Uma Comparação entre os Sistemas de Detecção de Ameaças Distribuídas de Rede Baseado no Processamento de Dados em Fluxo e em Lotes

Fábio C. Schuartz[1], **Mauro Fonseca**[1], **Anelise Munaretto**[1]

[1]CPGEI – Universidade Tecnológica Federal do Paraná (UTFPR)
Av. Sete de Setembro, 3165 – 80230-901 – Curitiba – PR – Brazil

phabyo@gmail.com.br, {maurofonseca,anelise}@utfpr.edu.br

Abstract. *With a greater massification of devices connected to the Internet of Things, there is a growing number of vulnerability exploitations detected every year. Thus, faster and more accurate systems are needed to efficiently detect distributed attacks. This paper proposes a system for online detection of distributed network threats using data stream processing. The results obtained by the proposed system are compared with the results obtained by a system using batch processing. The proposed system is evaluated through two metrics: accuracy and number of false-positive and false-negative. The results show that using data stream processing improved detection accuracy by up to 17,50%, reducing the number of false-positives and false-negatives by up to 66,61%.*

Resumo. *Com uma maior massificação de dispositivos conectados à Internet das Coisas, surge um número crescente de ataques de exploração de vulnerabilidades detectados a cada ano. Assim, são necessários sistemas cada vez mais rápidos e precisos para detectar com eficiência ataques distribuídos. Este artigo tem como proposta um sistema de detecção online de ameaças de rede utilizando processamento por fluxos. Os resultados obtidos pelo sistema proposto são comparados com os resultados obtidos por um sistema utilizando processamento por lotes. O sistema proposto é avaliado através de duas métricas: acurácia e número de falsos-positivos e falsos-negativos. Os resultados mostram que utilizar o processamento por fluxo de dados melhorou a acurácia de detecção em até 17,50%, reduzindo o número de falsos-positivos e falsos-negativos em até 66,61%.*

1. Introdução

Com a expansão das redes de computadores mundialmente, as pessoas estão cada vez mais conectadas, utilizando dispositivos que compartilham informações através da Internet. Este crescente aumento de dispositivos em funcionamento acarreta em uma gama de vulnerabilidades que podem ser exploradas por um agente malicioso. A Internet das Coisas (*Internet of Things* - IoT) proporciona a conectividade entre milhões de dispositivos globalmente e isso permite que uma vulnerabilidade possa afetar esses aparelhos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

simultaneamente. Um ataque de varredura de portas pode descobrir uma vulnerabilidade que, futuramente, permite um ataque de negação de serviço (*Denial of Service* - DoS) em larga escala. De acordo com o relatório de ameaças divulgado pela empresa Symantec [Symantec 2018], a Internet das Coisas continua a crescer como alvo primário para os cibercriminosos explorarem. O número de ataques à IoT cresceu de aproximadamente 6.000 em 2016 para 50.000 em 2017, um aumento de 600% em apenas um ano, segundo o relatório. Atualmente, bilhões de dispositivos que utilizam a IoT estão em uso, e a companhia de pesquisa e análises Gartner [Gartner 2016] estima que mais de 50% dos novos processos e sistemas de negócios incluirão um componente ligado à IoT em 2020. Ainda, a empresa F5 Labs [Labs 2017] divulgou seu relatório global sobre ameaças à IoT, o qual examinou como atacantes desenvolveram *botnets* para visar dispositivos conectados especificamente à IoT, os quais cresceram 280% em relação ao relatório semestral anterior.

Com isso em mente, é importante termos mecanismos de segurança que possam detectar essas formas de ataque de uma maneira rápida e com acurácia para termos uma Internet das Coisas segura. Entretanto, as técnicas comuns de detecção para os ataques de varredura e negação de serviço não são eficientes e rápidas o suficiente para trabalhar com um grande volume de dados de tráfego. Assim, tem se tornado mais comum o uso de algoritmos de aprendizagem de máquina para detectar tais intrusões, pois a alta capacidade de processamento distribuído por máquinas utilizando o processamento de fluxo permite construir algoritmos complexos e eficientes que podem tratar os dados de maneira *online*.

