizar o ataque. A vítima é a infraestrutura ou dispositivo que disponibiliza serviços que sofre o ataque.

De acordo com (Mirkovic e Reiher, 2004), os ataques DDoS são julgados pela forma de preparação do ataque, pelas características do ataque e o efeito que causa na vítima, isso levando em consideração: *(i)* o grau de automação, *(ii)* a forma como são exploradas as vulnerabilidades, *(iii)* a validade do endereço de origem, *(iv)* a dinâmica do ataque, *(v)* a possibilidade de caracterização, *(vi)* persistência do grupo de agentes, *(vii)* tipo de vítima, *(viii)* impacto na vítima. Ainda sobre a classificação dos ataques, Zargar et al. (2013) afirma que os ataques DDoS são classificados com base na camada de rede em que o ataque é realizado.

Independente da magnitude dos danos dos ataques DDoS e a disponibilidade de sistemas para defesa, ainda perduram os desafios para impedi-los, vista a sua contínua evolução em relação a frequência, sofisticação e volume. Entre os desafios é possível verificar a necessidade de mecanismos de defesa. Dessa forma, para encontrar a melhor solução de defesa é fundamental a escolha do ponto em que se deve alocar o sistema de defesa, qual o custo para a implantação do sistema, detalhar as necessidades do sistema (ex., conhecimento dos ataques), escolher o melhor sistema e realizar testes em larga escala antes da implantação (Mirkovic e Reiher, 2004).

De acordo com Zargar et al. (2013), existem duas principais técnicas para realizar ataques DDoS. A primeira consiste no envio de pacotes falsos para o alvo com o objetivo de confundir o protocolo ou a aplicação em execução, negando o serviço ou a infraestrutura. Esse ataque é conhecido como ataque de vulnerabilidade. A segunda técnica consiste em exaurir os recursos do alvo e pode ser implementada de duas formas: *(i)* esgotar a largura de banda, a capacidade de processamento do roteador ou dos recursos da rede por meio do envio de grande volume de requisições; e *(ii)* esgotar os recursos dos servidores como conexões, processador, memória ou armazenamento. A primeira técnica refere-se a um ataque de inundação na camada

de rede ou transporte (ex., *ping flooding*). A segunda técnica também é um ataque de inundação, porém, é realizada na camada de aplicação (ex., HTTP *flooding*).

Por outro lado, (Mahjabin et al., 2017) classifica os diferentes tipos de ataques DDoS entre: *(i)* exaustão da largura de banda, *(ii)* exaustão de recursos, *(iii)* ataque direcionado a infraestrutura e *(iv)* ataques *zero-day*. Os ataques de exaustão de largura de banda têm como objetivo consumir toda a largura de banda que o sistema da vítima possa utilizar. Como resultado, a vítima tem seu serviço negado aos usuários legítimos até que o ataque seja mitigado. Este tipo de ataque também pode explorar protocolos e ser feito por amplificação, onde o atacante utiliza uma entidade amplificadora para aumentar a densidade do ataque e o dano causado a vítima. Os ataques de exaustão de recursos têm como objetivo exaurir os recursos como: a memória, os *sockets* e o processamento. Estes ataques são realizados de duas formas, *(i)* explorando os protocolos de transporte e de aplicação e *(ii)* utilizando fragmentação e geração de pacotes falsos. Os ataques direcionados a infraestrutura buscam exaurir os recursos computacionais de serviços cruciais para a Internet, como servidores de Serviços de Nomes e Domínios (do inglês, *Domain Name Service*). O impacto em servidores de DNS afeta o acesso a Internet de forma geral, pois são eles que proveem a tradução de endereços de protocolo de Internet (do inglês, *Internet Protocol*) para domínios que possam ser acessados via navegador. Por fim, os ataques *zero-day* ocorrem explorando vulnerabilidades desconhecidas e são chamados por esse nome pois a vulnerabilidade somente é descoberta um dia após o ataque (Mahjabin et al., 2017).

2.2 APRENDIZAGEM DE MÁQUINA

A aprendizagem de máquina é uma área da Inteligência Artificial que cria técnicas computacionais de aprendizado e constrói modelos com habilidade de aprender e identificar, dentro de um conjunto de categorias, a qual categoria uma determinada instância pertence. Os problemas da classificação de instâncias e o reconhecimento de padrões em dados são antigos. Com o advento do aprendizado de máquina, buscou-se automatizar as descobertas de padrões por meio de algoritmos. O objetivo do desenvolvimento de algoritmos para esse campo é construir métodos que recebam como entrada um conjunto de instâncias com um vetor X de características, e produzam como saída uma classificação y, isto é, dado um conjunto de características, o método deve classificar as instâncias entre as categorias possíveis. Assim, o aprendizado pode ser supervisionado, onde o sistema recebe um vetor X e um y de características e rótulos para treinar. O aprendizado de máquina também pode ser não supervisionado, onde o sistema somente recebe um vetor X de características e busca classificar as instâncias buscando semelhanças no vetor de características (Bishop, 2006; Mohri et al., 2018).

2.2.1 Aprendizagem de Máquina Supervisionada

No aprendizado supervisionado, são utilizados algoritmos de indução, ou seja, algoritmos que utilizam exemplos anteriores para aprender e classificar novos exemplos. Este tipo de aprendizado, conforme apresentado na Figura 2.3, que mostra o fluxo da classificação por um modelo supervisionado. Os modelos supervisionados recebem um conjunto de dados de treinamento utilizado para criar o modelo de classificação. A partir do modelo criado, é utilizado um conjunto de dados de teste, este não deve ter sido utilizado para treinar o modelo. O conjunto de dados de teste é classificado pelas regras de classificação do modelo criado a partir dos dados de treinamento. Por fim, é obtida a classificação das instâncias do conjunto de teste, que resultam em uma lista de rótulos indicados pelo classificador e são utilizados para averiguar as métricas de classificação. Cada instância é constituída por um vetor de características e o rótulo da categoria,

no conjunto de treinamento o rótulo é utilizado para aprendizado, e no conjunto de teste é utilizado para computar o desempenho de classificação do modelo. Assim, o objetivo é construir um classificador que seja capaz de acertar a categoria de novas instâncias sem rótulos (Nguyen e Armitage, 2008).

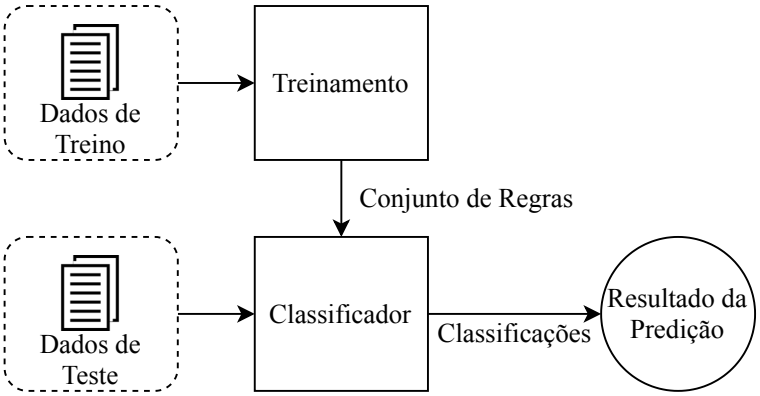


Figura 2.3: Arquitetura da aprendizagem supervisionada. Adaptado de (Muhammad e Yan, 2015)

Para criação do modelo de classificação podem ser utilizados vários tipos de algoritmos, este é um passo importante para alcançar bons resultados. Os algoritmos podem ser: *(i)* baseados em lógica (ex., árvores de decisão), *(ii)* de aprendizagem estatística (ex., *naive bayes*), *(iii)* de aprendizagem baseada em instâncias (ex., kNN), *(iv)* máquina de vetores de suporte (ex., *Support Vector Machine* - SVM), *(v)* de aprendizagem profunda (ex., redes neurais). Estes algoritmos são avaliados utilizando métricas como acurácia e precisão (Muhammad e Yan, 2015).

2.2.2 Aprendizagem de Máquina Não Supervisionada

O aprendizado não supervisionado analisa o conjunto de instâncias fornecidas, sem rótulos, e busca agrupá-las. O resultado obtido pelo uso do aprendizado não supervisionado é um conjunto de rótulos referentes a cada *cluster* em que as instâncias foram classificadas (Nguyen e Armitage, 2008). Ainda, para utilizar a aprendizagem não supervisionada são necessários dois requisitos, a seleção de características utilizadas para o agrupamento ou *clustering* e a seleção da métrica de similaridade para comparação entre instâncias (Gentleman e Carey, 2008).

A seleção para cada requisito antes da utilização da aprendizagem não supervisionada é importante (Gentleman e Carey, 2008). A seleção de características é um problema relacionado ao desempenho dos algoritmos de *clustering*. Os métodos de seleção de características avaliam no conjunto completo de características disponíveis, quais são as características com maior ganho de informação. Utilizando somente as características selecionadas (conjunto menor que o original) é possível obter maior desempenho em tempo de classificação (Moussa et al., 2019). A seleção da métrica de similaridade entre instâncias influencia no desempenho do algoritmo de *clustering*, a métrica de similaridade implica em como é realizado o cálculo da distância entre as instâncias, portanto, altera o resultado da etapa de agrupamento (Gentleman e Carey, 2008).

Com relação a classificação dos métodos de agrupamento, eles podem ser classificados entre, *(i)* algoritmos de particionamento e *(ii)* baseados em hierarquia. Os algoritmos de particionamento comumente procuram dividir o conjunto de instâncias em uma quantidade específica de agrupamentos ou *clusters*, esta quantidade é fornecida como entrada para o algoritmo. Além disso, a maioria desses algoritmos de particionamento é baseada em otimização por um critério de concordância entre os dados de entrada e os agrupamentos construídos, ou seja, os agrupamentos mudam conforme o algoritmo otimiza a divisão no mapa de características.

Os algoritmos de agrupamento hierárquico têm por objetivo construir uma hierarquia de agrupamentos. Dessa forma, esses métodos podem ser do tipo aglomerativo ou divisivo. Nos algoritmos aglomerativos, inicialmente, cada instância é considerada um agrupamento, e a cada iteração são agrupados os dois agrupamentos mais próximos conforme apresentado no dendograma da Figura 2.4. Por outro lado, nos algoritmos divisivos, o conjunto inteiro de instâncias é considerado apenas um agrupamento, e a cada iteração ocorre a divisão dos agrupamentos até alcançar uma distância mínima (Grira et al., 2004).

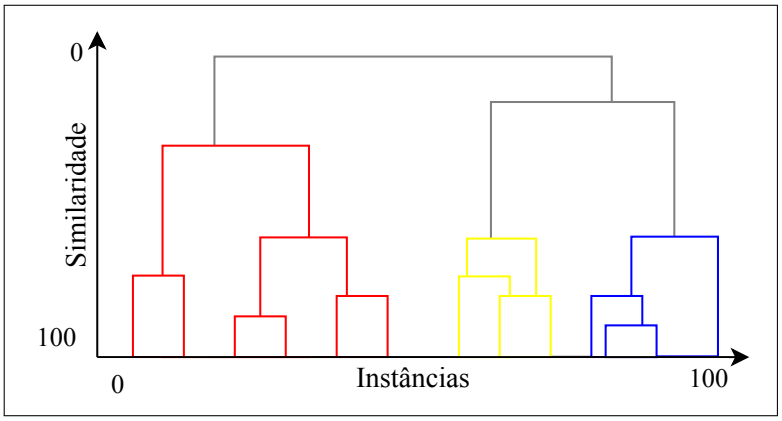


Figura 2.4: Dendograma, resultado do agrupamento hierárquico

2.2.3 Aprendizagem *Online*

A aprendizagem *online* é um paradigma de aprendizado bem estabelecido e tem áreas teóricas e práticas. O objetivo da aprendizagem *online* é fazer uma sequência de classificações precisas com o passar do tempo, baseando-se em um conjunto de dados passado e buscando aprimorar o modelo de classificação atual a cada nova classificação (Shalev-Shwartz et al., 2011). Além das tarefas de classificação, outra usabilidade é a análise regressiva, que se refere ao processo de aprendizagem para estimar as relações entre as características das instâncias (normalmente com uma característica dependente e uma ou mais independentes). O aprendizado *online* são comumente aplicados em tarefas de regressão (ex., análise de mercado financeiro e tráfego de rede) visto que chegam naturalmente de uma forma sequencial (Hoi et al., 2018).

Hoi et al. (2018) propuseram uma taxonomia dos métodos e técnicas de aprendizagem *online*. De forma geral, o aprendizado *online* é baseado nas teorias de aprendizagem, da otimização e dos jogos. Além disso, os métodos e técnicas podem ser agrupados em diferentes categorias de acordo com seus princípios de aprendizagem e aplicação. Especificamente de acordo com os tipos de retornos e estratégias de supervisão. Assim, principalmente, os métodos e técnicas podem ser classificados entre aprendizado *online* supervisionado, aprendizado *online* supervisionado com retornos limitadas e aprendizado *online* não supervisionado.

A aprendizagem online supervisionada trata tarefas de aprendizagem onde ao obter o resultado da classificação final o modelo de classificação tem um retorno sobre qual o real rótulo da instância classificada. Dessa forma, essa estratégia utiliza o rótulo real para ajustar o modelo de classificação para as novas entradas (Hoi et al., 2018). Neste cenário, Carela-Español et al. (2016) e Casas et al. (2019) propuseram técnicas para classificação incremental aplicando os rótulos reais após a classificação para aprendizagem.

A aprendizagem *online* com retornos limitados trata tarefas onde o modelo de classificação recebe apenas informações parciais sobre os rótulos reais das instâncias. Por exemplo, dada uma tarefa de classificação multi classe, o classificador faz a classificação e recebe o retorno

da classificação em correta ou errada ao invés do rótulo real. Para essas tarefas o modelo de classificação necessita realizar atualizações no modelo a fim de se adequar as novas instâncias e obter resultados precisos na classificação (Hoi et al., 2018). Utilizando essa estratégia Khanchi et al. (2018) propuseram uma técnica de classificação alimentando o modelo de classificação com uma quantidade reduzida de rótulos verdadeiros, reduzindo o custo de rotulação.

A aprendizagem *online* não supervisionada trata de tarefas onde o modelo de classificação somente recebe a sequência de instâncias de dados sem qualquer forma de supervisão, ou seja, executa a classificação baseada apenas nas características de cada instância. Por ser uma aprendizagem sem supervisão é considerada uma extensão natural da aprendizagem não supervisionada tradicional, adaptada apenas para lidar com o fluxo de dados sequenciais (Hoi et al., 2018). Como exemplo de aprendizagem *online* não supervisionada temos a clusterização *online* (Yu et al., 2010; Yahyazadeh e Abadi, 2015), a redução de dimensionalidade *online* (Kuncheva e Faithfull, 2013; Qahtan et al., 2015) e a detecção de anomalias *online* (Al Shorman et al., 2020). A aprendizagem *online* não supervisionada tem menos suposições restritas a utilização dos rótulos verdadeiros e essa estratégia pode ser aplicada em cenários onde a obtenção de rótulos é muito custosa (ex., detecção de *botnets*) (Al Shorman et al., 2020).

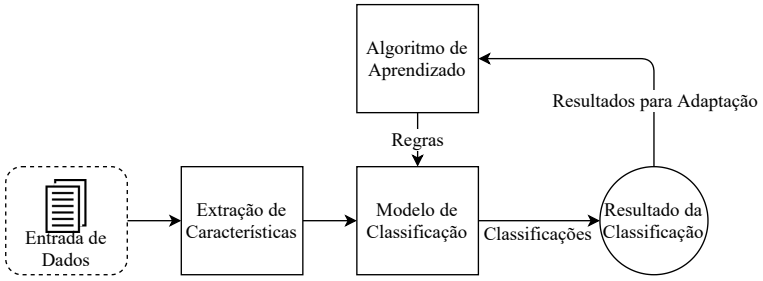


Figura 2.5: Arquitetura da aprendizagem *online*. Adaptado de (Zheng et al., 2017)

De forma geral, a aprendizagem segue uma abordagem iterativa, conforme apresentado na Figura 2.5, ela aprende classificando e adaptando o modelo uma instância por vez. O aprendizado ocorre pela adaptação do modelo de classificação levando em consideração as novas instâncias processadas (Zheng et al., 2017). No processo de adaptação a detecção de mudanças de conceito pode auxiliar a manter um conjunto de dados atualizados e dentro de uma mesma distribuição estatística (Gözüaçık et al., 2019; Bifet e Gavalda, 2007)

2.2.3.1 Mudanças de Conceito

Em ambientes não estacionários a distribuição dos dados pode variar, e dessa forma, produzir o fenômeno da mudança de conceito (Gomes et al., 2017). As mudanças de conceito podem ser de dois tipos: mudança de conceito real e mudança de conceito virtual. A mudança de conceito real refere as mudanças condicionais na distribuição dos resultados (categoria classificada), ou seja, para ocorrer uma mudança de conceito real o classificador deve errar a categoria dado um conjunto de características de entrada que teve a distribuição dos dados alterada. A mudança de conceito virtual refere-se a mudanças na distribuição dos dados de entrada que não afetam a assertividade do classificador atual, e são identificadas por meio da análise estatística dos dados (Gama et al., 2014). A Figura 2.6 apresenta as mudanças de conceito em relação aos dados, os círculos representam instâncias de dados e as cores representam categorias diferentes.

