Um Sistema para Detecção Não Supervisionada e Online de Botnets

Bruno Henrique Schwengber, Michele Nogueira

[1]Centro de Ciência de Segurança Computacional (CCSC)
Universidade Federal do Paraná (UFPR)

{bhschwengber,michele}@inf.ufpr.br

Abstract. *The networks of bots (a.k.a., botnets) are a threat to network security due to their dynamic nature, causing damage to companies and users with their support to attacks (e.g., Denial of Services – DoS) and the theft of personal data. Detecting botnets is a challenge, once the infected devices (bots) are widely spread in different geographic places and keep masked in daily-used devices. Techniques presented in the literature usually ignore the fast changes in the statistical data distribution and detects botnet over a static window. These changes are known as concept drifts and they make the classification models obsolete. Hence, this work presents TRUSTED, a system for on-line and unsupervised botnet detection aware of concept drifts. Unlike other works, TRUSTED applies concept drift awareness to optimize the learning for botnet detection in an on-line and unsupervised environment. Results show TRUSTED feasibility detecting botnets using concept drift identification and reched 87% accuracy.*

Resumo. *As redes de bots (i.e., botnets) são uma ameaça à segurança de redes devido à sua natureza dinâmica, causando danos às empresas e usuários com o suporte a ataques (ex., negação de serviços) e roubo de dados pessoais. Detectar as botnets é um desafio, uma vez que os dispositivos infectados (bots) estão amplamente espalhados em diferentes locais geográficos e mantêm-se camuflados em dispositivos de uso rotineiro. As técnicas apresentadas na literatura geralmente ignoram as mudanças na distribuição estatística dos dados e detectam as botnets com base em uma janela estática. Essas mudanças são conhecidas como mudanças de conceito e tornam o modelo de classificação obsoleto. Portanto, este trabalho apresenta o TRUSTED, um sistema para detecção de botnets on-line e não supervisionada com a percepção de mudanças de conceito. Diferente de outros trabalhos, TRUSTED aplica a percepção de mudanças de conceito para otimizar o aprendizado na detecção de botnets em um ambiente on-line e não supervisionado. Os resultados mostram a viabilidade do sistema TRUSTED na detecção de botnets usando identificação de mudanças de conceito e acurácia de 87%.*

1. Introdução

As *botnets* são formadas por dispositivos infectados por programas maliciosos (a.k.a., *malwares*). Esses dispositivos infectados são chamados de *bots* e podem ser controlados remotamente pelo atacante a fim de realizar ataques como negação de serviços (do inglês, *Denial of Service* - DoS), enviar *spam* ou roubar dados pessoais

[Wainwright and Kettani 2019]. Esses ataques excedem os limites de recursos computacionais da vítima e geram prejuízos financeiros. Em média, um ataque DoS custa 1,35 milhão de dólares para a empresa atacada [eSales 2020]. Com a ascensão no uso de dispositivos da Internet das Coisas (do inglês, *Internet of Things* - IoT), que possuem diversas vulnerabilidades (*ex.,* falhas de autenticação) [Ammar et al. 2018], o volume de dados e a velocidade de transmissão aumentaram [Souza 2020], beneficiando os atacantes e requerendo uma rápida adaptação e redefinição dos sistemas de detecção de *botnets*.

O tráfego de dados da Internet segue *streams* de alta velocidade gerando muito tráfego, sendo necessário processar e analisar de forma *on-line* e adaptativa para evitar a propagação do ataque. Contudo, processar, analisar e adaptar é um desafio para a aprendizagem *on-line* devido à alta volatilidade das propriedades estatísticas do tráfego de rede [Carela-Español et al. 2016]. As estratégias tradicionais de detecção de *botnets* (treino e teste) consideram a estaticidade no comportamento do tráfego de rede [Casas et al. 2019]. Porém, os *streams* de alta velocidade evoluem e mudam suas distribuições estatísticas. Essas mudanças são denominadas de mudanças de conceito [Gözüaçık et al. 2019]. Quando uma mudança de conceito ocorre o modelo de classificação se torna obsoleto devido à mudança na distribuição estatística dos dados. Isso afeta de forma negativa o desempenho do modelo de classificação [Gama et al. 2014].

Em geral, os trabalhos existentes relacionados à detecção de *botnets* ignoram a volatilidade e as rápidas mudanças no comportamento do tráfego de redes, por exemplo [Carela-Español et al. 2016, Barthakur et al. 2015, Yu et al. 2010, Yahyazadeh and Abadi 2015]. Porém, a percepção de mudanças de conceito foi aplicada para a detecção de ataques [Casas et al. 2019]. Nesse trabalho, os autores empregaram a percepção de mudanças de conceito em um ambiente de detecção *on-line* e supervisionado para a detecção de tráfego malicioso, ressaltando a melhoria gerada no desempenho pela adaptação do modelo às novas distribuições de dados do tráfego. Contudo, a utilização de classificadores supervisionados é lenta e custosa para a tarefa de detectar tráfego malicioso devido à alta velocidade e ao volume de dados gerados pela Internet [Al Shorman et al. 2020]. Assim, visto que a percepção de mudanças de conceito melhora o desempenho da classificação [Casas et al. 2019], é imprescindível um sistema de detecção *on-line* não supervisionada com a percepção de mudanças de conceito para identificar as mudanças nas distribuições estatísticas dos dados e detectar *botnets*.