Este artigo propõe uma comparação entre os métodos de classificação de dados através de um sistema de aprendizado e classificação em lotes em relação a um sistema de aprendizado e classificação por fluxos, utilizando o sistema de detecção de intrusão para classificar tráfego distribuído de rede, proposto por Schuartz et al. [Schuartz et al. 2017]. A principal contribuição deste trabalho é demonstrar que o uso de processamento de dados em fluxo apresenta melhores resultados comparados ao processamento de dados em lotes, comumente utilizado nos diversos trabalhos na literatura. O sistema proposto é avaliado quanto a acurácia e número de falsos-positivos e falsos-negativos sobre um banco de dados de tráfego de rede simulada, amplamente testada pela comunidade científica. Os resultados obtidos mostram que a utilização do processamento em fluxos traz ganhos consideráveis para a detecção de ataques distribuídos de varredura de portas e de negação de serviço, sem deteriorar a detecção *online*.

O restante deste artigo é dividido da seguinte maneira. Na Seção 2 está a discussão dos trabalhos relacionados. A Seção 3 apresenta o sistema proposto de detecção de ataques. A Seção 4 apresenta o conjunto de dados de tráfego rotulados utilizado na classificação. A Seção 5 apresenta os métodos de classificação e de seleção de características que foram utilizados para a avaliação. A Seção 6 apresenta os resultados obtidos e discute os mesmos. Por último, a Seção 7 conclui o artigo.

2. Trabalhos Relacionados

A literatura apresenta diversas técnicas para detecção de ameaças utilizando aprendizado de máquina. Essas abordagens de aprendizado de máquina podem ser classificadas em quatro tipos de aprendizados: supervisionado, não-supervisionado, semi-

supervisionado e por reforço. O aprendizado supervisionado ocorre quando o conjunto de dados estiver rotulado, ou seja, cada amostra contém a saída (classificação) correta que ela deve gerar, enquanto no aprendizado não-supervisionado as amostras não possuem esse rótulo. O semi-supervisionado contém amostras com e sem rotulação, e no aprendizado por reforço os agentes tentam encontrar a melhor maneira de atingir uma meta, recebendo uma recompensa quando realizam uma ação que os deixem mais perto do objetivo [Pecht and Kang 2019]. Na área de redes, utiliza-se o aprendizado supervisionado para realizar a classificação de ataques, enquanto o não-supervisionado é utilizado para detectar padrões e anomalias. Na classificação de ataques, os métodos mais utilizados na literatura são as árvores de decisão, o K-Vizinho Mais Próximo (*K-Nearest Neighbor* - KNN), as redes Bayesianas e as Máquinas de Vetores de Suporte (*Support Vector Machine* - SVM) [Chen et al. 2015].

Corrêa et al. apresentam um estudo sobre a utilização da técnica de mineração de dados conhecida como Árvore Adaptativa de Hoeffding (*Hoeffding Adaptive Tree* - HAT) para criar um modelo de previsão com objetivo de detectar intrusões em uma rede [Corrêa et al. 2017]. Os resultados mostram uma acurácia no modelo de previsão do HAT em torno de 95%, com tempo de processamento de cada amostra na base de dados sendo aproximadamente 10^{-5}s. Desale et al. discutem técnicas de classificação e melhoria de desempenho em sistemas de detecção de intrusão utilizando diferentes classificadores [Desale et al. 2015]. Os resultados gerados pela análise dos classificadores mostram que Naive Bayes e Árvore de Hoeffding apresentam os melhores resultados. Em termos de acurácia, Naive Bayes apresenta melhor resultado, porém leva mais tempo para classificar que a Árvore de Hoeffding.

No trabalho apresentado por Lobato et al., é proposto um sistema de detecção de ameaças em tempo real por processamento de dados utilizando a arquitetura Lambda [Lobato et al. 2016]. Os resultados mostram uma acurácia com valores acima de 95% e taxa de falsos-positivos abaixo de 6%, analisando até 3,57 milhões de amostras por minutos. Schuartz et al. propõe uma plataforma distribuída para detecção de ameaças em tempo real por análise de fluxos [Schuartz et al. 2017]. O sistema proposto trabalha com diferentes algoritmos de aprendizado de máquina em cima de uma plataforma distribuída, permitindo diversos classificadores operarem simultaneamente. Os resultados obtidos mostram uma acurácia média acima de 90% e tempo de detecção aproximado de $95\mu s$ por fluxo.