As variações da distribuição dos dados podem ocorrer de diferentes formas, como apresentado na Figura 2.7 considerando uma ambiente uni-dimensional onde ocorrem as

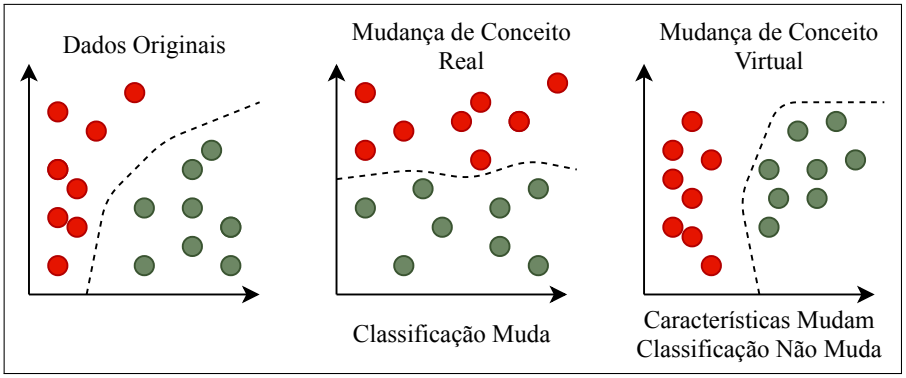


Figura 2.6: Tipos de mudanças de conceito. Adaptado de Gama et al. (2014)

mudanças. Uma mudança de conceito pode ocorrer de forma abrupta, incremental, gradual e recorrente. A forma abrupta substitui uma distribuição de dados por outra nova em um instante. A forma incremental consiste em várias mudanças de conceito entre uma distribuição e outra. A forma gradual apresenta várias mudanças de conceito abruptas ocorrendo de forma sazonal. A mudança recorrente refere-se à ocorrência de mudanças de conceito repetidas (ex., quando uma mudança de conceito volta a uma distribuição antiga) (Minku et al., 2010).

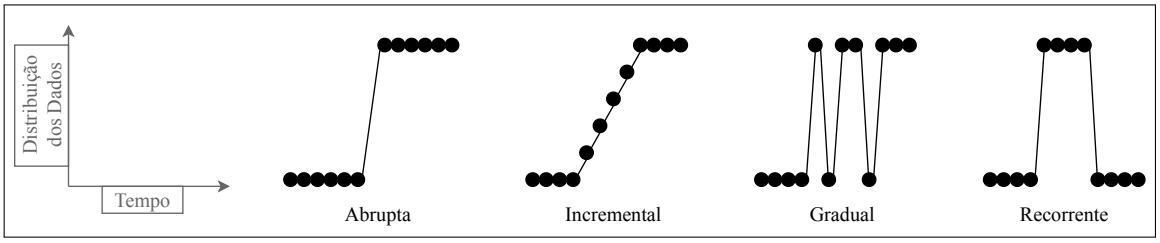


Figura 2.7: Tipos de variações na distribuição de dados. Adaptado de Gama et al. (2014)

O cenário para avaliar a mudança de conceito considera um ambiente *online*, ou seja, as instâncias de dados são processadas de acordo com a ordem de chegada. Para que o sistema de classificação funcione em conjunto com a detecção da mudança de conceitos é necessário utilizar classificadores incrementais e que possam se adaptar conforme as mudanças de conceito ocorrem (Gama et al., 2014). Neste contexto, segundo Gama et al. (2014), a aprendizagem incremental *online* é definida a seguir:

- **Predição:** Quando uma nova instância X chega, uma predição y é feita utilizando o modelo atual L_t.
- **Diagnose:** Comparando a predição y com o real rótulo $\overline{y}$ é estimado o erro do classificador.
- **Atualização:** Se ocorreu uma mudança no conceito, atualiza o classificador para um novo modelo L_{t+1}

A detecção de mudanças de conceito pode ser realizada de forma supervisionada (Bifet e Gavalda, 2007) e de forma não supervisionada (Gözüaçık et al., 2019). Na metodologia de detecção de mudanças de conceito supervisionada são consideradas estimativas de erro do classificador para avaliar se houve uma mudança de conceito no conjunto de dados da entrada. Em um modelo genérico de aprendizagem adaptativa e detecção supervisionada de mudanças de

conceito, o sistema recebe como entrada um conjunto de dados para treino e classificação, a partir do modelo criado classifica as instâncias do conjunto de dados. Com o rótulo da classificação e uma entrada do rótulo real da instância realiza a estimativa de erro do classificador e detecta se houve uma mudança de conceito para gerar o alarme e adaptar o modelo de classificação (Gama et al., 2014).

A detecção não supervisionada de mudanças de conceito, por outro lado, não considera estimativas de erro do classificador para detectar as mudanças de conceito. Nesta metodologia é considerada a distribuição do conjunto de dados no mapa de características, isto é, um método de detecção não supervisionada de mudança de conceito compara o conjunto de dados atual com um conjunto de dados novo. Isso é feito a fim de detectar uma possível discrepância entre os conjuntos de dados (Sethi e Kantardzic, 2017).

2.3 RESUMO

Este capítulo apresentou os conceitos necessários para o entendimento do restante deste manuscrito. Iniciou pela fundamentação das *botnets*, conceituando-as com suas características, arquiteturas, ataques e estágios de vida. Conceituou também a aprendizagem de máquina como uma área da inteligência artificial que busca criar técnicas computacionais de aprendizado e desenvolver modelos com habilidade de aprender automaticamente e identificar, dentro de um conjunto de categorias, a qual categoria uma determinada instância pertence. Adicionalmente, auxilia no entendimento das técnicas de classificação, também descritos neste capítulo a fim de construir o arcabouço de conceitos utilizados nesta pesquisa que propõe um sistema de detecção *online* não supervisionada de *botnets*.

3 TRABALHOS RELACIONADOS

O problema de detectar *botnets* é constantemente tratado por trabalhos na literatura, tais como (Yu et al., 2010; Yahyazadeh e Abadi, 2015; Barthakur et al., 2015; Carela-Español et al., 2016; Wu et al., 2018; Casas et al., 2019). Esses trabalhos de uma forma geral utilizam estratégias supervisionadas ou não supervisionadas, e seguem abordagens *online* ou *off-line*. Os trabalhos supervisionados empregam a estratégia tradicional para modelos de classificação (treino e teste), contudo o uso de supervisão requer um conjunto de dados rotulados para treinar (Carela-Español et al., 2016). Esta necessidade apresenta desafios como armazenamento e intervenção humana para rotular os dados tornando a estratégia supervisionada custosa e lenta (Al Shorman et al., 2020). O emprego de técnicas não supervisionadas oferece a classificação dos dados sem treinamento ou conhecimento prévio. Essas técnicas se apresentam de duas formas principais *off-line* e *online*. Os trabalhos que aplicam técnicas de classificação não supervisionada *off-line* realizam a detecção de *botnets* por meio de uma captura do tráfego e posterior análise para classificar os dados (Barthakur et al., 2015; Wu et al., 2018). Por outro lado, os trabalhos que aplicam técnicas de classificação não supervisionada *online* conseguem analisar o tráfego conforme ele é capturado, oferecendo maior velocidade para a tomada de decisão do administrador da rede (Yu et al., 2010; Yahyazadeh e Abadi, 2015).

Contudo, estes trabalhos ignoram a volatilidade e as rápidas mudanças no comportamento do tráfego de redes, conhecidas como mudanças de conceito (do inglês, *concept drift*). Assim, diferentes trabalhos descrevem os problemas que a mudança de conceito ocasiona na classificação de um *stream* de dados (Gama et al., 2004, 2014; Casas et al., 2019). Dessa forma, a mudança de conceito influencia diretamente no modelo de aprendizagem, normalmente prejudicando o desempenho da classificação (Gama et al., 2014).

Assim, este capítulo está organizado como segue. A Seção 3.1 apresenta o estado da arte das principais técnicas de detecção de *botnets*. A Seção 3.2 apresenta os diferentes métodos para a detecção da mudança de conceito. A Seção 3.3 apresenta o resumo do capítulo.

3.1 DETECÇÃO DE *BOTNETS*

A área de detecção de *botnets* tem recebido considerável atenção na literatura (Carela-Español et al., 2016; Boutaba et al., 2018; Casas et al., 2019). Os trabalhos nesta área podem ser classificados principalmente entre os critérios: *(i)* técnica de aprendizagem usada (Yahyazadeh e Abadi, 2015), *(ii)* detecção *off-line* ou *online*, e *(iii)* percepção da mudança de conceito (Casas et al., 2019). As técnicas de aprendizagem podem ser categorizadas em supervisionada e não supervisionada (Yahyazadeh e Abadi, 2015; Wu et al., 2018). Na supervisionada, o sistema deve utilizar uma base de dados rotulada para treinar um modelo de aprendizagem. No entanto, estas técnicas são suscetíveis a erros e a reconhecer somente *botnets* rotuladas. As técnicas de aprendizagem não supervisionadas facilitam a aprendizagem *online* além de adequar-se para a detecção de *botnets* desconhecidas, pois não necessitam de rótulos para a classificação (Yahyazadeh e Abadi, 2015). Além disso, as mudanças de conceito afetam o desempenho da técnica de aprendizagem (Casas et al., 2019). Assim, se for detectada uma mudança de conceito, a técnica de aprendizagem deve ser adaptada para o novo estado na distribuição dos dados.

3.1.1 Detecção de *Botnets* com Aprendizagem Supervisionada

Os principais trabalhos relacionados à detecção supervisionada de *botnets* podem ser divididos entre *(i) off-line* e *(ii) on-line* (Casas et al., 2019). A detecção *off-line* por métodos supervisionados funciona separando a base de dados em dados para treino e dados para teste. Inicialmente, o classificador é treinado com a base de dados para treino e depois é utilizado para classificar os dados da base de teste (Perdisci et al., 2009). Por outro lado, é possível utilizar treino e classificação de forma *online*, o classificador incremental é inicialmente treinado com um pequeno conjunto de dados para iniciar a classificação. Com o resultado da classificação o modelo gerado deve ser reconstruído para conhecer os novos dados (Carela-Español et al., 2016).

Utilizando detecção supervisionada *off-line*, Kim et al. (2005) apresentaram um sistema de detecção de intrusos (do inglês, *Intrusion Detection System* - IDS) com a fusão de um algoritmo genético e uma máquina de vetores de suporte (do inglês, *support vector machines* - SVM). O algoritmo genético provê um conjunto de características reduzido e representativo para o SVM. Assim, é alcançada uma alta taxa de detecção, porém não é averiguada a taxa de falsos positivos. Hu et al. (2008) apresentam um sistema utilizando um algoritmo *ensemble* AdaBoost. Este combina classificadores para obter melhores resultados. Dessa forma, o sistema alcançou uma alta taxa de detecção e baixa taxa de falsos positivos. Ainda, Stevanovic e Pedersen (2014) realizaram uma análise de desempenho em oito algoritmos de classificação supervisionada a fim de verificar a melhor eficiência com relação à taxa de detecção, à taxa de falsos positivos e ao custo de classificação. O melhor desempenho de classificação com alta taxa de detecção, baixa taxa de falsos positivos e baixo custo computacional foi obtido utilizando árvore de decisão aleatória. Contudo, estes trabalhos somente consideram a classificação de forma *off-line* e necessitam do conjunto de rótulos para cada instância no treinamento, o que os torna inviáveis para a detecção em ambientes com *stream online* de dados.

Na detecção supervisionada *online*, Carela-Español et al. (2016) apresentaram uma técnica para a classificação incremental de tráfego de rede. A técnica consiste em uma árvore de Hoeffding (do inglês, *Hoeffding Tree* - HT). A técnica foi avaliada e obteve uma taxa de detecção acima de 90% e baixa significante do custo computacional em relação a árvores de decisão em modo *off-line*. Casas et al. (2019) realizaram uma avaliação de quatro algoritmos de classificação *online* sem a detecção de mudança de conceito e utilizando janelamento dinâmico (do inglês, *Adaptative Windowing* - ADWIN) e janelamento fixo (do inglês, *Fixed Windowing* - FIXWIN) para a detecção de mudança de conceito. Os resultados mostram que ambas as técnicas de detecção de mudança de conceito resultam em um aumento significativo no desempenho geral dos classificadores avaliados. Na detecção *online* e com aprendizagem adaptativa, isto é, se adapta aos novos contextos de rede, podem ser detectados ataques *zero-day*, ou seja, ataques sem rótulos conhecidos. No entanto, estes métodos utilizam os rótulos para treino dos classificadores e para a detecção de mudanças de conceito, e a obtenção de rótulos para treino ou detecção de mudanças de conceito é custosa e lenta tornando o uso de abordagens supervisionadas inviável na área de segurança de redes (Al Shorman et al., 2020).

3.1.2 Detecção de *Botnets* com Aprendizagem Não Supervisionada

De forma predominante os métodos de detecção são supervisionados (Stevanovic e Pedersen, 2014). Estes utilizam diversas formas de classificação como árvores de decisão, classificadores probabilísticos e redes neurais artificiais. No entanto, os classificadores não supervisionados se destacam pela execução sem a necessidade de rótulos para treinamento e por isso, de forma geral, são capazes de detectar ataques *zero-day* visto que são independentes de

treinamento. Além disso, como os métodos supervisionados, os trabalhos de detecção de *botnets* não supervisionados também podem ser classificados como: *(i) off-line* e *(ii) on-line*.

Na detecção de *botnets* de forma *off-line* e não supervisionada, Barthakur et al. (2015) apresentaram o CluSIBotHealher (*Botnet Detection through Similarity Analysis of Clusters*) que identifica *botnets* P2P por meio do agrupamento iterativo e estatístico de instâncias de fluxo e análise de similaridade entre instâncias de um mesmo grupo. Narang et al. (2014) apresentaram um *framework* para a detecção de *botnets* P2P, o PeerShark. Neste, foram combinados dois modelos de detecção, supervisionada e não supervisionada. Primeiro, são agrupadas as instâncias de fluxos, com os agrupamentos e um classificador supervisionado, que é treinado com instâncias benignas, o tráfego malicioso é identificado. Este trabalho pode ser considerado híbrido entre supervisionado e não supervisionado, dessa forma, está marcado com ✓- na Tabela 3.1. Wu et al. (2018) realizaram uma análise de três algoritmos de agrupamento sobre instâncias de fluxo em duas bases de dados e utilizam análise de similaridade para a identificação final de tráfego malicioso. Os resultados mostram que os algoritmos *Simple-K Means* e *X-Means* alcançaram acurácia de 89%, porém não apresentaram métricas de precisão e *recall* que seriam mais adequadas, visto que as bases de dados utilizadas são desbalanceadas. Contudo, estes trabalhos apresentam formas de detecção *off-line* de *botnets*, assim, uma vez que os ataques causam danos de forma célere, torna-se inviável a utilização destes métodos de detecção de *botnets* pela demora na apresentação dos resultados.

Utilizando estratégias de detecção *online* de *botnets*, Wurzinger et al. (2009) apresentaram um sistema que identifica *bots* sem conhecimento prévio sobre os canais de comunicação ou modo de propagação. O sistema é baseado em um modelo de dois estados. No primeiro, o sistema busca por sinais específicos no tráfego. Se um sinal é encontrado o sistema entra no segundo estado. Nesse estado, o sistema busca analisar se o *bot* respondeu ao sinal do *bot-master*. Dessa forma, caso o sistema encontre a resposta, uma infecção é alarmada. No entanto, este método é frágil quando o tráfego de comunicação da *botnet* é encriptado.

Yu et al. (2010) apresentaram uma técnica de detecção *online* de *botnets* centralizadas. Primeiramente é realizada a transformação do tráfego em um *stream* de características multi-dimensional. A partir das instâncias é construída uma janela deslizante para agrupar no algoritmo *K-Means*. A partir dos grupos formados, é realizada a análise de similaridade entre as instâncias de cada grupo. Quando um grupo ultrapassar o limiar de similaridade, ou seja, possuir instâncias muito parecidas, o grupo é rotulado como suspeito. Entretanto, esta técnica não considera a mudança de conceito relacionada a distribuição dos dados, o que pode influenciar de forma negativa na capacidade de desempenho do algoritmo de agrupamento.

Yahyazadeh e Abadi (2015) apresentaram um sistema de detecção *online* de *botnets* baseado em reputação negativa, o BotGrab. O sistema rastreia o histórico de atividades coordenadas entre dispositivos na rede e calcula a reputação para cada dispositivo com base na participação das atividades. Ainda, este sistema estende o método BotOnus (Yahyazadeh e Abadi, 2012) que realiza a detecção *online* de *botnets* utilizando um algoritmo de agrupamento de tamanho fixo e análise de similaridade intra-*cluster*. Todavia, o sistema BotGrab utiliza de janelas de tamanhos fixos baseados em tempo e não considera a ocorrência de mudanças de conceitos que possam estar inclusas na janela pré-definida, isso pode influenciar de maneira negativa no resultado do sistema.