Este artigo apresenta o sistema TRUSTED (do inglês, *an on-line sys**T**em fo**R** **U**nsupervis**e**d bo**T**n**E**t **D**etection*). O sistema TRUSTED aplica a identificação de mudanças de conceito não-supervisionada sobre uma janela deslizante para prover uma entrada otimizada para a etapa de detecção *on-line* não-supervisionada de *botnets*. Assim, utilizando a identificação de mudança de conceitos, clusterização e a análise de similaridades, 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 dos modelos de classificação para a detecção das *botnets*.

A avaliação do sistema TRUSTED segue uma abordagem orientada a traços, utilizando a base de dados Botnet 2014 [Beigi et al. 2014]. A base de dados inclui tráfego benigno e de *botnets* seguindo um modelo *Peer-to-Peer* (P2P) e os protocolos *Internet*

Relay Chat (IRC) e *HyperText Transfer Protocol* (HTTP). Dessa forma, são extraídas as características dos fluxos de tráfego de rede constituindo as instâncias para análise. Neste trabalho, consideramos como instância um conjunto de características de um fluxo. Assim, as instâncias são submetidas à transformação em um *stream* de dados para simular uma detecção *on-line*. A partir do *stream*, é realizada a construção da janela de dados, a identificação de mudanças de conceito e a detecção de dispositivos infectados. Os resultados demonstram a viabilidade do sistema TRUSTED por alcançar 87% de acurácia para detectar *botnets* usando identificação de mudanças de conceito para prover um conjunto de dados otimizado para aprendizagem. Em comparação com a estratégia tradicional de janelamento fixo o sistema TRUSTED supera nas métricas consideradas.

Este artigo está organizado como segue. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 detalha o sistema TRUSTED. A Seção 4 descreve a metodologia da avaliação de desempenho e discute os resultados. Por fim, a Seção 5 apresenta as conclusões e os trabalhos futuros.

2. Trabalhos Relacionados

O problema de detectar *botnets* é constantemente tratado por trabalhos na literatura, tais como [Yu et al. 2010, Yahyazadeh and 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 *on-line* ou *off-line*. Os trabalhos supervisionados empregam a estratégia tradicional para modelos 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 *on-line*. 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 *on-line* 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 and Abadi 2015].

Os trabalhos de detecção de *botnets* também são observados seguindo a perspectiva de critérios como a detecção sem ou com conhecimento prévio, a detecção em estágios iniciais ou estágios avançados, e a independência de criptografia [Yahyazadeh and Abadi 2015]. Adicionamos a esses critérios a percepção ou não de mudanças de conceito. Na comparação entre os trabalhos relacionados, a percepção de mudanças de conceito é apresentada apenas por [Casas et al. 2019], que alcançou resultados de 99% de acurácia ressaltando a relevância e melhoria de desempenho de classificadores quando utilizadas técnicas de percepção de mudanças de conceito, porém, [Casas et al. 2019] emprega somente algoritmos supervisionados. Estes são considerados custosos e lentos para a tarefa detecção de tráfego malicioso [Al Shorman et al. 2020]. As técnicas de mudanças de conceito atualizam os modelos de classificação obsoletos devido às mudanças de conceito na distribuição estatística dos dados [Gözüaçık et al. 2019]. Assim, os trabalhos de detecção *on-line* não

supervisionada de *botnets* apresentam uma carência na percepção de mudanças de conceito [Yu et al. 2010, Yahyazadeh and Abadi 2015].

O estudo de [Yu et al. 2010] apresentou uma técnica de detecção *on-line* de *botnets* centralizadas. Os autores realizam a transformação do tráfego em um *stream* de instâncias com as características de rede, as instâncias representam as características de um fluxo. Para processamento, é mantida uma janela de instâncias, deletando instâncias antigas e substituindo por novas. A janela serve como entrada para o algoritmo *K-Means* de agrupamento. A partir dos grupos formados pelo algoritmo, a técnica proposta analisa a similaridade entre as instâncias de cada grupo para detectar as *botnets*. Quando um grupo ultrapassa um certo limiar de similaridade, ou seja, quando ele possuir instâncias similares, o grupo é rotulado como suspeito. Entretanto, esta técnica não considera mudanças de conceito relacionadas à distribuição dos dados, o que pode influenciar de forma negativa no desempenho da técnica proposta [Casas et al. 2019].