Embora os trabalhos apresentados anteriormente apresentem uma técnica de detecção em tempo real, os classificadores são inicialmente preparados através de uma base de dados de treinamento e depois utilizados para classificar um fluxo de dados, de maneira estática. Este artigo propõe um sistema distribuído de detecção de ameaças de rede utilizando algoritmos de processamento em fluxo, onde o modelo aprendido está constantemente sendo atualizado para refletir as novas amostras que chegam pelo fluxo de dados. Os dados que chegam são classificados e incorporados à base de dados, atualizando os modelos de classificação e permitindo os classificadores trabalharem sobre as mudanças de relacionamento entre dados de entrada e saída com o decorrer do tempo (*concept drift*).

3. O Sistema de Classificação Proposto

O sistema proposto para detecção de ameaças de rede em tempo real é dividido em cinco módulos que englobam os sensores para a coleta de dados em diversas fontes, a normalização das informações coletadas, o transporte dos dados normalizados para o sistema de processamento em fluxo, a análise do fluxo de dados em tempo real e a visualização dos resultados obtidos e de alertas que sejam gerados. Os sensores são responsáveis pela coleta dos dados e os demais módulos estão contidos em uma nuvem dedicada para processamento distribuído de dados, conforme proposto em [Schuartz et al. 2017].

O módulo de coleta de dados é composto por ferramentas de coleta de tráfego executadas em sensores distribuídos na rede. Os sensores podem ser máquinas virtuais em ambientes virtualizados ou máquinas físicas com espelhamento de tráfego de um *link* da rede. Esses dados são coletados através da captura dos pacotes Ethernet que entram pelas interfaces do sistema e de arquivos de *log* do sistema operacional e outras aplicações.

O módulo de normalização das informações obtêm características de fluxo através da extração de dados dos cabeçalhos dos pacotes, utilizando janelas de tempo bem definidas. Os fluxos são definidos por uma sequência de pacotes que possuem um mesmo IP de origem e um mesmo IP de destino, com o objetivo de detectar ameaças de rede, como varredura de portas e ataques de negação de serviço. A partir dos cabeçalhos dos pacotes TCP/IP de cada fluxo, obtêm-se 84 características, tais como a quantidade de portas de origem e destino, e número de pacotes TCP, UDP e ICMP.

Para transportar os dados de um ponto ao outro é utilizado um agente de mensagens, isto é, uma ferramenta para a publicação e assinatura (*publish and subscribe*) de uma fila de mensagens. Cada fonte produz dados normalizados e os publica em uma fila. Cada unidade de processamento em fluxos escolhe quais filas assinar, podendo ele escolher uma ou mais filas.

O módulo de processamento em fluxos classifica as amostras recebidas, uma por uma, através de diferentes algoritmos de aprendizado de máquina. Antes da classificação, um filtro de características é aplicado às amostras recebidas com o objetivo de reduzir a complexidade do processamento. O filtro é definido através de algoritmos de redução de dimensionalidade e seleção de características pré-estabelecidos. A definição do modelo de classificação é realizada de forma *online*, através do processamento da amostra recebida e da atualização de sua estrutura de dados contendo a nova amostra, sem exceder o limite de memória e de maneira mais rápida possível.

O módulo de visualização recebe as informações obtidas na camada anterior e fornece uma interface entre o sistema e o usuário. O módulo mostra a análise da rede em tempo real e gera alertas de tráfego suspeito, caso necessário.

3.1. Mineração de Fluxo de Dados

Um fluxo de dados pode ser definido como uma sequência contínua e mutável de dados que constantemente chega em um sistema para ser processado ou armazenado. Os dados são massivos (por exemplo, em volumes de *terabytes*), temporalmente ordenados, variando rapidamente e potencialmente infinito. Essas características causam desafios no campo de fluxo de dados [Kholghi et al. 2010]. Os métodos tradicionais de mineração de

dados normalmente requerem múltiplos exames dos dados e por isso são inviáveis para as aplicações de fluxo de dados.