3.1.3 Discussão dos Trabalhos Relacionados à Detecção de Botnets

A Tabela 3.1 apresenta os trabalhos relacionados à detecção de *botnets* utilizando aprendizagem supervisionada e não supervisionada. A classificação é realizada levando em consideração as seguintes características: *(i)* capacidade de detecção *zero-day*, *(ii)* capacidade de

detecção em estágios iniciais, *(iii)* capacidade de detecção independente de encriptação de dados, *(iv)* capacidade de detecção não supervisionada, *(v)* capacidade de detecção *online* e *(vi)* capacidade de percepção de mudanças de conceito. A maioria dos trabalhos utilizando aprendizagem supervisionada (Kim et al., 2005; Hu et al., 2008; Stevanovic e Pedersen, 2014) não possuem capacidade de detecção em estágios iniciais, detecção *zero-day*, detecção *online* ou percepção de mudanças de conceito. Assim, os trabalhos utilizando aprendizagem não supervisionada são mais completos, pois apresentam estas características com maior frequência (Yu et al., 2010; Yahyazadeh e Abadi, 2015).

Referência	Detecção *Zero-day*	Detecção em Estágio Inicial	Independente de Encriptação	Detecção Não Supervisionada	Detecção *On-line*	Percepção da Mudança de Conceito
(Kim et al., 2005)			✓			
(Hu et al., 2008)			✓			
(Stevanovic e Pedersen, 2014)			✓			
(Carela-Español et al., 2016)	✓		✓		✓	
(Casas et al., 2019)	✓		✓		✓	✓
(Barthakur et al., 2015)			✓	✓	✓	
(Narang et al., 2014)	✓	✓	✓	✓-		
(Wu et al., 2018)	✓	✓	✓	✓		
(Wurzinger et al., 2009)			✓	✓	✓	
(Yu et al., 2010)	✓	✓	✓	✓	✓	
(Yahyazadeh e Abadi, 2015)	✓	✓	✓	✓	✓	
TRUSTED	✓	✓	✓	✓	✓	✓

Tabela 3.1: Classificação dos métodos de detecção de *botnet*

3.2 CONCEPT DRIFT OU MUDANÇA DE CONCEITO

As técnicas para detecção de mudança de conceitos podem ser divididas em duas categorias: *(i)* Explícita ou Supervisionada e *(ii)* Implícita ou Não Supervisionada (Sethi et al., 2016). As técnicas de detecção supervisionada de mudança de conceito dependem de dados rotulados para computar métricas como acurácia e F-*score* de forma *online*. Estas técnicas apontam mudanças no conceito quando acontece uma queda no desempenho do classificador. As técnicas de detecção não supervisionada de mudança de conceito se embasam nas propriedades das características extraídas dos dados. Identificar mudanças de conceito com base nas propriedades estatísticas dos dados torna os métodos propensos a falsos positivos, visto que detectam mudanças na distribuição estatística que não afetam a classificação. Entretanto, a aptidão de detecção sem rótulos compensa, visto que a rotulação de dados é custosa (Sethi e Kantardzic, 2017).

3.2.1 Detecção de Mudança de Conceito Supervisionada

As principais técnicas que detectam a mudança de conceito de forma supervisionada podem ser divididas em: *(i)* metodologia de análise sequencial, *(ii)* metodologia baseada em controle estatístico de processos e *(iii)* metodologia de monitoramento de distribuições baseada em janelas (Gama et al., 2014). Na metodologia de análise sequencial existe o método *Page-Hinckley Test* (PHT) (Page, 1954), que monitora as métricas de forma sequencial e cumulativa a fim de diferenciar os valores novos do resultado médio. Na metodologia baseada em controle estatístico de processos existem o *Drift Detector Method* (DDM) (Gama et al., 2004) e o *Early Drift Detector Method* (EDDM) (Baena-Garcıa et al., 2006), que monitoram a probabilidade e o desvio padrão de erros para indicar uma mudança de conceito. Por fim, na metodologia de monitoramento de distribuições baseadas em janelas existem os métodos *Adaptive Windowing* (ADWIN) e *Adaptive Sliding Windowing* (FIXWIN) (Bifet e Gavalda, 2007), que utilizam janelas de tamanho adaptativo e fixo, respectivamente, monitorando os resultados do classificador para diferenciar os resultados novos dos antigos.

Na metodologia de análise sequencial, o método PHT monitora a média cumulativa dos dados em relação a nova entrada de dados (Page, 1954). Ele mantém uma média cumulativa de todos os dados passados e compara com o resultado atual do classificador. Quando o resultado da média atual for maior que o limiar de detecção definido, uma mudança de conceito é sinalizada. No entanto, esta metodologia se adapta melhor a detecção univariada de mudança de conceito por sequência de medida de desempenho da métrica monitorada. Ainda, é necessária a entrega dos rótulos dos dados para a técnica obter as métricas de desempenho.

Na segunda metodologia, o método DDM monitora a probabilidade de erro em um tempo t, como p_t, além de monitorar o desvio padrão (s_t) (Gama et al., 2004). Quando a soma entre p_t e s_t atinge um determinado limiar é sinalizado um alerta. Este método pode apresentar tanto uma possível mudança de conceito quanto uma real mudança de conceito. Uma possível mudança de conceito é alertada quando a soma da probabilidade de erro com o desvio padrão for maior que o limiar y, e uma real mudança de conceito é alertada quando a soma da probabilidade de erro com o desvio padrão for maior que o limiar $(y * 1,5)$. O método EDDM (Baena-Garcıa et al., 2006) é uma extensão do DDM com o intuito de tratar as mudanças de conceito graduais. Além disso, o método EDDM monitora a quantidade de instâncias entre dois erros de classificação. O método assume que em ambientes estacionários a quantidade de instâncias entre dois erros subsequentes aumenta, baseando-se na quantidade de instâncias analisadas até no erro e_t e quantidade de instâncias analisadas até no erro e_{t-1}. Caso o aumento não aconteça, o método indica que houve uma mudança de conceito. Entretanto, esta metodologia apresenta métodos dependentes de rótulos que somente controlam os erros do classificador como principal métrica. Dessa forma, são recomendados para detecções onde existe a entrega dos rótulos *online* para a técnica de detecção, sendo inviável em *streams* de rede.

Diferente dos métodos supracitados que operam mantendo um resumo de todo o *stream*, na metodologia de monitoramento de distribuições baseadas em janelas, o método ADWIN utiliza blocos de instâncias para detectar mudanças de conceito (Bifet e Gavalda, 2007). O método mantém duas janelas $W = W_0 \cup W_1$, onde, W_0 mantém os dados de comparação e W_1 mantém um bloco de dados recentes. Assim, de forma adaptativa, o método aumenta o tamanho do bloco de dados enquanto não é detectado uma mudança de conceito. Quando duas janelas W_0 e W_1 apresentarem tamanho e diferença de desempenho suficientes, baseado no limite de Hoeffding, W_1 substitui W_0 e uma nova janela W_1 é iniciada. Porém, esta abordagem de aumento dinâmico do tamanho da janela permite o crescimento infinito da quantidade de instâncias dificultando a detecção de mudança de conceitos. Para isso, o método FIXWIN (Bifet e Gavalda, 2007) mantém duas janelas para comparação e detecção da mudança de conceito. No entanto, o bloco

de dados é mantido com tamanho fixo e deslizante a fim de controlar o tamanho máximo da janela de dados a ser comparada e utilizada para classificação. Contudo, semelhante aos outros métodos supervisionados, os métodos ADWIN e FIXWIN também necessitam dos rótulos para verificar os resultados da classificação. Entretanto, a abordagem de janelamento deslizante pode ser aplicada para a detecção não supervisionada como será discutido na próxima subseção.

3.2.2 Detecção de Mudança de Conceito Não Supervisionada

Os principais trabalhos relacionados à detecção de mudança de conceito não supervisionada podem ser classificados em três métodos: *(i)* detecção de novos agrupamentos (*novelty detection/clustering*) (Faria et al., 2013; Ryu et al., 2012; Hayat e Hashemi, 2010; Masud et al., 2010; Sethi et al., 2016), *(ii)* detecção de distribuições multivariadas (Lee e Magoules, 2012; Ditzler e Polikar, 2011; Kuncheva e Faithfull, 2013; Qahtan et al., 2015; Gözüaçık et al., 2019) e *(iii)* detecção de mudança de conceito dependente de modelos de classificação (Dredze et al., 2010; Lindstrom et al., 2010; Sethi e Kantardzic, 2017).

Em relação a detecção de mudança de conceito que busca novos agrupamentos, Faria et al. (2013) apresentaram o algoritmo MINAS. Este algoritmo utiliza micro agrupamentos, obtidos de forma incremental, para monitorar e se adaptar às novas distribuições de dados. Os conjuntos de novas instâncias que são marcados como desconhecidos são agregados aos agrupamentos mais similares. Hayat e Hashemi (2010) apresentaram o algoritmo DETECTNOD, que emprega a transformada discreta de cosseno para construir uma representação compacta dos agrupamentos. Isto é utilizado como informação para prover uma aproximação de vizinhos nas instâncias do *stream*. Dessa forma, as instâncias classificadas fora da representação compacta, são identificadas como mudança de conceito no mapa de características. Ryu et al. (2012) e Masud et al. (2010) apresentaram métodos que agrupam as instâncias e sempre que uma instância é classificada fora dos limites dos agrupamentos existentes ela é considerada uma detecção da mudança de conceito. A abordagem GC3 (Sethi et al., 2016) estende o conceito de micro-agrupamentos para utilizar densidade. Nesta abordagem uma mudança de conceito é identificada quando um novo agrupamento denso é formado. Todos estes trabalhos de detecção de novos agrupamentos são baseados na identificação de novas regiões no mapa de características que não foram identificadas previamente. Assim, estes trabalhos que buscam por novos agrupamentos apresentam problemas relacionados à dimensionalidade, pois utilizam cálculos de distância para identificar os novos agrupamentos. Além disso, estes métodos são adequados para cenários onde os dados possuem multi-classes, o que não acontece em cenários de detecção binaria (*bot* ou não).

Os trabalhos de detecção de distribuições multivariadas monitoram diretamente a distribuição de cada característica nos dados (Lee e Magoules, 2012; Ditzler e Polikar, 2011; Kuncheva e Faithfull, 2013; Qahtan et al., 2015). No geral, estes trabalhos utilizam blocos de instâncias antigos para sumarizar as informações e comparar com as distribuições de blocos atuais (Cha, 2007). Por exemplo, o método Mudança de Conceito (do inglês, *Change of Concept* - CoC) (Lee e Magoules, 2012) considera cada característica um *stream* de dados independente. Assim, a detecção de mudança de conceito compara dois blocos de instâncias, o atual e o de treino, por meio da avaliação de métricas (ex., média, desvio padrão) das distribuições dos blocos. A metodologia de detecção de mudança de conceito baseada na distância de Hellinger (do inglês, *Hellinger Distance Drift Detection Methodology* - HDDDM) utiliza dois blocos de instâncias para identificação da ocorrência da mudança de conceito. Assim, quando é apresentado um aumento na distância entre os dois blocos, atual e de treino, é sinalizada uma mudança de conceito.

Existem trabalhos que empregam a análise de componentes principais (do inglês, *Principal Component Analysis* - PCA) para a detecção de mudança de conceito em distribuições multivariadas de alta dimensionalidade (Kuncheva e Faithfull, 2013; Qahtan et al., 2015). Esta

técnica reduz o conjunto de características a serem analisadas pelo método de detecção. Kuncheva e Faithfull (2013) utilizaram a verossimilhança semi paramétrica para monitorar as mudanças nos dados projetados pelo PCA. Os autores propuseram o uso de apenas 10% dos menores autovalores do PCA, considerando ser suficiente para a efetividade na detecção de mudanças de conceito. Em contraste, Qahtan et al. (2015) mostraram que o PCA com os maiores autovalores é mais adequado, pois as características originais são melhor sumarizadas pelo topo do vetor que retem maior variância depois da redução. Contudo, estes trabalhos sofrem com a detecção de alarmes falsos, visto que são sensíveis a qualquer mudança em uma característica dos dados.

O método de detecção discriminativo de mudança de conceito (do inglês, *Discriminative Drift Detector* - D3) (Gözüaçık et al., 2019), que é a base da detecção de mudança de conceito deste trabalho, mantém dois blocos deslizantes de instâncias, dados antigos e novos. O método submete os dois blocos a uma classificação rotulando os dados antigos como "0"e os novos como "1". Dessa forma, treina um classificador para verificar se os dados são linearmente separáveis. Se o desempenho do classificador for alto, isto significa que os dados são separáveis e é sinalizada uma mudança de conceito. No entanto, estes métodos de detecção de distribuição multivariadas sinalizam muitos alarmes falsos, pois são sensíveis a mudanças em cada característica separadamente. Porém, para a detecção de mudança de conceito em *streams* multidimensionais são os mais adequados, devido ao tempo de resposta para a detecção da mudança de conceito ser rápida.

A detecção de mudança de conceito dependente de modelo de classificação verifica a probabilidade a *posteriori* estimada do classificador. Ou seja, dado um classificador probabilístico e uma instância de características X os métodos avaliam qual a probabilidade de acerto ou erro do classificador sobre a instância. Assim, Dredze et al. (2010) utilizaram o teste *Kolmogorov-Smirnov, Wilcox rank sum* e *sample t-test* para monitorar a área de incerteza do classificador *Support Vector Machine*. Lindstrom et al. (2010) usaram a divergência de Kullback-Leibler para realizar uma análise similar sobre as probabilidades a *posteriori* de um classificador. Por fim, Sethi e Kantardzic (2017) propuseram o algoritmo de detecção de mudança de conceito sobre densidade de margem (do inglês, *Margin Density Drift Detection* - MD3) que procura a quantidade de instâncias em uma área de incerteza de um classificador probabilístico. Isso serve como métrica para a detecção de mudança de conceitos. Dessa forma, as técnicas de detecção de mudança de conceito apresentam uma significante redução na sinalização de falsos positivos. Porém, são dependentes de classificadores probabilísticos, o que limita a escolha de algoritmos para a classificação dos dados.

3.2.3 Discussão dos Trabalhos Relacionados à Detecção de Mudanças de Conceito

A Tabela 3.2 apresenta a classificação dos métodos de detecção de mudança de conceito com relação a metodologia utilizada. Os trabalhos (Page, 1954; Gama et al., 2004; Baena-Garcıa et al., 2006; Bifet e Gavalda, 2007) apresentaram métodos para a detecção de mudanças de conceito supervisionadas com metodologias de análise sequencial, controle estatístico e baseadas em janelas. A metodologia de análise sequencial se adequa melhor a detecções univariadas, ou seja, em cenários onde é avaliada apenas uma métrica para detecção da mudança de conceito. A metodologia de controle estatístico monitora apenas o erro do classificador, o que retarda a detecção da mudança de conceito e não considera a distribuição dos dados. A metodologia baseada em janelas também monitora o erro do classificador, porém utiliza o conceito de janelas para separar os dados, assim, acarreta nos problemas da metodologia de controle estatístico além da possibilidade do incremento infinito da janela de dados. Por fim, todas as metodologias supervisionadas são dependentes da entrega de rótulos *online* ao método de detecção, o que é inviável em *streams* de rede.

Referência	Supervisionado			Não Supervisionado		
	Análise Sequencial	Controle Estatístico	Baseado em Janelas	Novos Agrupamentos	Dist. Multivariadas	Dependente de Modelo
(Page, 1954)	✓					
(Gama et al., 2004)		✓				
(Baena-Garcıa et al., 2006)		✓				
(Bifet e Gavalda, 2007)			✓			
(Faria et al., 2013)				✓		
(Ryu et al., 2012)				✓		
(Hayat e Hashemi, 2010)				✓		
(Masud et al., 2010)				✓		
(Sethi et al., 2016)				✓		
(Lee e Magoules, 2012)					✓	
(Ditzler e Polikar, 2011)					✓	
(Kuncheva e Faithfull, 2013)					✓	
(Qahtan et al., 2015)					✓	
(Gözüaçık et al., 2019)					✓	
(Dredze et al., 2010)						✓
(Lindstrom et al., 2010)						✓
(Sethi e Kantardzic, 2017)						✓

Tabela 3.2: Classificação dos métodos de detecção de mudança de conceito

Os trabalhos (Faria et al., 2013; Ryu et al., 2012; Hayat e Hashemi, 2010; Masud et al., 2010; Sethi et al., 2016; Lee e Magoules, 2012; Ditzler e Polikar, 2011; Kuncheva e Faithfull, 2013; Qahtan et al., 2015; Gözüaçık et al., 2019; Dredze et al., 2010; Lindstrom et al., 2010; Sethi e Kantardzic, 2017) são listados na Tabela 3.2 e apresentam métodos de detecção de mudança de conceito não supervisionada e são classificados entre as metodologias: metodologia de detecção de novos agrupamentos, metodologia de detecção de distribuições multivariadas e metodologia de detecções dependentes de modelo. A detecção de novos agrupamentos em geral utiliza algoritmos de *clustering* para agrupar as instâncias e quando detecta um novo agrupamento sinaliza uma mudança de conceito. Dessa forma, pelo uso dos algoritmos de *clustering* sofre com *streams* de alta dimensionalidade, o que ocorre em *streams* de rede. A detecção dependente de modelo utiliza a probabilidade a *posteriori* do classificador, e com base nisso, monitora a probabilidade de erros para detectar mudanças de conceito. Os métodos de detecção dependentes de modelo estão vinculados ao uso de classificadores probabilísticos o que limita os possíveis classificadores. A detecção de distribuições multivariadas monitora cada característica do *stream* como uma variável independente. Eles apresentam uma significante sinalização de falsos positivos. Porém, em *streams* multidimensionais com controle de janela apresentam resultados com alta taxa de detecção e com entrega *online*. Assim, o algoritmo que serviu como base para a detecção de mudança de conceito nesta dissertação segue a metodologia de detecção de distribuições multivariadas e controle de janelas.