O trabalho de [Yahyazadeh and Abadi 2015] apresentou um sistema de detecção *on-line* de *botnets* apoiado por reputação negativa e análise de similaridade. O sistema rastreia o histórico de atividades coordenadas entre os dispositivos na rede e calcula a reputação para cada dispositivo com base na participação deles em atividades da rede entre dispositivos (ex., envio de arquivos e mensagens). Esse sistema utiliza o agrupamento incremental e a análise de similaridade para rotular o tráfego. O sistema utiliza janelas de tamanhos fixos baseados em tempo e não considera a ocorrência de mudanças de conceitos na janela pré-definida. Isto influencia de maneira negativa no resultado do sistema [Casas et al. 2019], pois as mudanças na distribuição estatística dos dados tornam o modelo atual de detecção obsoleto [Gözüaçık et al. 2019].

Este trabalho apresenta o sistema TRUSTED, um sistema para a detecção *on-line* não supervisionada de *botnets* com a percepção de mudanças de conceito. O objetivo do sistema é detectar tráfego de *botnets* de forma *on-line* para agilizar a tomada de decisão. Diferente de outros trabalhos, o sistema TRUSTED emprega a identificação de mudanças de conceito no tráfego para manter o modelo de classificação atualizado.

3. Sistema TRUSTED

Esta seção descreve o sistema de Detecção Não Supervisionada e *On-line* de *Botnets*, denominado TRUSTED (do inglês, *an on-line sysTem foR Unsupervised boTnEt Detection*). Seu objetivo consiste em detectar o tráfego de *botnets* de forma *on-line* e sem supervisão. O sistema TRUSTED detecta o tráfego malicioso em seus estágios iniciais e durante ataques. As seguintes subseções descrevem o uso da detecção de mudanças de conceito na aprendizagem *on-line* e detalha o funcionamento do sistema proposto.

3.1. Mudanças de Conceito e a Aprendizagem *On-line*

Em geral, os *streams* de dados reais evoluem e sofrem mudanças em suas distribuições estatísticas. Um *stream* refere-se ao envio e/ou recebimento de dados pela Internet de forma constante e contínua. Essas mudanças causam problemas de desempenho nos modelos de aprendizagem, que adotam uma abordagem estática entre os dados de entrada e a saída do modelo. As mudanças nas distribuições dos dados referem-se ao fenômeno de mudança de conceito em que o comportamento e características dos dados mudam ao longo do tempo [Gama et al. 2014].

Na aprendizagem *on-line*, existem dois tipos de mudanças de conceito, a real e a virtual. Um modelo de classificação recebe uma entrada X e a classifica com um rótulo y levando em consideração um treinamento prévio. A mudança de conceito real refere-se à mudança no desempenho de classificação do modelo e ocorre quando a distribuição dos dados $p(X)$ afeta a classificação $p(y, X)$ do modelo de aprendizagem. Por outro lado, a mudança de conceito virtual refere-se às mudanças somente na distribuição dos dados $p(X)$ sem considerar as classificações $p(y, X)$ [Gama et al. 2014]. Contudo, a identificação de mudanças de conceito real e virtual utilizam métodos de identificação supervisionados e não supervisionados, respectivamente [Sethi and Kantardzic 2017]. Dessa forma, para o correto funcionamento da classificação e detecção de mudanças de conceito é necessária a utilização de um classificador incremental em que é possível realizar a adaptação do modelo de classificação de forma sequencial [Gözüaçık et al. 2019].

A estratégia tradicional do uso de identificação de mudanças de conceito segue *(i)* classificação, *(ii)* diagnóstico e *(iii)* atualização [Gama et al. 2014]. Porém, essa estratégia considera apenas a aprendizagem supervisionada em que o método de identificação de mudanças de conceito depende da supervisão com rótulos para identificar as mudanças de classificação do modelo de aprendizagem. Neste trabalho consideramos a seguinte estratégia para realizar a aprendizagem *on-line (i)* diagnóstico, *(ii)* atualização e *(iii)* classificação. Na fase de diagnóstico, são comparados os dados novos com os dados antigos a fim de identificar as mudanças de conceito. A fase de atualização organiza o conjunto de dados usados para realizar a classificação. Inicialmente, quando não existem dados antigos para a identificação de mudanças de conceito, somente é realizada a fase de classificação. A fase de classificação categoriza os novos dados de acordo com o novo modelo de classificação.

3.2. Detalhamento do Sistema TRUSTED

O sistema TRUSTED congrega as etapas de pré-processamento, identificação de mudanças de conceito e detecção de *botnet*, ilustradas na Figura 1. Como entrada de dados para o sistema, assume-se a captura do tráfego de rede em uma interface de rede em modo promíscuo no *hardware* em que o sistema estiver sendo executado (ex., *firewall*). A primeira etapa compreende o pré-processamento dos dados em que é feita a medição, transformação em fluxos e extração das características. A segunda etapa realiza a construção da janela de dados e identifica possíveis mudanças de conceito no comportamento dos dados. Na terceira etapa é realizada a detecção de comportamento malicioso por meio do agrupamento do *stream* de dados e análise de similaridade.