Devido ao grande volume de dados gerado pelas redes de comunicação e tráfego de Internet, e ao alto custo de armazenamento, é impossível armazenar completamente todos os fluxos de dados ou examiná-los múltiplas vezes. Assim, é necessário utilizar técnicas para extrair as informações embutidas nos dados. A mineração do fluxo de dados (*Data Stream Mining* - DSM) é caracterizada por permitir o processamento de dados em tempo real através do conceito de processamento incremental, onde o novo dado é analisado assim que é recebido.

A DSM refere-se à extração da estrutura de conhecimento que é representada como modelos e padrões dos fluxos infinitos de informações. Devido a limitação dos fluxos de dados, os métodos propostos na literatura são baseados em estatísticas, cálculo e teorias da complexidade. Os dados de entrada (origem da informação) pode ocorrer de diversas fontes. Uma porção dos dados de entrada é selecionada utilizando técnicas tais como amostragem ou agregação. Estes dados são então entregues a um algoritmo que é responsável em atualizar o modelo de predição. Por último, o resultado da classificação é conhecido para cada amostra processada.

O fluxo de entrada $a_1, a_2, ...$ chega sequencialmente, item por item, e descreve um sinal subjacente A, uma função unidimensional

$$A : [1...N] \rightarrow R^2 \tag{1}$$

onde a_i são incrementos de $A[j]$. Considere

$$a_i = (j, I_i), I_i \geq 0 \tag{2}$$

e

$$A_i[j] = A_{i-1}[j] + I_i \tag{3}$$

onde A_i é o estado do sinal após observar o i-ésimo item no fluxo. Múltiplos a_i podem incrementar um dado $A[j]$ com o tempo. O objetivo é que o fluxo de dados A seja processado e para cada amostra a_i existe uma classe predita associada com ela. Isso pode ser representado pela tupla (a_i, b_i), onde a classe predita é representada por b_i. Assim, a predição será correta caso $a_i = b_i$ e incorreta caso $a_i \neq b_i$.

Este modelo de fluxo de dados é o mais popular e utilizado para monitoramento de endereços IP que acessam um servidor *web*, endereços IP de origem que enviam pacotes em um *link*, etc., pois o mesmo endereço pode acessar o servidor *web* múltiplas vezes ou enviar múltiplos pacotes pelo *link* com o tempo [Kholghi et al. 2010].

3.2. Características Utilizadas

No agrupamento de fluxos de rede, os pacotes de rede são abstraídos em fluxos na camada de rede, onde um fluxo é caracterizado como a sequência de pacotes que vai de um endereço IP de origem para o mesmo endereço IP de destino, durante uma janela

fixa de tempo. Para este tipo de agrupamento, as características originais inferidas são variáveis numéricas que estão relacionadas ao conjunto completo de todos os pacotes transmitidos entre dois endereços de IP.

Inicialmente são extraídas um total de 84 características do tráfego de rede a partir de um arquivo *pcap*, através do extrator de características baseado em fluxos *CICFlowMeter* [CICFlowMeter 2017], [Lashkari et al. 2017], incluindo IP de origem, porta de origem, IP de destino, porta de destino e protocolo. Todas as características são definidas e explicadas no site da ferramenta *CICFlowMeter* [CICFlowMeter 2017].

A seguir, são encontrados os conjuntos contendo as melhores características para detectar cada tipo de ataque, dentre as 84 características extraídas inicialmente. Em seu trabalho, Sharafaldin et al. apresenta os conjuntos contendo as melhores características para cada tipo de ataque [Sharafaldin et al. 2018]. Os ataques detectados neste artigo são três: *Botnet*, *PortScan* e *DDoS*. As vantagens da redução de dimensionalidade podem ser encontradas no trabalho de Kezih e Taibi [Kezih and Taibi 2013]. A Tabela 1 mostra a lista das melhores características selecionadas para detecção de cada ataque.