3.3 RESUMO

Neste capítulo, os trabalhos relacionados foram listados e discutidos com base nas características e técnicas utilizadas. Entre as abordagens discutidas estão métodos de detecção de *botnets* utilizando aprendizagem supervisionada e não supervisionada, bem como, estão os métodos de detecção supervisionada e não supervisionada de mudança de conceito. Na primeira seção, foram apresentados os trabalhos relacionados à detecção de *botnets*. Na segunda seção, foram apresentados os trabalhos relacionados à detecção de mudanças de conceito. A partir deste capítulo é possível observar a carência de trabalhos considerando as mudanças de conceito na detecção não supervisionada de *botnets*.

4 SISTEMA TRUSTED

Este capítulo apresenta o sistema de detecção e identificação de *botnets* denominado de TRUSTED (do inglês, *an on-line sysTem foR UnSupervised boTnEt Detection*). O sistema TRUSTED complementa as estratégias e soluções existentes de detecção e identificação de *bots*, trazendo principalmente a característica de ser *online*, isto é, o sistema processa os dados um a um logo que são obtidos. O sistema atua independente do conhecimento prévio do comportamento da rede e do conhecimento das assinaturas dos *bots* e leva em consideração possíveis mudanças de conceito da rede. O sistema proposto detecta e identifica *bots* na rede construindo blocos de instâncias de fluxo de rede obtidas de forma dinâmica para detecção de mudanças de conceito, agrupamento e análise de similaridade. Neste trabalho, uma instância refere-se a uma amostra de fluxo de rede obtido pelo agrupamento dos pacotes pertencentes ao fluxo definido a partir da tupla ⟨IP de origem, IP de destino, Porta de origem, Porta de destino e protocolo de comunicação⟩. O bloco de instâncias refere-se ao conjunto de instâncias (amostras) mais atuais obtidas da rede. O bloco de instâncias é submetido a um algoritmo de agrupamento para distinguir e agrupar as instâncias de forma a identificar os agrupamentos que possuem tráfego malicioso. Com as instâncias agrupadas e os *clusters* formados, o sistema calcula a similaridade intra-*cluster* para identificar os *clusters* que possuem instâncias com tráfego de *bots*. Dessa forma, a Seção 4.1 apresenta uma visão geral do sistema e seus principais aspectos. A Seção 4.2 detalha o sistema TRUSTED, suas características, a dinâmica de obtenção dos blocos de instâncias, a maneira de detecção de mudança de conceito, a forma de agrupamento das instâncias, a identificação dos *bots* e o funcionamento do sistema como um todo. A Seção 4.3 apresenta o resumo do capítulo.

4.1 VISÃO GERAL

O sistema TRUSTED distingue-se por detectar e identificar *botnets* na rede com base no agrupamento das instâncias de fluxo de rede e a similaridade entre as instâncias de um mesmo grupo considerando possíveis mudanças de conceito no *stream*. Isso é realizado sem a necessidade de conhecimento prévio de comportamento da rede ou assinaturas de *botnets*. Por meio da identificação dos *bots* das *botnets*, é possível disparar gatilhos para a reconfiguração dos equipamentos de monitoramento e segurança da rede, como um *firewall*, a fim de evitar ou conter o ataque. Nesta dissertação são consideradas quatro premissas: *(i)* o sistema considera um cenário de obtenção dos dados de forma *online*, *(ii)* considera a extração de características de amostras do tráfego de rede (ex., um fluxo a cada cem), *(iii)* o sistema é independente de conhecimento prévio para a classificação, e *(iv)* o sistema detecta mudanças de conceito para adaptar a classificação às distribuições atuais do tráfego. Assim, o sistema TRUSTED realiza a detecção e identificação de *botnets* por meio de etapas que consistem na medição e preparação dos dados, janelamento e detecção de mudança de conceito, agrupamento de instâncias, cálculo da similaridade entre as instâncias em um mesmo grupo, análise da similaridade, detecção e identificação de *botnets*.

A Figura 4.1 ilustra a arquitetura do sistema e as etapas para a detecção e identificação de *botnets*. O método segue três etapas: *(i)* pré-processamento, *(ii)* identificação de mudanças de conceito, e *(iii)* detecção de *botnet*. A preparação dos dados consiste em coletar o tráfego de rede *online*, separar em fluxos sobre a tupla ⟨IP de origem, IP de destino, porta de origem, porta de destino e protocolo de comunicação⟩. A partir dos fluxos são extraídas as características de volume e *timestamps*, como quantidade de pacotes e quantidade de *bytes*, pois segundo Garcia

et al. (2014) volume e *timestaps* são características com maior ganho de informação. Essas características devem formar um *stream* de dados. Um *stream* é definido como a transmissão ou recebimento de dados pela Internet como uma sequência constante e contínua. Com base nas características extraídas, é realizado o janelamento deslizante e a detecção de mudança de conceito sobre os dados, além disso, o bloco de dados (instâncias de fluxo de rede) construído pela segunda etapa é utilizado para o processo de agrupamento. Após a construção do bloco de instâncias e agrupamento pelo algoritmo de agrupamento, o sistema realiza a análise da similaridade intra-*cluster*, calculando a proximidade entre a instâncias de um mesmo *cluster* e deve apontar os *clusters* que possuem *bots*.

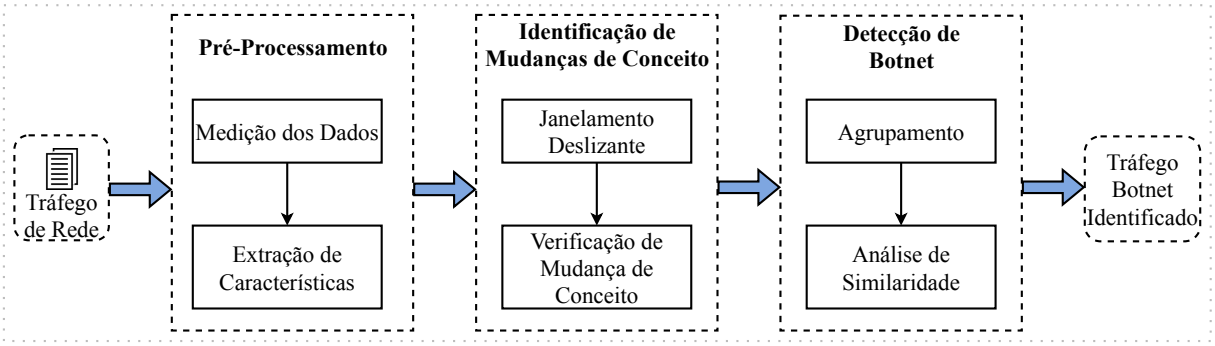


Figura 4.1: Etapas de processamento do sistema TRUSTED

4.2 DETALHAMENTO DO SISTEMA TRUSTED

Esta seção apresenta os detalhes de cada uma das três etapas do sistema TRUSTED. A Subseção 4.2.1 apresenta como funciona a etapa de extração dos dados a partir do tráfego de rede. A Subseção 4.2.2 demonstra o funcionamento da etapa de construção do janelamento deslizante e detecção de mudanças de conceito. A Subseção 4.2.3 mostra a forma de agrupamento das instâncias. A Subseção 4.2.4 expõe a forma de cálculo da similaridade intra-*cluster* dos agrupamentos para identificar *botnets*. Por fim, a Subseção 4.3 resume o capítulo.

4.2.1 Extração dos dados

A etapa de medição e preparação dos dados é responsável por *(i)* coletar o tráfego da rede e *(ii)* separar o tráfego por fluxos e extrair as características dos fluxos. Essa etapa é necessária para obter as características do tráfego e organizar os dados para detectar mudanças de conceito no comportamento da rede, além de servir como entrada para o processo de agrupamento. Ainda, assume-se que a coleta dos dados é realizada por uma interface de rede no modo promíscuo no *hardware* em que o sistema estiver atuando.

Primeiramente, o tráfego coletado é separado em fluxos $F = \{p_1, p_2, p_3, ..., p_t\}$ pela tupla ⟨IP de origem, IP de destino, porta de origem, porta de destino e protocolo de comunicação⟩. Em um fluxo F, p é um pacote e t é o momento de captura do pacote. A partir do conjunto de pacotes de F é extraído o conjunto de características estatísticas do fluxo, definidas por $X^F = \{x_1, x_2, x_3, ..., x_i\}$, neste, x é uma característica extraída do fluxo e i é o número de características extraídas. Essas características devem formar o bloco de dados $W = \{X_1^F, X_2^F, X_3^F, ..., X_z^F\}$ que é mantido de forma deslizante e com tamanho fixo.

4.2.2 Janelamento Deslizante e Detecção de Mudança de Conceito

Nesta etapa do sistema TRUSTED, o módulo recebe as características que formam o conjunto de dados provenientes do módulo detalhado na Subseção 4.2.1. O tratamento dos dados será feito em dois passos: *(i)* janelamento deslizante e *(ii)* detecção de mudança de conceito. O bloco de dados resultante do janelamento deslizante é constituído pelo conjunto de características extraídas dos fluxos de rede e possui um tamanho fixo pré-determinado. Esse bloco de dados será utilizado para a detecção de mudanças de conceito e para a etapa de agrupamento que irá agrupar as instâncias a fim de encontrar *botnets*.

Para a construção do bloco de dados W na fase do janelamento deslizante o módulo recebe como entrada um *stream* de dados do tipo X^F. Para a detecção de mudanças de conceito será usado o método de detecção de mudança de conceito não linear discriminativa (do inglês, *Non-linear Discriminative Drift Detector* - D3N) adaptado de Gözüaçık et al. (2019), a adaptação se dá por meio da troca do classificador discriminativo, originalmente um algoritmo de regressão linear para uma árvore de decisão. O objetivo da adaptação é detectar mudanças de conceito não lineares na distribuição dos dados. Este método se destaca por ser um método não supervisionado de detecção de mudanças de conceito e ser classificado entre os detectores de distribuições multivariadas como um dos mais recomendados para *streams* multidimensionais, além disso, o método foi comparado com o método supervisionado ADWIN.

Dessa forma, o Algoritmo 1 apresenta o funcionamento do método. O método de detecção de mudanças de conceito recebe como parâmetros de entrada o *stream*, a quantidade mínima de instâncias (δ), um percentual de novas instâncias (σ) e um limiar (β) para a detecção da mudança de conceito. Como resultado o algoritmo retorna verdadeiro ou falso para a ocorrência de mudança de conceito. No Algoritmo 1 linha 1 é inicializado o bloco de dados com tamanho $\delta(1+\sigma)$ que deve ser preenchido para ocorrer a verificação de mudança de conceito. No *stream* de dados, as instâncias chegam uma a uma para a verificação, no Algoritmo 1 linha 4 é verificado se o bloco de dados está cheio, se não estiver cheio é adicionada a instância ao bloco de dados, se o bloco W estiver cheio com $\delta(1+\sigma)$ o método aplica rótulos às instâncias. Assim, no Algoritmo 1 linhas 8 e 9, δ instâncias recebem rótulo 0 e σ instâncias recebem rótulo 1, respectivamente. Essas instâncias são submetidas a uma classificação supervisionada utilizando uma árvore de decisão na linha 10 do Algoritmo 1. Uma mudança de conceito é encontrada quando o resultado do classificador conseguir distinguir entre as instâncias com desempenho superior a β verificado na linha 11 do Algoritmo 1, pois, segundo Gözüaçık et al. (2019), se é possível classificar o conjunto com alta assertividade, os dados são separáveis no mapa de características. Assim, se ocorreu a mudança de conceito no comportamento da rede é sinalizada uma mudança de conceito e eliminadas δ instâncias do final do bloco W, apresentados nas linhas 12 e 13 do Algoritmo 1. Caso contrário, não é sinalizada a mudança de conceito e são eliminadas $\delta * \sigma$ instâncias do final do bloco W, apresentados nas linhas 15 e 16 do Algoritmo 1. Na Figura 4.3, é possível identificar como deve ser o comportamento dos dados para a ocorrência da mudança de conceito.

O desempenho do classificador discriminante do método é obtido pelo cálculo da área abaixo da curva de característica de operação do receptor (do inglês, *area under the receiver operating characteristic curve* - AUC). Esta métrica é obtida relacionando as taxas de verdadeiros positivos no eixo Y e falsos positivos no eixo X, quanto maior a área abaixo da curva gerada melhor o resultado do classificador, na Figura 4.2 pode-se verificar um exemplo de AUC. Quando uma mudança de conceito é encontrada um conjunto de δ instâncias é removido do fim do bloco de dados W de tamanho $\delta(1+\sigma)$. O bloco de dados é novamente construído de forma incremental até alcançar o tamanho completo para realizar a nova comparação para detecção da mudança de conceito. Quando não ocorre uma mudança de conceito, somente é eliminado um conjunto predefinido de instâncias do fim do bloco conforme mostra a Figura 4.3.

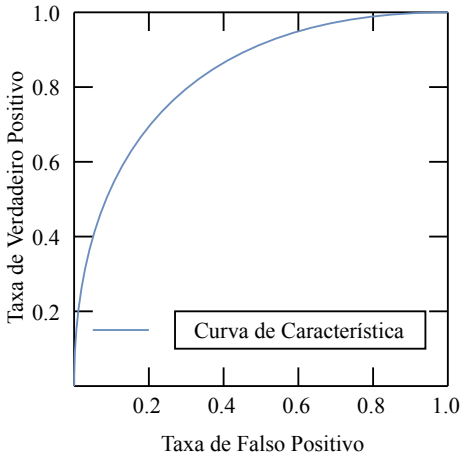


Figura 4.2: Exemplo de AUC

Algoritmo 1 D3N Non-linear Discriminative Drift Detector

Entrada: *stream* $\{X_n^F\}_{n=1}^N$, δ, σ, β
Resultado: Verdadeiro ou falso para a ocorrência de mudança de conceito

```
Inicializa a janela W onde |W| = δ(1 + σ)
Inicia Classificador discriminativo C  // Árvore de Decisão
for all (X, y) em stream do
    if W não estiver cheio then
        W ← W ∪ X  // adiciona X no início de W
    else
        S é um vetor de s, |W| = |S| // s é uma variável temporária de rótulos
        s = 0 for W[1,δ]  // rótulo velho (0) e novo (1)
        s = 1 for W[δ + 1, final]
        Treine C(W, S)  // treine C com os rótulos S de W
        if AUC(C,S ≥ β) then
            drift = verdadeiro
            Elimine δ instâncias do final de W
        else
            drift = Falso
            Elimine δ ∗ σ elementos do final de W
        end if
    end if
end for
```

A Figura 4.3 ilustra uma janela deslizante W de tamanho 8 (apenas como exemplo). A janela deslizante contém instâncias antigas (em azul) e instâncias novas (em verde). Considerando a regra de o primeiro a entrar é o primeiro a sair, instâncias novas significam aquelas recentemente adicionadas a W. A janela completa tem um tamanho variável K definido como $K = \delta * (1 + \sigma)$, contendo δ instâncias antigas e uma porcentagem σ de novas instâncias de dados de rede. Enquanto a janela não estiver cheia, o sistema continua adicionando novas instâncias X^F para preenchê-la. Quando a janela enche, o sistema TRUSTED executa a identificação de mudança de conceito. Se uma mudança de conceito for encontrada, as instâncias δ antigas são todas removidas, mantendo apenas as mais recentes em W. Caso contrário, se não for encontrado uma mudança de conceito, o sistema removerá apenas uma fração predefinida da quantidade de

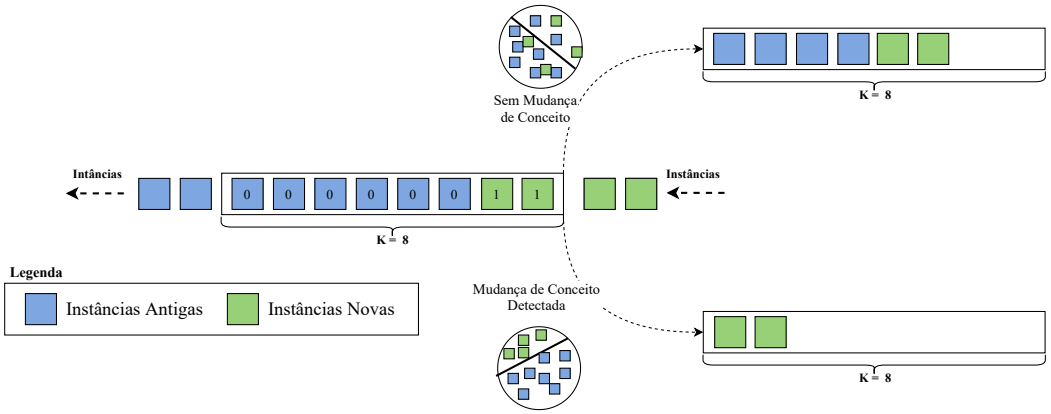


Figura 4.3: Comportamento dos dados para a detecção da mudança de conceito. Adaptado de Gözüaçık et al. (2019)

instâncias antigas. As instâncias δ antigas em W são rotuladas por "0", enquanto as instâncias novas são rotuladas por "1".