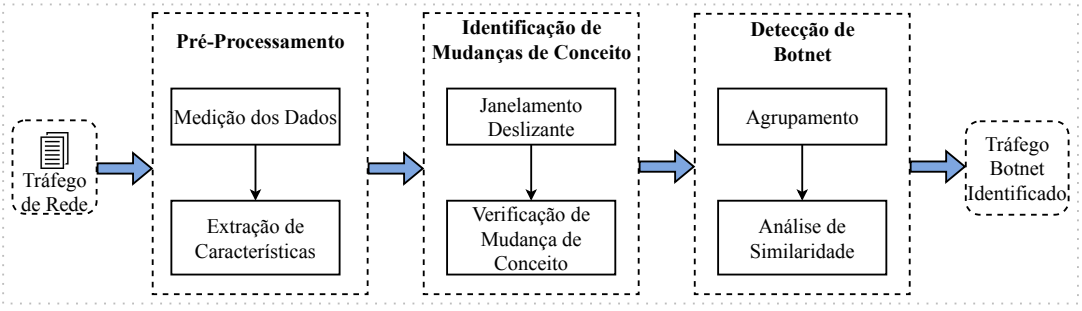


Figura 1. Arquitetura do Sistema TRUSTED

3.2.1. Pré-Processamento

A etapa de pré-processamento é responsável por coletar o tráfego da rede, separar o tráfego de rede por fluxos e extrair as características dos fluxos para formar as instâncias de fluxo. Essa etapa é necessária para obter as características do tráfego de rede e organizar os dados para a etapa de identificação de mudanças de conceito no comportamento dos dados, além de servir como entrada para a etapa de detecção de *botnet*.

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 um índice que indica a ordem sequencial de captura dos pacotes. A partir do conjunto de pacotes de F, o sistema TRUSTED extrai as características estatísticas do fluxo, definidas como uma instância por $X^F = \{x_1, x_2, x_3, ..., x_i\}$, neste, x é uma característica extraída do fluxo e i é um índice associado às características extraídas. Essas instâncias de dados compõem a janela de dados $W = \{X_1^F, X_2^F, X_3^F, ..., X_z^F\}$ contendo as instâncias de fluxos e é mantida de forma deslizante. Uma janela deslizante possui tamanho fixo e é mantida seguindo a regra do primeiro que entra é o primeiro que sai, isto é, quando um novo conjunto de instâncias é recebido o mesmo número de instâncias é retirado da janela.

3.2.2. Identificação de Mudanças de Conceito

Nesta etapa do sistema TRUSTED, é recebido como entrada o conjunto de instâncias de fluxos provenientes da etapa de pré-processamento. O tratamento do conjunto de instâncias é feito em dois passos: *(i)* janelamento deslizante e *(ii)* identificação de mudança de conceito. A janela de dados resultante do janelamento deslizante é constituído pelo conjunto de instâncias de fluxos de rede e possui um tamanho fixo predeterminado. A janela de dados será utilizada para a identificação de mudanças de conceito e para a etapa de *clustering* que irá agrupar as instâncias a fim de encontrar *botnets*.

Para a construção da janela de dados W na fase do janelamento deslizante, o módulo recebe como entrada um *stream* de dados do tipo X^F. Para a identificação de mudanças de conceito o método de detecção de mudança de conceito discriminativa de [Gözüaçık et al. 2019] é adaptado neste trabalho para detectar mudanças de conceito em dados não linearmente separáveis e é apresentado no Algoritmo 1. A adaptação se da pela alteração do classificador discriminativo, originalmente um algoritmo de regressão logística, modificado para uma árvore de decisão. Este método se destaca por ser não supervisionado e ser classificado entre os detectores de distribuições multivariadas, o que é recomendado para *streams* multidimensionais (*ex.,* tráfego de rede).

Dessa forma, o Algoritmo 1 descreve o funcionamento do método. O método de identificação de mudanças de conceito recebe como parâmetros de entrada o *stream* X^F, a quantidade máxima de instâncias (δ), um percentual de novas instâncias (σ) e um limiar (β) para a identificação 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, na linha 4, verifica-se 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 $\delta(1+\sigma)$ instâncias, o método aplica rótulos às instâncias. Assim, nas 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. Uma mudança de conceito é encontrada quando o resultado do classificador conseguir distinguir entre as instâncias com desempenho superior a β como mostra a linha 11, pois segundo [Gözüaçık et al. 2019], se é possível classificar o conjunto com alta eficiência, os dados são separáveis no mapa de características, e portanto existe uma mudança de conceito na distribuição dos dados. Assim, se ocorreu a mudança de conceito no comportamento da rede é sinalizado uma mudança de conceito e eliminadas δ instâncias do final do bloco W, apresentados nas linhas 12 e 13. Caso contrario, não é sinalizada a mudança de conceito e são eliminadas $\delta * \sigma$ instâncias do final do bloco W, apresentados nas linhas 15 e 16. Na Figura 2, é possível identificar esse comportamento da eliminação dos dados devido a ocorrência da mudança de conceito.