Tabela 1. Melhores características para detecção dos três tipos de ataque (*Botnet*, *PortScan* e *DDos*), segundo [Sharafaldin et al. 2018].

Tipo de Ataque	Características
Botnet	Subflow F. Bytes Total Len F. Packets F. Packet Len Mean B. Packets/s
PortScan	Init Win F. Bytes B.Packets/s PSH Flag Count
DDoS	B. Packet Len Std Avg Packet Size Flow Duration Flow IAT Std

Conforme mostrado na Tabela 1, para detectar um ataque do tipo *Botnet*, as melhores características são encaminhando de *bytes* de sub-fluxo (*Subflow Forwarding Bytes*), encaminhamento do comprimento e média dos pacotes (*Total Length Forwarding Packets* e *Forwarding Packet Length Mean*) e pacotes de retorno (*Backward Packets per Second*). Para ataques do tipo *PortScan*, utilizam-se os *bytes* da janela de encaminhamento inicial (*Initial Windows Forwarding Bytes*), pacotes de retorno (*Backward Packets per Second*) e *push flags* (*PSH Flag Count*). Por último, os ataques de *DDoS* são melhores detectados utilizando as características de comprimento do pacote de retorno (*Backward Packets Length Standard*), tamanho médio dos pacotes (*Average Packet Size*), e dois intervalos de tempo de chegada (*Flow Duration* e *Flow IAT Standard*).

4. Conjunto de Dados Utilizado

Com o objetivo de avaliar o desempenho da abordagem proposta para detecção de anomalias, foram realizados experimentos utilizando um conjunto de dados disponibilizado publicamente na Internet. Dos conjunto de dados existentes, a maioria está

desatualizada e não são mais confiáveis. Alguns dos conjuntos sofrem de uma falta de diversidade e volume no tráfego, outros não cobrem a variedade dos ataques conhecidos enquanto outros possuem conjunto de características e *metadata* incompletos [Sharafaldin et al. 2018].

O conjunto de dados CICIDS2017 [Sharafaldin et al. 2017] contém tráfego legítimo e os ataques atuais mais comuns encontrados na literatura. O conjunto de dados utilizado refere-se ao dia sete de julho de 2017, onde ataques do tipo *Botnet*, *PortScan* e *DDos* ocorreram em horários específicos pré-determinados. O CICIDS2017 consiste de fluxos de rede rotulados, incluindo o *payload* completo dos pacotes, no formato *pcap*, os perfis correspondentes com os fluxos rotulados, e arquivos *CSV* para fins de aprendizado de máquina e aprendizado profundo.

5. Algoritmos de Classificação

Nesta seção são apresentados alguns algoritmos de mineração de dados em fluxos mais utilizados na classificação de tráfego com aprendizado de máquina [Buczak and Guven 2016]. Para classificar as ameaças, três classificadores de características distintas são utilizados: árvore de decisão (*Decision Tree*), árvore adaptativa Hoeffding (*Hoeffding Adaptive Tree*) e bayesiano simples (*Naive Bayes*). A árvore adaptativa Hoeffding é um algoritmo baseado em árvore de decisão, enquanto o bayesiano simples é um algoritmo baseado em probabilidade. Esses classificadores visam estimar a eficiência de classificadores simples e de baixo processamento, e são capazes de realizar o treinamento do modelo de classificação com grande rapidez, permitindo a rápida detecção de ameaças. Os algoritmos foram escolhidos devido sua larga utilização e por seus comportamentos bem definidos na detecção de intrusão [Buczak and Guven 2016].