4.2.3 Clustering

Nesta etapa, o sistema TRUSTED agrupa o bloco de instâncias recebidas da fase de janelamento e detecção de mudança de conceito. A entrada será um bloco W de instâncias X^F de tamanho fixo, variando de 1 até $\delta(1+\sigma)$. As instâncias serão agrupadas de forma incremental, isto é, uma a uma, enquanto o *stream* existir. Esta subseção detalha o algoritmo de agrupamento e como são realizados os cálculos de agrupamento, além de demonstrar o comportamento esperado para a detecção de *botnets*.

Para realizar o agrupamento, o sistema TRUSTED recebe como base o bloco de dados construído pela fase de janelamento deslizante e detecção de *drift* exposto da Subseção 4.2.2. Dessa forma, utilizando o algoritmo (*Hierarchical Agglomerative Clustering* - HAC) exposto no Algoritmo 2, as instâncias são agrupadas de forma hierárquica com um valor γ de distância máxima entre as instâncias. Assim no Algoritmo 2 na linha 1, é inicializado o grupo de agrupamentos vazio. Na linha 3 do Algoritmo 2 cada instância é definida como um agrupamento e na linha 5 os agrupamentos são adicionados na árvore hierárquica T. Assim, enquanto o número de agrupamentos for maior que 1, na linha 7 do Algoritmo 2 são definidos os pares mais próximos entre todos os agrupamentos dentro da distância máxima γ. Nas linhas 8 e 9 do Algoritmo 2 são removidos os agrupamentos antigos e adicionados os novos agrupamentos criados, respectivamente. Na linha 10 do Algoritmo 2 é adicionado o conjunto de agrupamentos A na árvore hierárquica T.

Algoritmo 2 HAC: Hierarchical Agglomerative Clustering

Entrada: *stream* $\{X_n^F\}_{n=1}^N$, γ
Resultado: Árvore T Hierárquica

A $\leftarrow \oslash$ // Inicializa grupo de agrupamentos como vazio
for all (X, y) em stream **do**
    A $\leftarrow$ A $\cup$ $\{\{X_n^F\}\}$ // Adiciona cada instância a um agrupamento
end for
T $\leftarrow$ A // Adiciona os agrupamentos à árvore
while | A | > 1 **do**
    $G_1^*, G_2^* \leftarrow$ min DIST(G_1, G_2) // Agrupa os pares com as menores distâncias
    A $\leftarrow$ (A $\setminus$ $\{G_1^*\}$) $\setminus$ $\{G_2^*\}$ // Remove as instâncias que estavam sozinhas e agora estão agrupadas
    A $\leftarrow$ A $\cup$ $\{G_1^* \cup G_2^*\}$ // Adiciona o agrupamento ao grupo de agrupamentos
    T $\leftarrow$ T $\cup$ $\{G_1^* \cup G_2^*\}$ // Adiciona o agrupamento à árvore
end while

4.2.4 Similaridade Intra-Cluster

Esta etapa do sistema TRUSTED toma como entrada o bloco de dados e os rótulos da classificação do algoritmo de agrupamento. A partir disso, a equação 4.1 calcula a distância entre todas as instâncias de um mesmo *cluster*. O valor $\Delta(S)$ resultante da equação 4.1 é comparado com um limiar χ, se o valor da distância média do agrupamento for menor ou igual a χ, a instância é considerada como *bot* (Yahyazadeh e Abadi, 2012), caso contrário é considerado tráfego legítimo. Esta comparação com o limiar de similaridade indica que o conjunto de instâncias do agrupamento é muito parecido, então é rotulado como *bot*.

$$\Delta\ (S) = \frac{1}{|S| * (|S| - 1)} * \sum \{d\ (x, y)\} \tag{4.1}$$

$$x, y \in S \tag{4.2}$$

4.3 RESUMO

Este capítulo apresentou o sistema de detecção de *botnets* denominado TRUSTED, detalhando suas características e comportamento. Também demonstrou o funcionamento das três etapas do sistema que são: *(i)* extração dos dados, *(ii)* janelamento deslizante e detecção de mudanças de conceito, e *(iii)* agrupamento, análise da similaridade intra-*cluster* e identificação dos *bots*. Sendo que a primeira corresponde ao processo de obtenção de dados, a segunda está relacionada a construção da janela de dados e detecção de mudanças de conceito e a terceira está relacionada a dinâmica de agrupamento das instâncias, ao processo de análise da similaridade intra-*cluster* entre os *clusters* formados e a detecção de *botnets* na rede.

5 AVALIAÇÃO DE DESEMPENHO

Este capítulo apresenta a metodologia de avaliação de desempenho do sistema TRUSTED e responde as seguintes questões de pesquisa: *(i)* Qual o impacto da detecção de mudanças de conceito na detecção de *botnets*? *(ii)* Qual a eficácia ao implementar um sistema de detecção *on-line* de *botnets* de forma não supervisionada levando em consideração possíveis mudanças de conceito? Como descrito na Seção 3.1, os trabalhos de detecção *on-line* não supervisionada de *botnets* não consideram as possíveis mudanças de conceito no *stream* de dados. Portanto, esta dissertação avalia os efeitos da ocorrência de mudanças de conceito em um cenário de ataques gerados por *botnets*, bem como avalia e compara o sistema proposto. Para a avaliação do sistema foram utilizados dois conjuntos de dados o Botnet 2014 (Beigi et al., 2014) e o CTU-13 (Garcia et al., 2014), descritas na Seção 5.1. Em sequência a Seção 5.2 apresenta as configurações dos ambientes de execução. Após, a Seção 5.3 expõe as métricas de avaliação consideradas. Por fim, a Seção 5.4 apresenta e discute os resultados da avaliação de desempenho.

5.1 CARACTERÍSTICAS DAS BASES DE DADOS

Para a avaliação do sistema TRUSTED foram utilizados dois conjuntos de dados, *Botnet* 2014 (**dataset 1**) e CTU-13 (**dataset 2**). Esta seção descreve as principais características dos conjuntos de dados e a preparação dos dados. **Dataset 1** (*Botnet* 2014) apresenta generalismo, realismo e representatividade de ataques de *botnet*. Ele atinge o generalismo usando 16 variações de *botnets* com base em 3 tipos de protocolos de aplicação. A Tabela 5.1 apresenta a variação dos *botnets* (nome), tipo de protocolo de aplicação e a porcentagem de dados gerados de cada botnet. O realismo é fornecido pela mistura de tráfego real coletado de um ambiente não controlado e tráfego gerado de um ambiente controlado. O conjunto de dados apresenta representatividade ao criar um conjunto de coleções feitas com dados não sobrepostos de tráfego real e artificial (Beigi et al., 2014). **Dataset 2** (CTU-13) contém botnet e tráfego normal capturado na Universidade Técnica Tcheca. Possui 13 cenários de ataque. Ele usa diferentes *malwares* de botnet e uma quantidade de *bots* (Garcia et al., 2014). Para nossa avaliação de desempenho, consideramos os cenários 4, 5, 10 e 11, chamados de CTU-4, CTU-5, CTU-10 e CTU-11. A Tabela 5.2 apresenta o *malware* usado, a duração em tempo e o número de *bots* de cada cenário considerado.

5.1.1 Manipulação dos Dados

O sistema TRUSTED foi avaliado tendo como entrada os conjuntos de dados. Ele agrupa o tráfego de rede em instâncias de fluxo F seguindo a tupla $\langle$ IP de origem, IP de destino, porta de origem, porta de destino e protocolo de comunicação $\rangle$. Em seguida, o sistema extrai características estatísticas das instâncias de fluxo. As características são o número de bytes enviados e recebidos, o horário de início e término do envio de pacotes, o horário de início e término do recebimento do pacote, o intervalo entre o horário de início e término do envio de pacotes e o intervalo entre o início e o fim de receber pacotes. As Figuras 5.2, 5.3, 5.4 e 5.5 apresentam a distribuição de pacotes enviados e recebidos e as ocorrências de ataque nos datasets.

Especificamente, a Figura 5.1 mostra a distribuição de pacotes do conjunto de dados do *Botnet* 2014 e as Figuras 5.2, 5.3, 5.4 e 5.5 mostram a distribuição do conjunto de dados CTU-13 (cenários 4, 5, 10 e 11). As linhas verde, azul e vermelha representam os pacotes recebidos, os pacotes enviados e os pacotes de tráfego de ataque, respectivamente. Embora ambos os conjuntos

Nome da Botnet	Arquitetura	% de Dados
Neris	IRC	5,67
Rbot	IRC	0,018
Menti	IRC	0,62
Sogou	HTTP	0,019
Murlo	IRC	1,06
Virut	HTTP	12,80
NSIS	P2P	0,165
Zeus	P2P	0,109
SMTP Spam	P2P	4,72
UDP Storm	P2P	9,63
Tbot	IRC	0,283
Zero Access	P2P	0,221
Weasel	P2P	9,25
Smoke Bot	P2P	0,017
Zeus Control	P2P	0,006
ISCX IRC Bot	P2P	0,387

Tabela 5.1: Distribuição das Botnets na base de dados

Tabela 5.2: CTU-13 Dataset

CTU-13 Cenários	*Botnet*	Duração (h)	# *bots*
Cenário 4 (CTU-4)	Rbot	4.21	1
Cenário 5 (CTU-5)	Virut	11.63	1
Cenário 10 (CTU-10)	Rbot	4.75	10
Cenário 11 (CTU-11)	Rbot	0.26	3

de dados apresentassem tráfego de ataque, eles tinham comportamentos diferentes. Por exemplo, o *Botnet* 2014 e CTU-10 apresentaram tráfego de ataque mais denso em torno de 70 % e 90 % dos dados, enquanto os outros conjuntos de dados mostraram tráfego de ataque disperso. Assim, o sistema TRUSTED tem como objetivo detectar todas as alterações de tráfego de dados e, em seguida, os ataques de botnet.

5.2 AMBIENTES DE EXECUÇÃO

Realizamos uma avaliação de **configuração empírica** para determinar os parâmetros de configuração ideais, uma vez que influenciam nos resultados de desempenho e operação do sistema. Esses parâmetros envolvem o tamanho da janela, porcentagem de novos dados, limite de variação de conceito e distância de agrupamento. Assim, alteramos os valores dos parâmetros ao longo das avaliações para selecionar a melhor configuração. Posteriormente, criamos outras duas configurações de ambientes, *(i)* **offline** e *(ii)* **online**, empregando a melhor configuração obtida por meio da avaliação de configuração empírica (veja os resultados em Seção 5.4). Como o TRUSTED funciona em ambientes online, criamos fluxos de tráfego de dados usando os conjuntos de dados para detectar as mudanças de conceito e *botnets*. Os fluxos de tráfego de dados contêm as instâncias de fluxo. Com base nos fluxos, criamos as janelas deslizantes de dados e as analisamos de forma incremental. A implementação usou bibliotecas baseadas

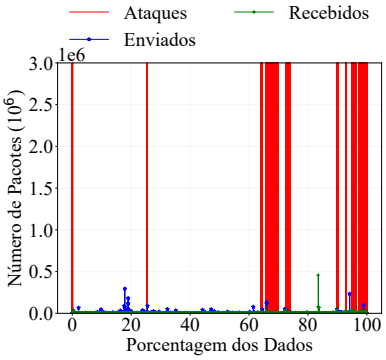


Figura 5.1: Botnet 2014 Ataques

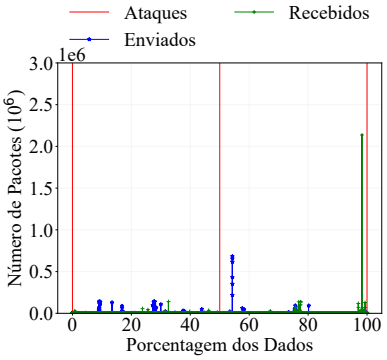


Figura 5.2: CTU-4 Ataques

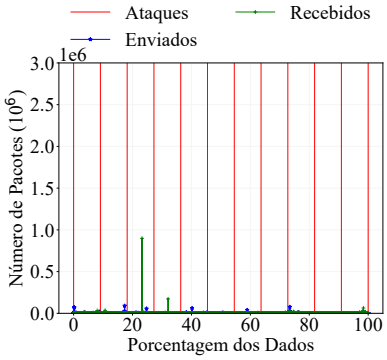


Figura 5.3: CTU-5 Ataques

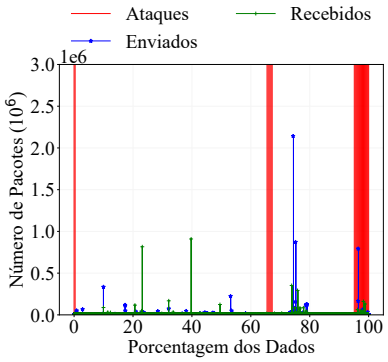


Figura 5.4: CTU-10 Ataques

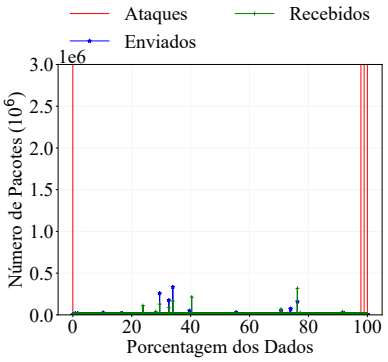


Figura 5.5: CTU-11 Ataques

em Python: Scapy[1] v2.4.4, scikit-learn[2] v0.23, scikit-multiflow[3] v0.5.3 e Pandas[4] v1.0.5. O ambiente **offline** foi implementado e avaliado em um sistema operacional Ubuntu 18.04 LTS com um processador Intel Core i7 8750H e 16 GB de RAM. Ele usou os conjuntos de dados como entrada para construir o fluxo de dados, ou seja, a sequência de instâncias de fluxo, chegando um por um para o sistema. O ambiente *offline* foi utilizado para avaliar o impacto das mudanças de conceito comparando com métodos da literatura. Além disso, avaliamos uma configuração empírica do sistema TRUSTED para definir os parâmetros ideais do sistema (ex., tamanho da janela). Finalmente, comparamos as instâncias do sistema TRUSTED (comparação do estado da arte): TRUSTED c/ HAC (o sistema com algoritmo de aprendizagem não supervisionado, análise de similaridade e identificação de desvio de conceito), FIXO c/ HAC (o sistema com algoritmo de aprendizagem não supervisionado, análise de similaridade e janelas fixas, mas ignorando a identificação de desvio de conceito) (Yu et al., 2010; Yahyazadeh e Abadi, 2015) e TRUSTED c/ ARF (o sistema com classificador *Adaptive Random Forest* supervisionado e com identificação de mudanças de conceito) (Casas et al., 2019).

O ambiente **online** seguiu uma avaliação experimental em uma rede de área ampla de ponta a ponta apoiada pela Rede Nacional de Ensino e Pesquisa[5] (RNP) do Brasil (parceria com o projeto MENTORED). A RNP tem seu *backbone* por todos os estados brasileiros, chamados de ilhas de recursos computacionais. Nosso ambiente considerou duas ilhas, *Rio Grande do Sul* (RS) e *Minas Gerais* (MG). Cada ilha contém um roteador de borda e um servidor de virtualização com Intel (R) Xeon (R) Gold, modelo 5118 e 256 GB de memória RAM DDR4. As ilhas RS e

[1]Scapy, https://scapy.net/, Último acesso em outubro de 2020
[2]Scikit-learn, https://scikit-learn.org/stable/, último acesso em novembro de 2020.
[3]Scikit-multiflow, https://scikit-multiflow.github.io/, último acesso em novembro de 2020.
[4]Pandas, https://pandas.pydata.org/, Último acesso em novembro de 2020.
[5]RNP: https://www.rnp.br/, Último acesso em novembro de 2020.

MG representam o cliente (rede alvo) e o servidor, respectivamente. O servidor de virtualização RS emulou uma rede local com contêineres *Docker* do Ubuntu para cada cliente para reproduzir o tráfego do cliente. Dessa forma, para realizar a avaliação reproduzimos os cenários do CTU-13 neste ambiente, simulando uma conexão real de Internet entre cliente e servidor. O cliente e o servidor enviam e recebem tráfego pelo *script* Scapy Python seguindo os cenários da base de dados CTU-13. No roteador de borda do cliente, executamos a detecção de botnet. Finalmente, neste ambiente, avaliamos TRUSTED c/ ARF e TRUSTED c/ HAC, ignorando o FIXO c/ HAC devido as suas propriedades estáticas e a capacidade de detecção com queda relevante observada nos resultados do ambiente offline.

5.3 MÉTRICAS DE AVALIAÇÃO

Para avaliar o sistema proposto, foram utilizadas diferentes métricas de desempenho. Em particular, essas métricas avaliam o comportamento dos algoritmos de classificação. Elas envolvem a acurácia, precisão, *recall* e *F1-score*. Tais métricas consideram as taxas de verdadeiro positivo (VP), verdadeiro negativo (VN), falso positivo (FP) e falso negativo (FN). Dessa forma, estatisticamente a acurácia se refere à proporção de instâncias de fluxos classificados corretamente em relação a todas as amostras de tráfego. A precisão (p) estima a porcentagem de verdadeiros positivos dentre todos os exemplos de tráfego classificados como positivos $(VP/(VP+FP))$. O *recall* (r) ou revocação consiste da porcentagem de verdadeiros positivos dentre todos os exemplos cuja classe esperada é a positiva $(VP/(VP+FN))$. Por fim, o *F1-Score* faz uma relação entre as medidas de precisão e *recall* por meio da estimação da média harmônica $(2rp/(r+p))$. Portanto, os resultados dos classificadores se embasam nessas métricas.