Algorithm 1 D3 Adaptado: Discriminative Drift Detector Adaptado

Entrada: *stream* $\{X_n^F\}_{n=1}^N$, δ, σ, β
Resultado: Verdadeiro ou Falso

1: Inicializa a janela W onde $|W| = \delta(1+\sigma)$
2: Inicia Classificador discriminativo C {Árvore de Decisão}
3: **for all** (X, y) em stream **do**
4: **if** W não estiver cheio **then**
5: $W \leftarrow W \cup X$ {adiciona X no início de W}
6: **else**
7: S é um vetor de s, $|W| = |S|$ {s é uma variável temporária de rótulos}
8: s = 0 for W[1,δ] {rótulo velho (0) e novo (1)}
9: s = 1 for W[$\delta + 1$, final]
10: Treine C(W, S) {treine C com os rótulos S de W}
11: **if** AUC(C,S $\geq \beta$) **then**
12: drift = verdadeiro
13: Elimine δ instâncias do final de W
14: **else**
15: drift = Falso
16: Elimine $\delta * \sigma$ elementos do final de W
17: **end if**
18: **end if**
19: **end for**

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. 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 identificação da mudança de conceito. Quando não ocorre uma mudança de conceito, somente é eliminado um conjunto de $\delta\sigma$ instâncias do fim do bloco conforme mostra a Figura 2.

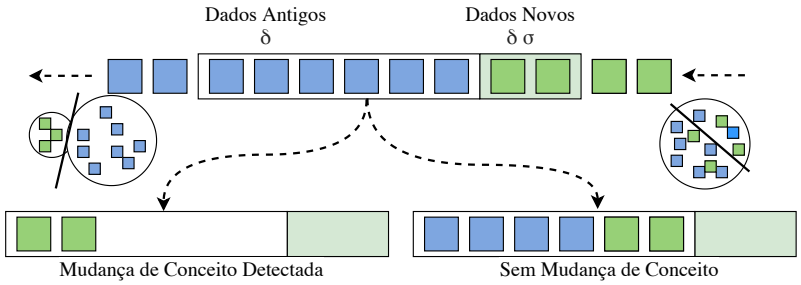


Figura 2. Janelamento Deslizante e Identificação de Mudanças de Conceito. Adaptado de [Gözüaçık et al. 2019]

3.2.3. Detecção de *Botnet*

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

Para o agrupamento, o sistema TRUSTED recebe como base o bloco de dados construído pela fase de janelamento deslizante e a identificação de mudanças de conceito. Assim, 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. No Algoritmo 2, linha 1, é inicializado o grupo de agrupamentos vazio. Na linha 3, cada instância é definida como um agrupamento e, na linha 5, os agrupamentos são adicionados na árvore hierárquica T. Enquanto o número de agrupamentos for maior que 1, na linha 7, são definidos os pares mais próximos entre todas os agrupamentos dentro da distância máxima γ. Nas linhas 8 e 9, são removidos os agrupamentos antigos e adicionados os novos agrupamentos criados, respectivamente. Na linha 10, é adicionado o conjunto de agrupamentos A na árvore hierárquica T.

Algorithm 2 HAC: Hierarchical Agglomerative Clustering

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

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

A partir dos agrupamentos formados pelo Algoritmo 2, calcula-se a distância entre todas as instâncias de um mesmo agrupamento seguindo a Equação 1. Na equação, S re-

presenta o agrupamento sendo avaliado e $d(x, y)$ é a distância entre as duas instâncias (x e y) em um mesmo grupo S. Para identificar os agrupamentos de *bots*, realiza-se a comparação do valor $\Delta(S)$ resultante da Equação 1 com um limiar χ. Se o valor $\Delta(S)$ é menor que o limiar χ, o agrupamento é rotulado como infectado, pois segundo [Yu et al. 2010, Yahyazadeh and Abadi 2015] um conjunto com alta similaridade, ou seja, $\Delta(S) < \chi$, é considerado infectado.

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

4. Avaliação de Desempenho

A avaliação de desempenho do sistema TRUSTED passou pela seleção do conjunto de dados, a análise do desempenho do sistema e da comparação com trabalhos da literatura. O sistema é projetado para funcionar *online*, mas para fins de avaliação do desempenho a abordagem seguida é orientada a traços, para melhor controle do cenário e para o uso de dados contendo registros de ataques reais. A captura dos dados utilizados foram realizadas por ferramentas de redes como TCPdump e tshark. Estas geram arquivos do tipo pcap. O conjunto de dados utilizado é oferecido pela Universidade de New Brunswick. Além disso, a execução dos *scripts* foi realizada em um sistema operacional Ubuntu 18.04 LTS com um processador Intel Core i7 8750H e 16GB de memória RAM.