5.1. Árvore de Decisão - (Decision Tree - DT)

A árvore de decisão é um dos principais métodos de mineração de dados. Uma árvore de decisão padrão consiste de uma raiz, um número de galhos, um número de nós e um número de folhas. Um galho é uma cadeia de nós, da raiz à folha, e cada nó apresenta uma característica do sistema [Sun 2010]. Cada nó armazena os valores de probabilidade de cada classe e são utilizados como parâmetros na tomada de decisões. Quando uma amostra desconhecida entra no modelo, o algoritmo percorre os nós, iniciando pela raiz, avaliando os atributos e calculando a probabilidade daquela amostra pertencer a uma classe específica. Assim, economiza-se tempo na classificação, pois normalmente não é necessário percorrer todas as características da amostra para ocorrer a classificação da mesma. A árvore é gerada inicialmente pela caracterização do nó raiz, contendo a probabilidade de cada classe na amostragem. A seguir, o nó é dividido, gerando filhos que representam novas características da amostragem e que estão associados a um conjunto de probabilidades da classe relativo a essa característica. O processo é reiterado para todos os nós até ser atingido a probabilidade de 100% para alguma classe, sendo este um nó folha. O caminho para cada folha explica o motivo e resultados para cada amostra. Baseado na estrutura da árvore, certas regras podem ser obtidas e a correlação entre amostras é encontrada para futuras classificações.

O algoritmo C4.5 é amplamente usado para classificar um conjunto de dados desconhecidos, onde os valores das características são testados na árvore de decisões. O procedimento do algoritmo C4.5 é detalhado a seguir [Sun 2010].

Passo 1: Calcular a entropia da informação para identificar a classe no conjunto de treinamento S.

$$Info(S) = -\sum_{i=1}^{k}\{[freq(C_i, S/[S])]log_2[freq(C_i, S/[S])]\} \tag{4}$$

onde $[S]$ é o número de amostras no conjunto de treinamento. C_i é uma classe, $i = 1, 2, ..., k$. k é o número de amostras e $freq(C_i, S)$ é o número de amostras inclusas em C_i.

Passo 2: Calcular o valor da informação esperada, $Info_x(S)$, do teste X à partição S.

$$Info_x(S) = -\sum_{i=1}^{k}[(S/[S])Info(S_i)] \tag{5}$$

onde i é o número de saídas para teste X e S_i é o subconjunto de S correspondente a i-ésima saída.

Passo 3: Calcular o ganho de informação após a partição de acordo com teste X.

$$Gain(X) = Info(S) - Info_x(S) \tag{6}$$

Passo 4: Calcular o valor da partição $SplitInfo(X)$ para a partição S em i subconjntos.

$$SplitInfo(X) = \sum_{i=1}^{k}[(S/[S])log_2(S/[S])] \tag{7}$$

Passo 5: Calcular a taxa de ganho de informação de $SplitInfo(X)$ em $Gain(X)$.

$$GainRation(X) = Gain(X)/SplitInfo(X) \tag{8}$$

O $GainRation(X)$ compensa pela quantidade de informações providas pelo conjunto de treinamento. Assim, o atributo com maior $GainRation(X)$ é colocado como raiz da árvore de decisão.

Passo 6: O modelo da árvore de decisão pode ser construído repetindo do **passo 1** ao **passo 5** para cada galho.

5.2. Árvore Adaptativa Hoeffding - (Hoeffding Adaptive Tree - HAT)

O algoritmo Hoeffding é um algoritmo de aprendizado por árvore de decisão, também conhecido como *Very Fast Decision Tree* (VFDT), e que apresenta características importantes para processamento de dados em fluxos. A árvore de decisão é gerada incrementalmente e o conteúdo das amostras coletadas são conhecidas dinamicamente durante o processamento. O VFDT possui estruturas compactas, não necessitando armazenar os

dados processados, apenas informações estatísticas relevantes. Ainda, cada amostra é processada apenas uma vez.

O VFDT utiliza uma métrica estatística chamada limite Hoeffding. Este limite apresenta informações sobre quando o grupo é grande o suficiente para representar uma população que não é inteiramente conhecida ainda [Hoeffding 1963]. O limite Hoeffding possui um fator de erro estimado que pode ser parametrizado, definido por:

$$\varepsilon = \sqrt{\frac{R^2 log(1/\delta)}{2n}} \tag{9}$$

onde R é uma variável aleatória pertencente ao conjunto de valores reais, n é o número de amostras processadas, δ é um valor configurável que descreve a margem aceitável do erro de cálculo.