5.4 RESULTADOS

Esta Seção demonstra os resultados obtidos na avaliação do impacto das mudanças de conceito em cenários de ataque e da avaliação de desempenho do sistema TRUSTED. São apresentados os comportamentos dos métodos de detecção de mudanças de conceito na classificação de dados contendo tráfego normal e ataques. Também, é exposto o comportamento do sistema TRUSTED na avaliação empírica, assim como são apresentadas e discutidas as comparações do sistema TRUSTED considerando um cenário sem detecção de mudanças de conceito e um cenário supervisionado com detecção de mudanças de conceito. Os gráficos gerados foram obtidos a partir das métricas obtidas durante as avaliações.

5.4.1 Avaliação dos Métodos de Detecção de Mudanças de Conceito

O método D3N foi comparado com dois métodos de detecção de mudança de conceito, o D3 não supervisionado e o ADWIN supervisionado, em todas as comparações os métodos utilizaram o classificador Árvore de Hoeffding para a detecção de botnet. Os resultados seguem duas abordagens: avaliação da acurácia dos classificadores durante a execução e a média de todas as métricas no contexto geral do conjunto de dados.

Em relação à avaliação da acurácia no decorrer da base, a Figura 5.6 apresenta o comportamento de todas as execuções a partir do classificador Árvore de Hoeffding sozinho e com os métodos de detecção de mudanças de conceito ADWIN, D3 e D3N. A linha laranja na Figura 5.6 mostra o desempenho do classificador Árvore de Hoeffding sem detecção de mudança de conceito, esse resultado apresenta a maior queda de desempenho quando enfrentando o ataque presente na base de dados. As linhas vermelha e azul referem-se ao desempenho do classificador

Árvore de Hoeffding com o detector supervisionado de mudanças de conceito ADWIN e detector não supervisionado de mudanças de conceito D3, respectivamente. Os métodos ADWIN e D3 apresentaram resultados melhores em relação ao classificador sem detecção de mudança de conceito, explicitando uma adaptação no momento dos ataques. A linha verde na Figura 5.6 mostra o desempenho do classificador Árvore de Hoeffding com o detector D3N, este alcançou os melhores resultados na comparação alcançando maior adaptabilidade nos momentos de ataque da base de dados. Contudo, é possível observar um comportamento característico dos cenários em relação a base de dados, onde, por volta de 60% da base ocorre a queda do desempenho da classificação. Porém os cenários utilizando métodos de detecção de mudanças de conceito se mostraram aptos a adaptação a nova distribuição dos dados, recuperando-se das quedas de desempenho de forma mais rápida em relação ao cenário que não considera mudanças de conceito. Ainda, em relação ao desempenho dos métodos de detecção de mudanças de conceito, é destacável o desempenho da classificação utilizando o método D3N adaptado neste trabalho a partir do método D3 (Gözüaçık et al., 2019). A adaptação do modelo de aprendizagem utilizando o D3N ocorre de forma mais rápida e com menos perda de desempenho.

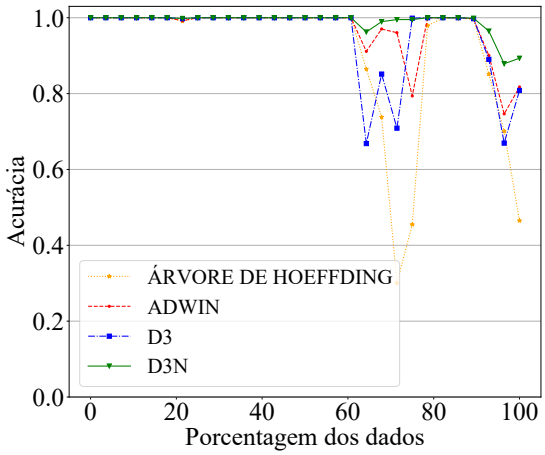


Figura 5.6: Comparação de Acurácia

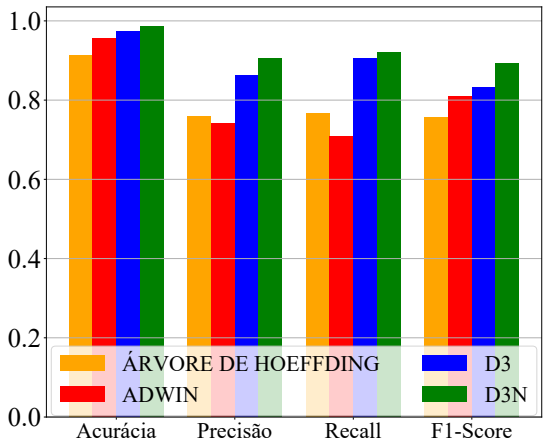


Figura 5.7: Comparação Geral

Em relação à avaliação das métricas no contexto geral da classificação da base Botnet 2014, a Figura 5.7 apresenta os resultados de desempenho. As métricas consideradas foram: acurácia, precisão, *recall* e *F1-Score*, todas apresentadas em porcentagem. Dessa forma, sustentando os resultados da acurácia no decorrer da base já apresentados, pode-se observar na Figura 5.7, que de forma geral a utilização dos métodos de mudança de conceito considerados incrementam o desempenho de classificação da Árvore de Hoeffding. Contudo, entre os métodos de detecção de mudanças de conceito, o método D3N adaptado neste trabalho supera em desempenho os dois outros cenários avaliados.

Este estudo de caso mostra a melhora quando utilizando as técnicas de detecção de desvio de conceito. Além disso, o método de detecção de conceito não supervisionado D3 obteve pior desempenho do que os métodos não supervisionados ADWIN e D3N. O D3N identificou com sucesso as *botnets* e obteve os melhores resultados durante a avaliação de desempenho devido à detecção de mudanças de conceito não lineares na distribuição de dados que o método D3 não detecta. Esse contraste fornece suporte para a classificação aprimorada do método D3N.

5.4.2 Ambiente empírico

Os parâmetros do sistema foram avaliados para determinar a configuração ideal com base nos resultados das métricas dos classificadores. Os parâmetros envolvem o tamanho da

janela, porcentagem de novos dados, limite de variação de conceito e distância de agrupamento. As Figuras 5.8, 5.9, 5.10 e 5.11 apresentam resultados de acurácia, precisão, *recall* e *F1-Score*. Inicialmente, analisamos a influência do tamanho da janela de dados, considerando tamanhos de 100, 250, 500, 750 e 1000, conforme mostrado na Figura 5.8. Com base nos resultados, quanto maior o tamanho da janela, melhor o desempenho do sistema em relação à acurácia, *recall* e *F1-Score*, contudo, a precisão sofreu uma leve redução devido a falsos positivos. O maior tamanho de janela (1000) recebeu mais instâncias de fluxo, oferecendo mais exemplos de dados para o sistema TRUSTED identificar as mudanças de conceito. As Figuras 5.9 e 5.10 mostram resultados semelhantes para a porcentagem de dados e o limiar de detecção de mudança do conceito. Uma vez que a porcentagem de novos dados varia de 10 % a 30 %, o sistema começou a analisar desvios de conceito com uma quantidade considerável de dados, ou seja, 10 % dos novos dados contêm instâncias de fluxo suficientes para identificar mudanças. O limiar de detecção de mudança de conceito é baseado nos resultados da métrica AUC obtidos por meio do classificador discriminativo e foi variado de 60 a 80. Quando o resultado do classificador atingiu um AUC maior que 60, o sistema foi capaz de identificar uma mudança de conceito com o mesmo desempenho ao usar 80 como limiar. Portanto, esses valores testados não mostraram mudanças significativas no desempenho do sistema. Assim, definimos para a melhor configuração 20 % para a nova porcentagem de dados e 70 como limiar de detecção de mudança de conceito.

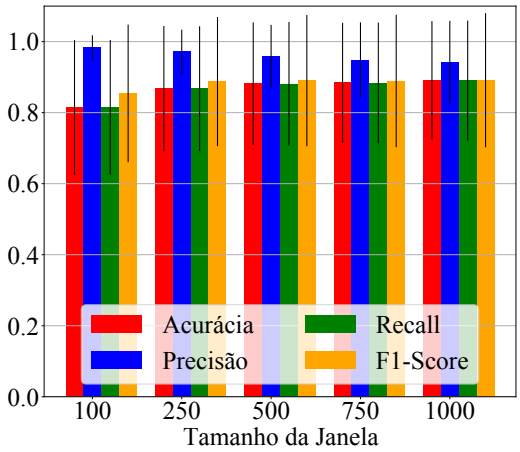


Figura 5.8: Tamanho da Janela

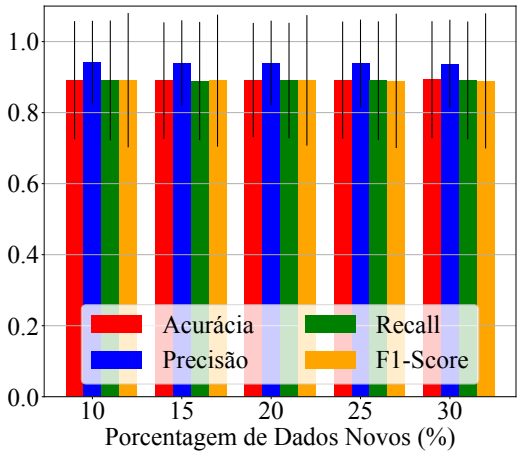


Figura 5.9: Porcentagem de Dados Novos

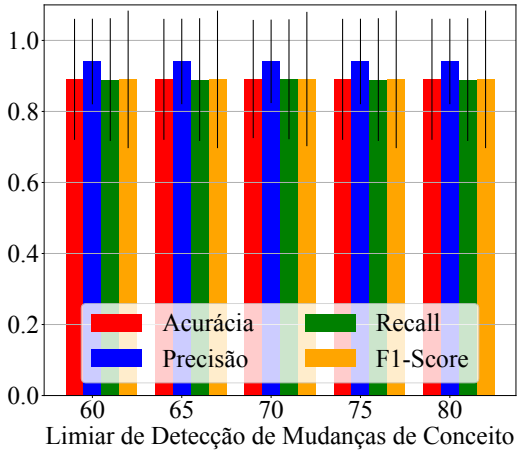


Figura 5.10: Limiar de Detecção de Mudança de Conceito

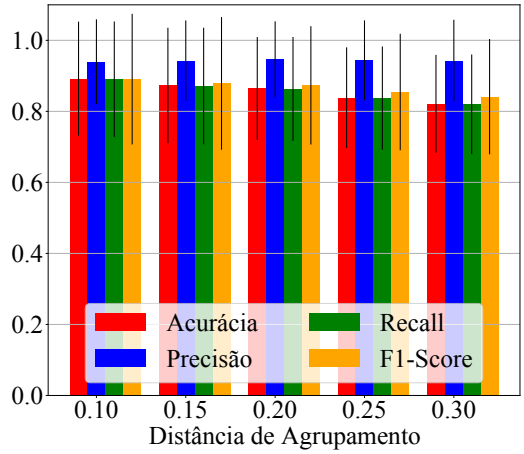


Figura 5.11: Distância de Agrupamento

A Figura 5.11 apresenta os resultados considerando a variação na distância Euclidiana máxima de um *cluster* definida no Algoritmo 2 (ver Capítulo 4). A entrada de dados do algoritmo varia entre 0 e 1 (tais valores representam o conjunto de dados de fluxo). As distâncias foram testadas com valores entre 0,10 e 0,30, aumentando 0,05 por rodada. De acordo com os resultados, quanto maior a distância máxima do *cluster*, menor é o desempenho do sistema. O aumento da distância dentro dos *clusters* gera um alto generalismo, que afeta o desempenho do sistema ao testar valores superiores a 0,1. Além disso, distâncias com valores menores que 0,1 devem causar overfitting no algoritmo de agrupamento. Assim, a partir dos resultados e discussões sobre a configuração empírica do sistema TRUSTED, definimos a melhor configuração com valores de sistema otimizados como tamanho de janela igual a 1000, porcentagem de novos dados de 20 %, limiar de detecção de mudança de conceito de 70 % e distância máxima do *cluster* de 0,10.

5.4.3 Ambiente *offline*

As Figuras 5.12 e 5.13 apresentam os resultados da comparação entre o sistema TRUSTED c/ HAC, o sistema com o algoritmo de aprendizagem supervisionada e detecção de mudanças de conceito *Adaptive Random Forest* (TRUSTED c/ ARF) (Casas et al., 2019) e o sistema com algoritmos não supervisionados com análise de similaridade com janelas fixas e sem detecção de mudanças de conceito (FIXO c/ HAC) (Yu et al., 2010; Yahyazadeh e Abadi, 2015) na base de dados Botnet 2014. A Figura 5.12 apresenta os resultados da acurácia no decorrer da base de dados das 3 comparações. A aplicação do sistema TRUSTED c/ ARF apresentou os melhores resultados com 99% de acurácia, contudo, segundo (Casas et al., 2019) isso se deve ao fato de utilizar a aprendizagem supervisionada que re-treina o modelo de classificação com instâncias rotuladas sempre que identifica uma mudança de conceito na performance do modelo. O uso de aprendizagem supervisionada de forma *online* é custosa e lenta no ambiente de segurança em redes (Al Shorman et al., 2020). O sistema TRUSTED c/ HAC em comparação com o método de janelamento fixo resulta em uma classificação mais estável e com generalidade adequada para o cenário de detecção de *botnets*. Ainda, analisando o desempenho do sistema TRUSTED c/ HAC e o FIXO c/ HAC nas linhas verde e azul pela Figura 5.12 percebe-se que o sistema FIXO apresenta uma alta generalização do tráfego, e dessa forma, classifica incorretamente o tráfego quando ocorrem ataques e é confundido pelo tráfego normal também. Durante a ocorrência da sobrecarga de ataques, a perda de desempenho do sistema TRUSTED é menor, além de manter a acurácia no decorrer da avaliação na base de dados com mais estabilidade sem muitas quedas de desempenho no decorrer da avaliação.

A Figura 5.13 mostra os resultados da comparação de forma geral nas métricas de classificação. O sistema TRUSTED c/ ARF alcançou 99% de desempenho em todas as métricas, sendo os melhores resultados da avaliação. Contudo, esses resultados se devem ao uso da aprendizagem *online* supervisionada em que ocorrem re-treinos a cada classificação. Os resultados dos métodos não supervisionados mostram o TRUSTED c/ HAC com maior desempenho e maior estabilidade no decorrer da base apresentando um desvio padrão menor em comparação com o método FIXO c/ HAC. O desvio padrão é apontado pela barra de erro na Figura 5.13.