A base de dados utilizada é a Botnet 2014[1] construída com o objetivo de apresentar um cenário com generalidade, realismo e representatividade. A generalidade é alcançada utilizando 16 variações de *botnets* e 3 arquiteturas diferentes apresentadas na Tabela 1, bem como a porcentagem de dados gerados de cada variante de *botnet*. O realismo é provido pela mescla do uso de tráfego real coletado em ambiente não controlado e tráfego gerado em ambiente controlado. A representatividade foi superada pela criação do conjunto combinado das coletas realizadas com dados não sobrepostos de tráfego real e tráfego artificial. Ainda, o conjunto de dados foi construído utilizando traços de três bases diferentes. Os dados foram mesclados utilizando a metodologia de sobreposição [Beigi et al. 2014].

Esse conjunto de dados foi usado como entrada para a avaliação de desempenho. A extração dos fluxos é realizada utilizando *scripts* Python para ler o arquivo pcap e identificar todos os fluxos contidos na base. Os fluxos F são agrupados em instâncias pela tupla ⟨IP de origem, IP de destino, Porta de origem, Porta de destino e protocolo de comunicação⟩. A partir das instâncias de fluxos F são extraídas as características estatísticas de cada fluxo utilizando *scripts* Python com base nos pacotes identificados dentro de cada fluxo. As características extraídas são a quantidade de pacotes enviados, a quantidade de pacotes recebidos, a quantidade de *bytes* enviados, a quantidade de *bytes* recebidos, o horário do início do envio de pacotes do fluxo, o horário do fim do envio de pacotes do fluxo, o horário do início do recebimento de pacotes do fluxo, o horário do fim do recebimento de pacotes do fluxo, o intervalo entre o horário de início e fim do envio de pacotes e o intervalo entre o horário de início e fim do recebimento de pacotes. Um conjunto de 49489 instâncias com 10 características cada foi obtido. Esse conjunto se apresenta de forma desbalanceada sendo 44732 instâncias benignas e 4757 instâncias de ataques. A Figura 3 apresenta a distribuição de pacotes e os momentos das ocorrências

[1]Botnet 2014, https://www.unb.ca/cic/datasets/botnet.html, Último acesso março de 2020.

Nome da Botnet	Tipo	% 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 1. Distribuição das Botnets na base de dados

dos ataques de *botnets* da base de dados. Ainda, são diferenciadas as quantidades de pacotes enviados e recebidos nos fluxos extraídos representados pelas linhas verde e azul. As linhas verticais vermelhas representam o momento dos ataques no decorrer da base de dados. A partir de 70% da base de dados ocorre a maior sobrecarga dos ataques de *botnet*.

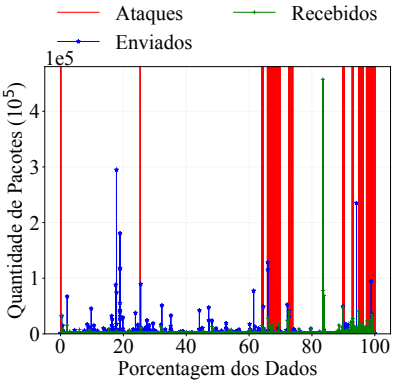


Figura 3. Distribuição dos pacotes e ataques na base de dados

As fases seguintes compreendem a avaliação do sistema de detecção de *botnets*. Para isso, aplicou-se sobre o conjunto de dados uma transformação para *data stream*, a fim de simular um ambiente *on-line*. A partir disso, os sistemas de detecção de *botnets* serviram para posterior comparação. Para realizar o agrupamento das instâncias provenientes da fase de janelamento e identificação de mudanças de conceito. Nas implementações, empregaram-se as bibliotecas *scikit-learn*[2] v0.23, *scikit-multiflow*[3] v0.5.3 e *pandas*[4] v1.0.5 implementadas em Python.

A análise do sistema proposto seguiu duas etapas *(i)* avaliação de variáveis do sistema TRUSTED e *(ii)* comparação das instâncias do sistema TRUSTED utilizando

[2]scikit-learn, https://scikit-learn.org/stable/, Último acesso em Março de 2020.

[3]scikit-multiflow, https://scikit-multiflow.github.io/, Último acesso em Março de 2020.

[4]pandas, https://pandas.pydata.org/, Último acesso em Março de 2020.

estratégias do estado da arte. Inicialmente, o conjunto de características dos fluxos é transformado em um *stream* de dados para simular um ambiente *on-line*. O *stream* serve de entrada para todas as avaliações realizadas. As janelas de dados são construídas pela fase de janelamento e identificação de mudanças de conceito e analisadas de forma incremental de acordo com a instância comparada. Para a comparação, são consideradas as abordagens propostas em [Casas et al. 2019] que apresenta um método de aprendizagem *on-line* supervisionada com percepção de mudanças de conceito, e [Yu et al. 2010, Yahyazadeh and Abadi 2015] que apresentam métodos de detecção *on-line* não supervisionada de *botnets* usando similaridade intra-*cluster*, porém somente consideram um janelamento fixo sem considerar mudanças de conceito.