Entretanto, o algoritmo VFDT não possui a habilidade de detectar e adaptar ao *concept drift* existente no processamento de fluxo de dados. Gama et al. mostra que uma sequência de dados processados ao decorrer do tempo tem seu padrão de comportamento alterado [Gama et al. 2014]. Bifet and Gavaldà [Bifet and Gavaldà 2009] adaptaram o algoritmo VFDT para detectar o *concept drift* e ajustar o modelo de previsão. Esta técnica é conhecida como árvore adaptativa de Hoeffding (*Hoeffding Adaptive Tree* - HAT).

O HAT utiliza uma janela para definir qual dado é relevante aos nós de decisão, definida através de estimações estatísticas coletadas durante o processamento das amostras. Com o tempo, as amostras mais antigas são descartadas pelo algoritmo e as novas amostras atualizam o modelo de conhecimento, adaptando, assim, o modelo para classificar o novo padrão de dados.

5.3. Bayesiano Simples - (Naive Bayes - NB)

O algoritmo bayesiano simples (*Naive Bayes* - NB) é um classificador probabilístico, onde cada característica do sistema não tem influência no valor das demais características, simplificando assim a predição de classificação e tornando-se robusto ao ruído nos dados de entrada. O método calcula inicialmente as probabilidades para cada característica (ou para um conjunto de características) corresponder a uma determinada classe, através dos dados de treinamento. Quando uma nova amostra desconhecida chega ao sistema, o algoritmo calcula a probabilidade de pertencer a cada uma das classes, para cada característica. A probabilidade da amostra pertencer a cada uma das classes é resultado do produto de todas as probabilidades de cada característica. A classe que obtiver a maior probabilidade estimada é escolhida para a classificação da amostra.

6. Avaliação e Resultados

Para avaliar a acurácia da classificação, um protótipo do sistema proposto foi implementado em um sistema composto por uma máquina com processador Intel Core i7-4770S a 3.10 GHz, com 8 GB de memória RAM e 1 TB de disco rígido, utilizando o sistema operacional Ubuntu v16.

O conjunto de características utilizado para classificação dos métodos de aprendizado de máquina foram escolhidos de acordo com os tipos de ataque contidos no conjunto

de dados utilizado. A partir das 84 características iniciais, foram selecionadas 10 características, consideradas as melhores para detectar os ataques, conforme apresentado em [Sharafaldin et al. 2018]. Os métodos de aprendizado de máquina utilizados são: árvore de decisão (DT), árvore adaptiva Hoeffding (HAT) e algoritmo bayesiano simples (NB). Os algoritmos foram implementados em linguagem Java. Para realizar a classificação em lotes, utilizaram-se as bibliotecas da ferramenta Weka [Hall et al. 2009]. Na classificação em fluxos, foram utilizadas as bibliotecas da ferramenta MOA [Bifet et al. 2011].

Na classificação em lotes, o treinamento dos algoritmos foi realizado através do algoritmo *K-fold* de validação cruzada, visando melhorar a generalização dos modelos e evitar, assim, o *overfitting* dos dados. A validação cruzada consiste em separar o conjunto de dados em k partições aleatórias sucessivamente, onde serão utilizadas $k-1$ partições no treinamento e a partição restante é utilizada na validação do modelo. Após k iterações, o modelo terá utilizado todo o conjunto inicial e é possível avaliar as estatísticas de validação dos modelos através da média de todos as iterações, proporcionando maior confiança no resultado. Foi utilizado para este trabalho, o valor de k igual a 10 iterações. O conjunto de dados inicialmente foi balanceado igualmente entre tráfego normal e ameaças, evitando classificação tendenciosa dos dados em função da classe predominante.