Para garantir a solidez da avaliação, foram avaliados 4 cenários do conjunto de dados CTU-13. As Figuras 5.14, 5.15, 5.16 e 5.17 apresentam a métrica de acurácia sobre os cenários avaliados comparando os três sistemas. Em todas as execuções apresentadas o TRUSTED c/HAC proposto nesta dissertação supera a estratégia tradicional FIXO c/ HAC. Embora o TRUSTED c/ ARF (Casas et al., 2019) apresenta os melhores resultados, este pre-treina e re-treina o classificador após cada classificação utilizando o rótulo verdadeiro da instância, e esta estratégia supervisionada é inviável no gerenciamento de segurança de rede de acordo com (Al Shorman

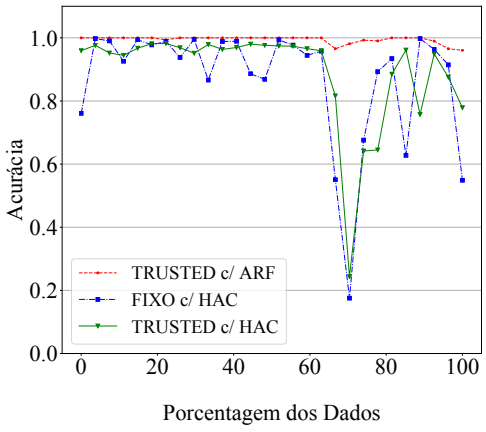


Figura 5.12: Botnet 2014 Acurácias *Offline*

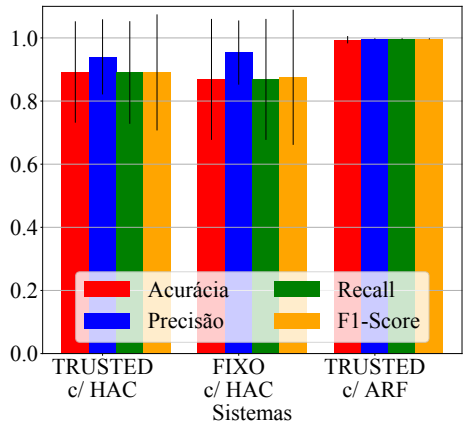


Figura 5.13: Botnet 2014 Comparação Geral *Offline*

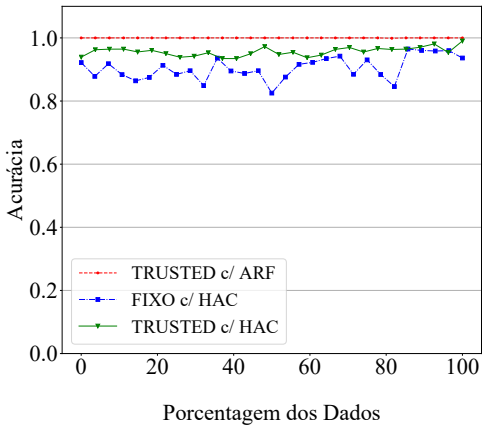


Figura 5.14: CTU-4 Acurácias *Offline*

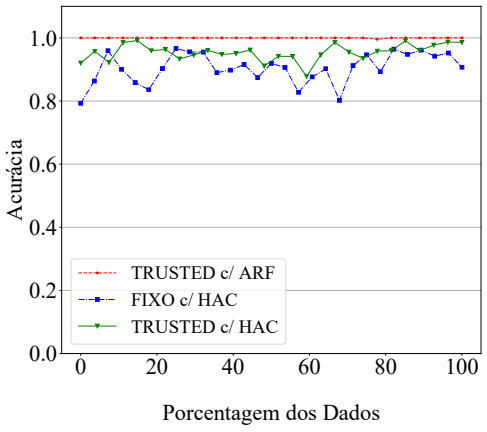


Figura 5.15: CTU-5 Acurácias *Offline*

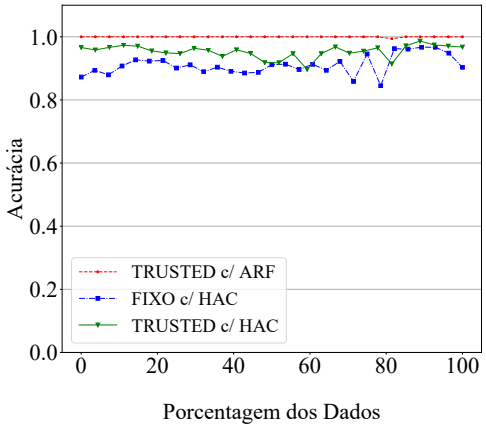


Figura 5.16: CTU-10 Acurácias *Offline*

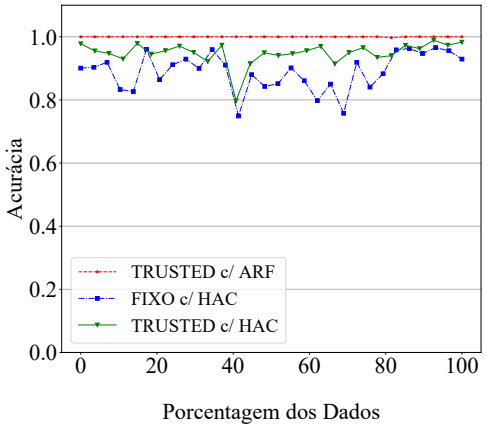


Figura 5.17: CTU-11 Acurácias *Offline*

et al., 2020). Além disso, as Figuras 5.18, 5.19, 5.20 e 5.21 suportam os resultados da acurácia apresentando para todos os cenários as métricas de precisão, *recall* e *F1-Score* que correspondem à diferença apresentada na acurácia entre os sistemas.

Os resultados alcançados pela TRUSTED c/ ARF no ambiente offline consideram o pré-treinamento com amostras offline e rótulos. Além disso, são fornecidos os rótulos corretos para treinar novamente o classificador sempre que ele classificar uma nova instância. Consideramos este o ambiente ideal para abordagens supervisionadas. No entanto, em cenários *online* sem

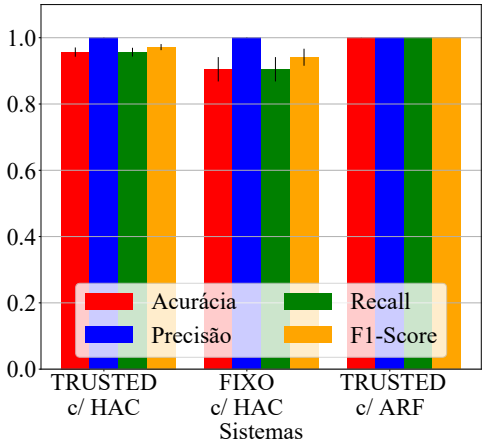


Figura 5.18: CTU-4 Comparação Geral *Offline*

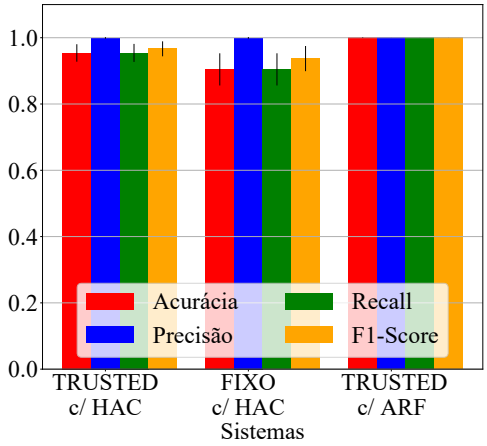


Figura 5.19: CTU-5 Comparação Geral *Offline*

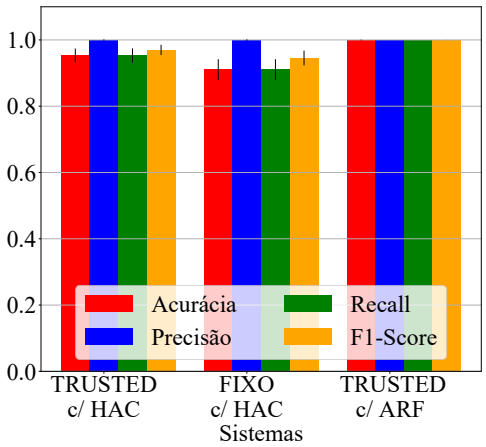


Figura 5.20: CTU-10 Comparação Geral *Offline*

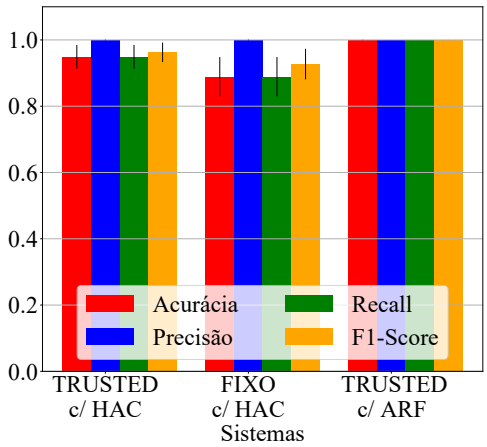


Figura 5.21: CTU-11 Comparação Geral *Offline*

entrega de rótulos, não deve funcionar bem, ver Seção 5.4.4. No ambiente *online* não fornecemos amostras para pré-treino na abordagem supervisionada.

5.4.4 Ambiente *online*

As Figuras 5.22, 5.23, 5.24 e 5.25 apresentam os resultados da acurácia nos 4 cenários da CTU-13 avaliados. Em geral, o TRUSTED c/ HAC apresenta queda em seus resultados em todos os cenários conforme os dados eram processados. Essa ocorrência se deve à falta de instâncias no início da avaliação, onde a janela de dados possui apenas algumas instâncias. Além disso, a diminuição contínua apresenta um ponto controverso contra a configuração empírica, que mostrou que quanto mais instâncias para analisar melhor o resultado.

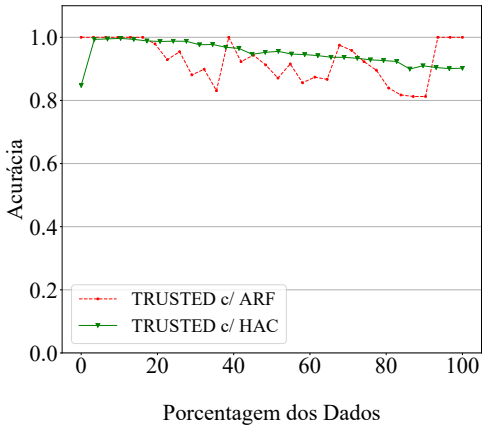


Figura 5.22: CTU-4 Acurácias *Online*

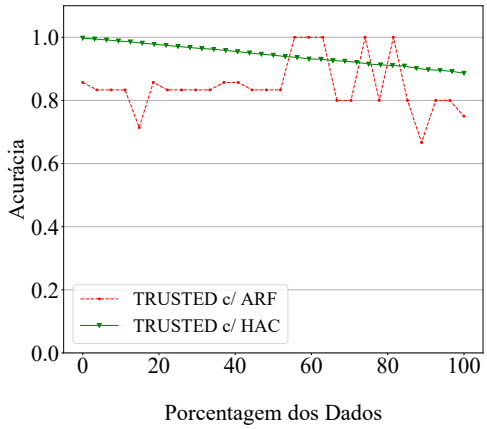


Figura 5.23: CTU-5 Acurácias *Online*

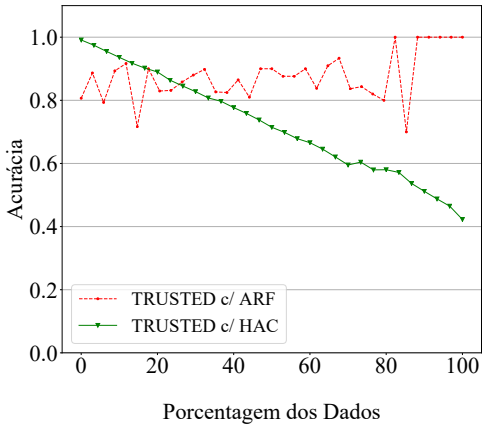


Figura 5.24: CTU-10 Acurácias *Online*

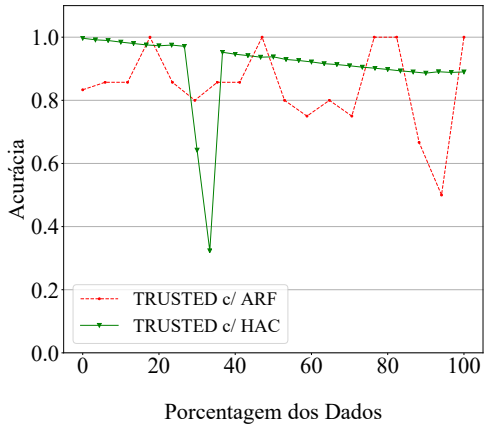


Figura 5.25: CTU-11 Acurácias *Online*

O TRUSTED c/ ARF não recebeu amostras para pré-treinar o classificador, apenas recebeu os rótulos para re-treinar durante a avaliação. Este fato faz com que os resultados da abordagem flutuem, diminuindo e aumentando o desempenho de classificação ao re-treinar o modelo para classificar as instâncias corretamente. Além disso, ele encontrou mais mudanças de conceito, pois monitora os resultados de desempenho, e estes apresentaram quedas abruptas em vários momentos nas avaliações. A cada mudança abrupta no desempenho da classificação, o TRUSTED c/ ARF foi capaz de adaptar a janela de dados e receber os rótulos para re-treinamento. Esses resultados enfatizam que as abordagens supervisionadas devem ter treinamento prévio para alcançar uma classificação de alto desempenho, isto é, mais de 99 % de acurácia como visto na Seção 5.4.3). Portanto, em um cenário não ideal (ex., mundo real), as abordagens supervisionadas são inviáveis.

Embora o TRUSTED c/ HAC seja não supervisionado, ele é estável durante a detecção de botnet em comparação com o TRUSTED c/ ARF, este comportamento é incomum visto que a abordagem supervisionada recebe rótulos para o re-treino do classificador após cada

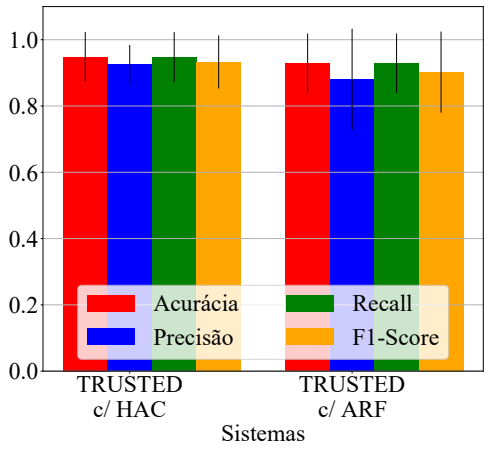


Figura 5.26: CTU-4 Comparação *Online* Geral

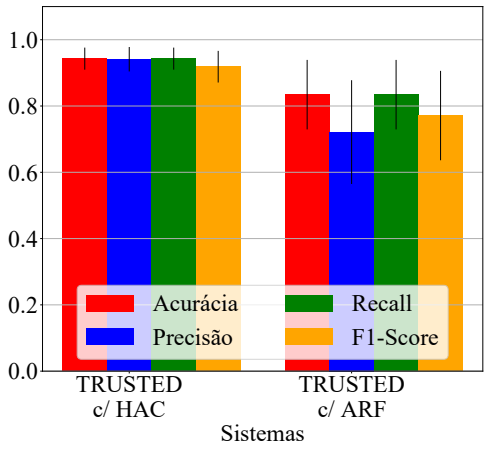


Figura 5.27: CTU-5 Comparação *Online* Geral

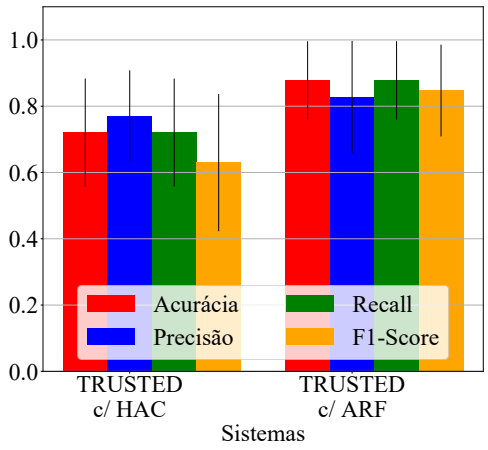


Figura 5.28: CTU-10 Comparação *Online* Geral

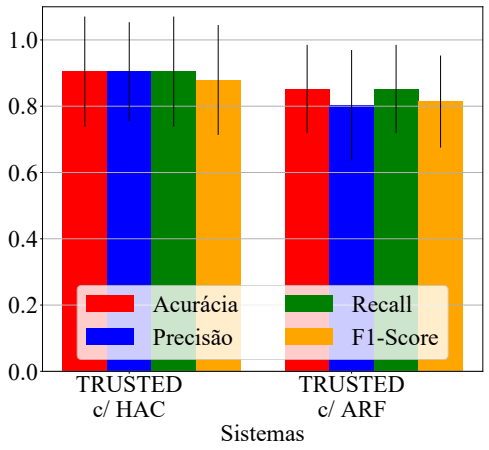


Figura 5.29: CTU-11 Comparação *Online* Geral

classificação. O desempenho do TRUSTED c/ HAC é pior somente na Figura 5.24, onde a acurácia cai constantemente durante a avaliação, isso aconteceu porque o sistema não identificou nenhuma mudança de conceito na distribuição de dados. Consequentemente, o TRUSTED c/ HAC não se adaptou às novas distribuições que chegavam. Além disso, as Figuras 5.26, 5.27, 5.28 e 5.29 apresentam os resultados de todas as métricas avaliadas. O TRUSTED c/ HAC possui resultados inferiores apenas na Figura 5.28 a base CTU-10. No entanto, em todas as outras avaliações online, a abordagem não supervisionada proposta nesta dissertação teve melhor desempenho do que a supervisionada.

Na Figura 5.25, há uma diminuição abrupta nos resultados do sistema TRUSTED c/ HAC perto de 40 % dos dados. Essa perda de desempenho refere-se a um ataque no conjunto de dados. O sistema não conseguiu identificar a diferença entre o tráfego benigno e da botnet naquele momento. Essa confusão do sistema ocorre quando o tráfego benigno tem alta similaridade entre os dispositivos ou quando o tráfego da botnet tem uma diferença significativa em cada bot.

5.5 RESUMO

Este capítulo apresentou a metodologia de avaliação de desempenho direcionada pelas perguntas de pesquisa. Estas compreendem analisar o impacto de mudanças de conceito para a detecção de *botnets* e a eficácia do sistema de detecção não supervisionado de *botnets* identificando mudanças de conceito. A avaliação foi realizada com base nos conjuntos de dados

disponibilizados pelo Instituto de Segurança Cibernética do Canadá e pela Universidade Técnica Checa. Os dados foram descritos com a finalidade de facilitar a compreensão e visualização da distribuição de pacotes e os momentos dos ataques, expondo assim suas principais características. Assim, a avaliação realizada confirma o acréscimo de desempenho do classificador quando utilizado algum método de detecção de mudanças de conceito. Além disso, a avaliação do sistema TRUSTED mostra desempenho superior a estratégias sem detecção de mudanças de conceito no cenário *offline* e supera também a estratégia supervisionada com detecção de mudança de conceito na avaliação considerando o cenário *online*.