Para avaliar o desempenho nas comparações, foram utilizadas diferentes métricas de desempenho de classificação. Essas métricas têm como objetivo quantificar o desempenho dos modelos de classificação. Elas são a acurácia, a precisão, o *recall* e o *F1-score*. Para obter essas métricas são necessários os valores de verdadeiro positivo (VP), verdadeiro negativo (VN), falso positivo (FP) e falso negativo (FN). Dessa forma, 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) consiste da porcentagem de verdadeiros positivos dentre todos os exemplos cuja classe esperada é a positiva $(VP/(VP + FN))$. 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. As métricas de precisão e *recall* são ponderadas entre os resultados de cada classe para balancear os resultados, visto que a base de dados utilizada é desbalanceada.

Resultados

Esta subseção apresenta os resultados da avaliação do sistema proposto. O sistema busca identificar mudanças de conceito para adaptar o conjunto de dados de entrada para a etapa da detecção de *botnets*. Os resultados são referentes às duas etapas de *(i)* avaliação de variáveis do sistema TRUSTED e *(ii)* comparação de instâncias do sistema TRUSTED utilizando estratégias do estado da arte.

A Figura 4 apresenta os resultados das métricas de acurácia, precisão, *recall* e *F1-Score* de cada variação das variáveis de tamanho da janela, porcentagem de dados novos recebidos na janela, limiar de detecção de mudanças de conceito e distância máxima dos agrupamentos. Para a variação do tamanho da janela foram fixados o valores de 10% da porcentagem de dados novos, 70 para o limiar da detecção de mudanças de conceito e 0,10 para a distância de agrupamento. Para a variação da porcentagem de dados novos o tamanho da janela foi de 1000, 70 para o limiar de detecção de mudanças de conceito e 0,10 para a distância de agrupamento. Na variação do limiar de detecção de mudanças de conceito o tamanho da janela foi de 1000, a porcentagem de dados novos de 20% e 0,10 para a distância de agrupamento. Para a variação da distância de agrupamento o tamanho da janela foi de 1000, a porcentagem de dados novos de 20% e o limiar de detecção de mudanças de conceito de 70. Inicialmente, foi avaliada a influência do tamanho da janela de dados. A Figura 4(a) mostra que quanto maior o tamanho da janela, melhor o desempenho do sistema em relação às métricas de acurácia, *recall* e *F1-Scores*, a precisão sofre uma leve diminuição, esta redução da precisão refere-se ao

aumento de falsos positivos compensado pelo aumento das métricas restantes referente aos verdadeiros positivos, verdadeiros negativos e falsos negativos. As Figuras 4(b) e 4(c) mostram valores similares entre as variações de porcentagem de dados novos e do limiar da identificação de mudanças de conceitos. Os valores testados não mostraram grande influência das variáveis no desempenho do sistema mantendo o mesmo desempenho em todas os valores testados.

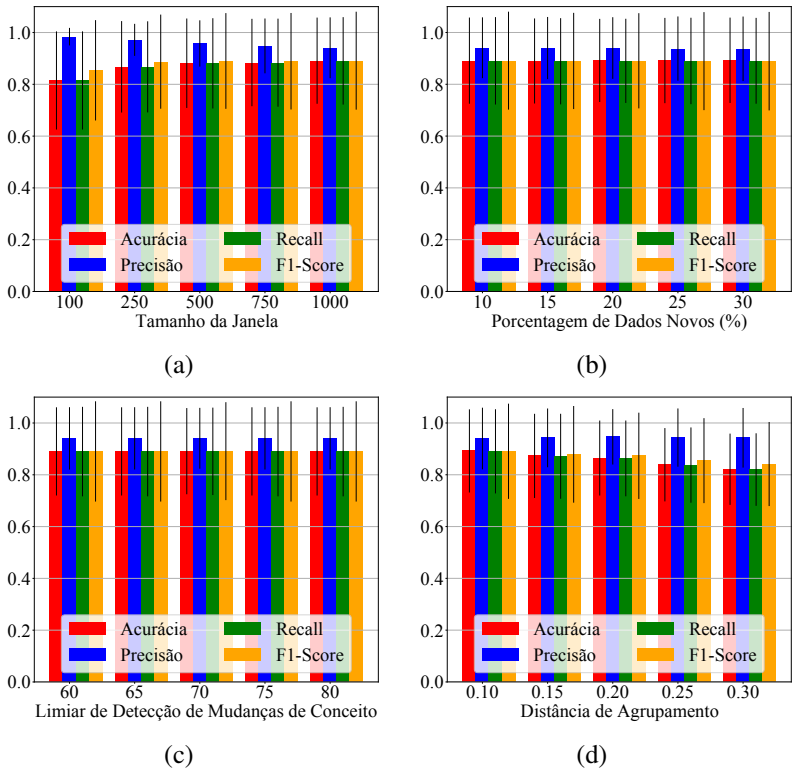


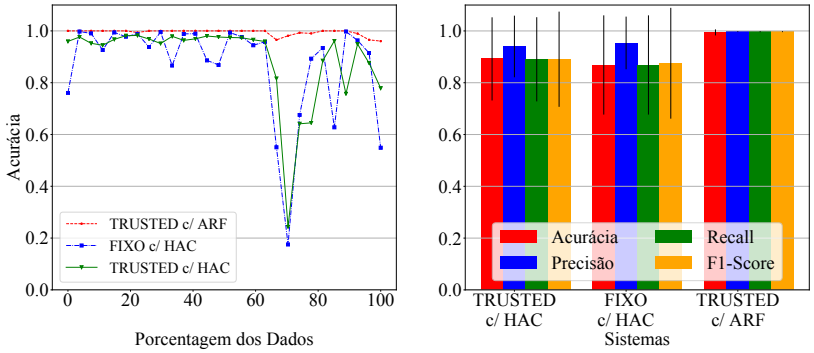
Figura 4. Avaliação das Variáveis do Sistema TRUSTED

A Figura 4(d) apresenta os resultados considerando a variação na distância máxima de um agrupamento formado pelo algoritmo 2. Os dados de entrada do algoritmo possuem valores entre 0 e 1. Para avaliar essa variação, foram executados testes com distâncias entre 0,10 e 0,30, aumentando 0,05 por execução. Os valores mostram que quanto maior a distância máxima de um agrupamento, menor o desempenho do sistema. Assim, a partir da avaliação das variáveis do sistema foram definidos os valores otimizados para a configuração do sistema como tamanho de janela igual a 1000, porcentagem de dados novos de 20%, limiar de identificação de mudança de conceito de 70% e a distância máxima de um agrupamento de 0,10.

A Figura 5 apresenta 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 and Abadi 2015]. A Figura 5(a) apresenta os resultados da acurácia no decorrer da base de dados das 3 comparações. A aplicação do sistema TRUSTED c/ ARF apre-

sentou 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 *on-line* é custosa e lenta no ambiente de segurança em redes [Al Shorman et al. 2020]. O sistema TRUSTED 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* apresentada na Figura 5. Ainda, analisando o desempenho do sistema TRUSTED e o FIXO nas linhas verde e azul pela Figura 5(a) percebe-se que o sistema FIXO apresenta uma alta generalização do tráfego, e dessa forma, classifica incorretamente o tráfego quando ocorrem ataques. 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(b) 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 *on-line* supervisionada em que ocorrem re-treinos constantes. Os resultados dos métodos não supervisionados mostram o TRUSTED c/ HAC com mais de desempenho e maior estabilidade no decorrer da base apresentando o um desvio padrão menor no decorrer da avaliação em comparação com o método FIXO c/ HAC.



(a) (b)

Figura 5. Comparação com o estado da arte

5. Conclusão

Este artigo apresentou o sistema TRUSTED, que detecta *botnets* de forma não supervisionada, *on-line* considerando as mudanças de conceito no tráfego de rede. O sistema constrói uma janela de dados de forma adaptativa identificando possíveis mudanças de conceito no tráfego de rede para prover um conjunto de dados otimizado para a fase de detecção de *botnets* que agrupa as instâncias de fluxos e analisa a similaridade. A avaliação do sistema seguiu uma abordagem orientada a traços e empregou uma base de dados contendo vários tipos de ataques e tráfego normal. Os resultados demonstram que o sistema possui 87% de acurácia e baixa taxa de falsos positivos. Além disso, o uso da identificação de mudanças de conceito aumentou o desempenho do sistema em comparação com trabalhos que usam a estratégia de janelamento fixo.

Referências

Al Shorman, A., Faris, H., and 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.

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

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

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

Carela-Español, V., Barlet-Ros, P., Bifet, A., and 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., and Vanerio, J. (2019). Should i (re) learn or should i go (on)? stream machine learning for adaptive defense against network attacks. In *Proceedings of the 6th ACM Workshop on Moving Target Defense*, pages 79–88.

eSales (2020). Cibersegurança: os prejuízos dos ataques cibernéticos para empresas. (https://esales.com.br/blog/ciberseguranca/). Último acesso Julho/2020.

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

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

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

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.

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

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

Yahyazadeh, M. and 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., and Yue, D. (2010). Data-adaptive clustering analysis for online botnet detection. In *2010 Third International Joint Conference on Computational Science and Optimization*, volume 1, pages 456–460. IEEE.