6 CONCLUSÃO

Este trabalho tratou o problema de detecção de *botnets*. A detecção de *botnets* se faz necessária diante do dano potencial que pode atingir uma vítima. O principal dano causado por *botnets* é a negação de serviços a usuários legítimos de uma aplicação. Assim, para mitigar este problema, definiu-se um sistema para detecção *online* não supervisionada de *botnets* levando em consideração possíveis mudanças de conceito no *stream* de dados. O sistema utiliza a detecção de mudanças de conceito com o objetivo de adaptar o conjunto de dados a ser classificado pelo processo de agrupamento e análise de similaridade. A detecção da mudança de conceitos ocorre de forma não supervisionada monitorando a distribuição dos dados provenientes do *stream online* e constrói o conjunto de dados para a fase de classificação. Se o método de detecção de mudança de conceito encontrar uma alteração na distribuição dos dados o bloco de dados é reconstruído para adaptar o modelo de aprendizagem, caso contrário o método segue a regra do primeiro que entra é o primeiro que sai em um bloco de tamanho fixo. A detecção de *botnets* ocorre no processo de agrupamento das instâncias, onde é utilizado um algoritmo de agrupamento hierárquico e análise de similaridade dos agrupamentos formados. Foi realizado um levantamento da literatura relacionado ao tema com a finalidade de fundamentar e apresentar os conceitos relacionados a esta dissertação evolvendo a detecção de mudanças de conceito não supervisionada e a detecção de *botnets* não supervisionada. As principais vantagens dos métodos não supervisionados de detecção de acordo com a literatura respaldam-se no fato de não precisar treinar o sistema para realizar a detecção das mudanças de conceito ou a detecção de *botnets*. Para demonstrar a viabilidade da proposta submetemos o sistema a uma avaliação orientada a traços de rede contendo registros de ataques de várias *botnets* disponibilizados pela Universidade de New Brunswick e da Universidade Técnica Tcheca. A avaliação do sistema TRUSTED atingiu valores de acurácia entre 87% a 95% e uma baixa taxa de falsos positivos representados pela métrica de precisão. Dessa forma, o sistema TRUSTED supera a estratégia FIXA em todos os cenários avaliados, além disso, supera em alguns pontos a estratégia supervisionada com detecções de mudanças de conceito na avaliação *online*. Assim, o sistema analisa o comportamento do fluxo de dados da rede de forma adaptativa e sem conhecimento prévio. À vista disso, a proposta apresentada contribui a literatura com avanços na detecção de *botnets* de forma não supervisionada alcançando resultados equivalentes e melhores do que uma abordagem supervisionada considerando um ambiente de processamento *online*.

6.1 TRABALHOS FUTUROS

Como trabalhos futuros esperamos expandir as análises considerando a complexidade de execução do sistema para viabilizar a detecção em tempo real, bem como analisar a detecção antecipada, isto é antes do ataque. A avaliação de detecção antecipada compreende a detecção na fase de construção de uma *botnet* e detecção do comportamento de comunicação entre os *bots*. Além disso, o conjunto de características utilizadas para construir as instâncias de fluxo também possuem influência no desempenho da execução do sistema, dessa forma, há a necessidade de realizar um aprofundamento na fase de pré-processamento com o intuito de aumentar o conjunto de características e selecionar somente as características com maior ganho de informação para serem utilizadas nos processos de detecção. Para melhorar os resultados da fase de classificação pode-se expandir a análise dos classificadores não supervisionados para redes neurais, modelos de detecção de anomalias e redes neurais recorrentes.

REFERÊNCIAS

ABRANET (2019). Dobra o tempo de duração de um ataque ddos em 2018. (https://bit.ly/2HOMbBA). Último acesso Janeiro/2020.

Al Shorman, A., Faris, H. e Aljarah, I. (2020). Unsupervised intelligent system based on one class support vector machine and grey wolf optimization for iot botnet detection. *Journal of Ambient Intelligence and Humanized Computing*, 11(7):2809–2825.

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

Ammar, M., Russello, G. e Crispo, B. (2018). Internet of things: A survey on the security of iot frameworks. *Journal of Information Security and Applications*, 38:8–27.

Baena-Garcıa, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavalda, R. e Morales-Bueno, R. (2006). Early drift detection method. Em *Fourth international workshop on knowledge discovery from data streams*, volume 6, páginas 77–86.

Barthakur, P., Dahal, M. e Ghose, M. K. (2015). Clusibothealer: botnet detection through similarity analysis of clusters. *Journal of Advances in Computer Networks*, 3(1).

Beigi, E. B., Jazi, H. H., Stakhanova, N. e Ghorbani, A. A. (2014). Towards effective feature selection in machine learning-based botnet detection approaches. Em *2014 IEEE Conference on Communications and Network Security*, páginas 247–255. IEEE.

Bifet, A. e Gavalda, R. (2007). Learning from time-changing data with adaptive windowing. Em *Proceedings of the 2007 SIAM international conference on data mining*, páginas 443–448. SIAM.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer Science.

Boutaba, R., Salahuddin, M. A., Limam, N., Ayoubi, S., Shahriar, N., Estrada-Solano, F. e Caicedo, O. M. (2018). A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, 9(1):16.

Carela-Español, V., Barlet-Ros, P., Bifet, A. e Fukuda, K. (2016). A streaming flow-based technique for traffic classification applied to 12+ 1 years of internet traffic. *Telecommunication Systems*, 63(2):191–204.

Casas, P., Mulinka, P. e Vanerio, J. (2019). Should i (re) learn or should i go (on)? stream machine learning for adaptive defense against network attacks. Em *Proceedings of the 6th ACM Workshop on Moving Target Defense*, páginas 79–88.

CERT.br, Centro de Estudos, R. e. T. d. I. d. S. n. B. (2020). Incidentes reportados ao cert.br – janeiro a dezembro de 2019 análise de alguns fatos de interesse observados neste período. (https://cert.br/stats/incidentes/2019-jan-dec/analise.html). Último acesso Março/2020.

Cha, S.-H. (2007). Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1.

Cooke, E., Jahanian, F. e McPherson, D. (2005). The zombie roundup: Understanding, detecting, and disrupting botnets. *SRUTI*, 5:6–6.

Ditzler, G. e Polikar, R. (2011). Hellinger distance based drift detection for nonstationary environments. Em *2011 IEEE symposium on computational intelligence in dynamic and uncertain environments (CIDUE)*, páginas 41–48. IEEE.

Dredze, M., Oates, T. e Piatko, C. (2010). We're not in kansas anymore: detecting domain changes in streams. Em *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, páginas 585–595. Association for Computational Linguistics.

EVEO (2018). nternet das coisas (iot): como o setor de ti será impactado por elas. (`https://www.eveo.com.br/blog/internet-das-coisas/`). Último acesso Janeiro/2020.

Faria, E. R., Gama, J. a. e Carvalho, A. C. P. L. F. (2013). Novelty detection algorithm for data streams multi-class problems. Em *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, página 795–800, New York, NY, USA. Association for Computing Machinery.

Feily, M., Shahrestani, A. e Ramadass, S. (2009). A survey of botnet and botnet detection. Em *2009 Third International Conference on Emerging Security Information, Systems and Technologies*, páginas 268–273. IEEE.

Gama, J., Medas, P., Castillo, G. e Rodrigues, P. (2004). Learning with drift detection. Em *Brazilian symposium on artificial intelligence*, páginas 286–295. Springer.

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.

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

Gentleman, R. e Carey, V. J. (2008). Unsupervised machine learning. Em *Bioconductor case studies*, páginas 137–157. Springer.

Gomes, H. M., Barddal, J. P., Enembreck, F. e Bifet, A. (2017). A survey on ensemble learning for data stream classification. *ACM Comput. Surv.*, 50(2).

Goodman, N. (2017). A survey of advances in botnet technologies. *arXiv preprint arXiv:1702.01132*.

Gözüaçık, Ö., Büyükçakır, A., Bonab, H. e Can, F. (2019). Unsupervised concept drift detection with a discriminative classifier. Em *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, páginas 2365–2368.

Grira, N., Crucianu, M. e Boujemaa, N. (2004). Unsupervised and semi-supervised clustering: a brief survey. *A review of machine learning techniques for processing multimedia content*, 1:9–16.

Hayat, M. Z. e Hashemi, M. R. (2010). A dct based approach for detecting novelty and concept drift in data streams. Em *2010 International Conference of Soft Computing and Pattern Recognition*, páginas 373–378. IEEE.

Hoi, S. C., Sahoo, D., Lu, J. e Zhao, P. (2018). Online learning: A comprehensive survey. *arXiv preprint arXiv:1802.02871*.

Hu, W., Hu, W. e Maybank, S. (2008). Adaboost-based algorithm for network intrusion detection. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(2):577–583.

Jelasity, M., Bilicki, V. et al. (2009). Towards automated detection of peer-to-peer botnets: On the limits of local approaches. *LEET*, 9:3.

Kaspersky (2019). Ddos attacks in 2018 get more sophisticated while declining in quantity. `(https://bit.ly/2Mlp1qJ)`. Último acesso Janeiro/2020.

Khanchi, S., Vahdat, A., Heywood, M. I. e Zincir-Heywood, A. N. (2018). On botnet detection with genetic programming under streaming data label budgets and class imbalance. *Swarm and evolutionary computation*, 39:123–140.

Kim, D. S., Nguyen, H.-N. e Park, J. S. (2005). Genetic algorithm to improve svm based network intrusion detection system. Em *19th International Conference on Advanced Information Networking and Applications (AINA'05) Volume 1 (AINA papers)*, volume 2, páginas 155–158. IEEE.

Kuncheva, L. I. e Faithfull, W. J. (2013). Pca feature extraction for change detection in multidimensional unlabeled data. *IEEE transactions on neural networks and learning systems*, 25(1):69–80.

Lee, J. e Magoules, F. (2012). Detection of concept drift for learning from stream data. Em *2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems*, páginas 241–245. IEEE.

Lindstrom, P., Delany, S. J. e Mac Namee, B. (2010). Handling concept drift in a text data stream constrained by high labelling cost. Em *Twenty-Third International FLAIRS Conference*.

Liu, C.-Y., Peng, C.-H. e Lin, I.-C. (2014). A survey of botnet architecture and batnet detection techniques. *International Journal of Network Security*, 16(2):81–89.

Mahjabin, T., Xiao, Y., Sun, G. e Jiang, W. (2017). A survey of distributed denial-of-service attack, prevention, and mitigation techniques. *International Journal of Distributed Sensor Networks*, 13(12):1550147717741463.

Mahmoud, M., Nir, M., Matrawy, A. et al. (2015). A survey on botnet architectures, detection and defences. *IJ Network Security*, 17(3):264–281.

Masud, M., Gao, J., Khan, L., Han, J. e Thuraisingham, B. M. (2010). Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Transactions on Knowledge and Data Engineering*, 23(6):859–874.

Minku, L. L., White, A. P. e Yao, X. (2010). The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on knowledge and Data Engineering*, 22(5):730–742.

Mirkovic, J. e Reiher, P. (2004). A taxonomy of ddos attack and ddos defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34(2):39–53.

Mohri, M., Rostamizadeh, A. e Talwalkar, A. (2018). *Foundations of Machine Learning*. Massachusetts Institute of Technology.

Moussa, A. A., Nogueira, M. e Guedes, A. L. (2019). Seleção online de features em streaming baseada em alpha-investing para dados de ataques ddos. Em *Anais do XXIV Workshop de Gerência e Operação de Redes e Serviços*, páginas 43–56. SBC.

Muhammad, I. e Yan, Z. (2015). Supervised machine learning approaches: A survey. *ICTACT Journal on Soft Computing*, 5(3).

Narang, P., Hota, C. e Venkatakrishnan, V. (2014). Peershark: flow-clustering and conversation-generation for malicious peer-to-peer traffic identification. *EURASIP Journal on Information security*, 2014(1):15.

Neshenko, N., Bou-Harb, E., Crichigno, J., Kaddoum, G. e Ghani, N. (2019). Demystifying iot security: an exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations. *IEEE Communications Surveys & Tutorials*, 21(3):2702–2733.

Nguyen, T. T. e Armitage, G. (2008). A survey of techniques for internet traffic classification using machine learning. *IEEE communications surveys & tutorials*, 10(4):56–76.

Page, E. S. (1954). Continuous inspection schemes. *Biometrika*, 41(1/2):100–115.

Perdisci, R., Ariu, D., Fogla, P., Giacinto, G. e Lee, W. (2009). Mcpad: A multiple classifier system for accurate payload-based anomaly detection. *Computer networks*, 53(6):864–881.

Qahtan, A. A., Alharbi, B., Wang, S. e Zhang, X. (2015). A pca-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams. Em *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, páginas 935–944.

Ramírez-Gallego, S., Krawczyk, B., García, S., Woźniak, M. e Herrera, F. (2017). A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing*, 239:39–57.

Risse, T., Knežević, P. e Wombacher, A. (2004). P2p evolution: From file-sharing to decentralized workflows (p2p entwicklung: Vom filesharing zu dezentralisierten workflows). *it-Information Technology*, 46(4):193–199.

Rodríguez-Gómez, R. A., Maciá-Fernández, G. e García-Teodoro, P. (2013). Survey and taxonomy of botnet research through life-cycle. *ACM Comput. Surv.*, 45(4).

Ryu, J. W., Kantardzic, M. M., Kim, M.-W. e Ra Khil, A. (2012). An efficient method of building an ensemble of classifiers in streaming data. Em Srinivasa, S. e Bhatnagar, V., editores, *Big Data Analytics*, páginas 122–133, Berlin, Heidelberg. Springer Berlin Heidelberg.

SecurityTrails (2018). Top 10 common network security threats explained. `(https://securitytrails.com/blog/top-10-common-network-security-threats-explained)`. Último acesso Janeiro/2020.

Sethi, T. S. e Kantardzic, M. (2017). On the reliable detection of concept drift from streaming unlabeled data. *Expert Systems with Applications*, 82:77 – 99.

Sethi, T. S., Kantardzic, M. e Hu, H. (2016). A grid density based framework for classifying streaming data in the presence of concept drift. *Journal of Intelligent Information Systems*, 46(1):179–211.

Shalev-Shwartz, S. et al. (2011). Online learning and online convex optimization. *Foundations and trends in Machine Learning*, 4(2):107–194.

Silva, J. A., Faria, E. R., Barros, R. C., Hruschka, E. R., Carvalho, A. C. d. e Gama, J. (2013a). Data stream clustering: A survey. *ACM Computing Surveys (CSUR)*, 46(1):1–31.

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

Souza, D. (2020). Big data: Grande volume de dados + coleta de dados. (https://www.linknacional.com.br/blog/big-data-volume-de-dados/). Último acesso Abril/2020.

Srivastava, S. (2020). Botnet attacks: Severity, protection and most dangerous invasions in past years. https://www.analyticsinsight.net/botnet-attacks-severity-protection-and-most-dangerous-invasions-in-past-years/. Último acesso Maio/2020.

Stallings, W. e Brown, L. (2014). *Segurança de Computadores*. Editora Elsevier Ltda.

Stevanovic, M. e Pedersen, J. M. (2014). An efficient flow-based botnet detection using supervised machine learning. Em *2014 international conference on computing, networking and communications (ICNC)*, páginas 797–801. IEEE.

TIINSIDE (2019). Desatenção na gestão de dados causa prejuízos bilionários a empresas brasileiras. (https://tiinside.com.br/18/10/2019/desatencao-na-gestao-de-dados-causa-prejuizos-bilionarios-a-empresas-brasileiras-aponta-pesquisa/). Último acesso Março/2020.

Vieira, N. (2019). Ataques ddos geram preocupação no setor financeiro. (https://canaltech.com.br/seguranca/ataques-ddos-geram-preocupacao-no-setor-financeiro-156861/). Último acesso Janeiro/2020.

Wainwright, P. e Kettani, H. (2019). An analysis of botnet models. Em *Proceedings of the 2019 3rd International Conference on Compute and Data Analysis*, ICCDA 2019, página 116–121, New York, NY, USA. Association for Computing Machinery.

Wang, P., Sparks, S. e Zou, C. C. (2008). An advanced hybrid peer-to-peer botnet. *IEEE Transactions on Dependable and Secure Computing*, 7(2):113–127.

Wu, W., Alvarez, J., Liu, C. e Sun, H.-M. (2018). Bot detection using unsupervised machine learning. *Microsystem Technologies*, 24(1):209–217.

Wurzinger, P., Bilge, L., Holz, T., Goebel, J., Kruegel, C. e Kirda, E. (2009). Automatically generating models for botnet detection. Em *European symposium on research in computer security*, páginas 232–249. Springer.

Yahyazadeh, M. e Abadi, M. (2012). Botonus: An online unsupervised method for botnet detection. *ISeCure*, 4(1).

Yahyazadeh, M. e Abadi, M. (2015). Botgrab: A negative reputation system for botnet detection. *Computers & Electrical Engineering*, 41:68–85.

Yu, X., Dong, X., Yu, G., Qin, Y. e Yue, D. (2010). Data-adaptive clustering analysis for online botnet detection. Em *2010 Third International Joint Conference on Computational Science and Optimization*, volume 1, páginas 456–460. IEEE.

Zargar, S. T., Joshi, J. e Tipper, D. (2013). A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. *IEEE communications surveys & tutorials*, 15(4):2046–2069.

Zhang, D., Zheng, C., Zhang, H. e Yu, H. (2010). Identification and analysis of skype peer-to-peer traffic. Em *2010 Fifth International Conference on Internet and Web Applications and Services*, páginas 200–206. IEEE.

Zheng, S., Lu, J. J., Ghasemzadeh, N., Hayek, S. S., Quyyumi, A. A. e Wang, F. (2017). Effective information extraction framework for heterogeneous clinical reports using online machine learning and controlled vocabularies. *JMIR medical informatics*, 5(2):e12.