Na classificação em fluxos, os métodos tradicionais de avaliação de mineração de dados, como o *K-fold* de validação cruzada, não pode ser usado para validação dos algoritmos de mineração de fluxo de dados (*Data Stream Mining* - DSM) [Gama et al. 2013]. O DSM apresenta características como dados potencialmente infinitos, modelos de previsão adaptivos e grandes quantidades de informações. Bifet et al. [Bifet et al. 2010] utiliza o conceito de *RAM-hora* para avaliar o desempenho, onde a quantidade de memória utilizada pelo modelo de predição é armazenada durante o processamento. Gama et al. [Gama et al. 2009] propuseram um método de avaliação para DSM chamado método sequencial preditivo (*Predictive Sequential Method - Prequential*). Neste método de avaliação, o número de amostras classificadas incorretamente é acumulada dentro de uma janela w do total de amostras processadas. O erro e classificação pode ser calculado a cada instante i através da equação:

$$P_e(i) = \frac{1}{i} \sum_{k=1}^{i} F(y \neq y') \tag{10}$$

A partir do método sequencial preditivo é possível calcular diversas medidas estatísticas para classificação do fluxo de dados. A ferramenta MOA oferece diversas métricas de validação, tal como a acurácia. Neste trabalho, utilizam-se a acurácia e a matriz de confusão para avaliar os métodos de classificação por processamento em lotes e em fluxos.

As Tabelas 2 e 3 apresentam os resultados de acurácia obtidos pelos métodos de classificação avaliados para o conjunto de dados, considerando os cenários onde todas as características foram utilizadas e utilizando um conjunto reduzido de características.

Os resultados mostram que reduzir o número de características aumentou a acurácia dos classificadores entre 0,48% e 6,82%, sendo o bayesiano simples o maior afetado pela redução das características. Entretanto, observou-se que os algoritmos de

Tabela 2. Acurácia de classificação do conjunto de dados utilizando algoritmos de classificação em lotes.

Classificador	84 Características	10 Características
Árvore de Decisão	97,4601%	97,9299%
Árvore Adaptiva Hoeffding	90,2387%	90,8732%
Bayesiano Simples	70,7794%	75,6124%

Tabela 3. Acurácia de classificação do conjunto de dados utilizando algoritmos de classificação em fluxos.

Classificador	84 Características	10 Características
Árvore de Decisão	98,7254%	98,9683%
Árvore Adaptiva Hoeffding	95,4019%	95,8556%
Bayesiano Simples	85,9081%	88,8470%

aprendizado, em média, executaram as classificações seis vezes mais rápido utilizando apenas 10 características. Os algoritmos de processamento em fluxo apresentaram ganho na acurácia de 1,06%, 5,48% e 17,50% para os algoritmos árvore de decisões, árvore adaptiva Hoeffding e bayesiano simples, respectivamente, em comparação ao processamento em lotes. Observa-se que em ambos os métodos de classificação em lotes e por fluxos, a redução da dimensionalidade de 84 para 10 características aumenta a acurácia nos três métodos de classificação. Porém, mesmo reduzindo a dimensionalidade do método em lotes, o processamento em fluxos apresenta melhores resultados utilizando o espaço total de características. Com a redução para 10 características no método por fluxos, os resultados apresentam ainda maior acurácia.

As Tabelas 4, 5, 6, 7, 8 e 9 apresentam as matrizes de confusão dos métodos de classificação avaliados, considerando o conjunto de 10 características principais.

Tabela 4. Matriz de confusão do algoritmo Árvore de Decisão, utilizando classificação em lotes e 10 características principais.

Árvore de Decisão	Normal	Botnet	PortScan	DDoS
Normal	**1.010.171**	18	193	205
Botnet	48.736	**940.165**	7.701	13.985
PortScan	5.224	2.339	**997.793**	5.231
DDoS	47	1	0	**1.010.539**

Tabela 5. Matriz de confusão do algoritmo Árvore de Decisão, utilizando classificação em fluxos e 10 características principais.

Árvore de Decisão	Normal	Botnet	PortScan	DDoS
Normal	**1.010.390**	7	92	98
Botnet	21.399	**980.665**	3.314	5.209
PortScan	1.988	1.108	**1.005.160**	2.331
DDoS	26	0	0	**1.010.561**

Tabela 6. Matriz de confusão do algoritmo Árvore Adaptiva Hoeffding, utilizando classificação em lotes e 10 características principais.

Árvore Adaptiva Hoeffding	Normal	Botnet	PortScan	DDoS
Normal	**998.931**	1.367	2.336	7.953
Botnet	11.823	**769.107**	188.302	41.355
PortScan	65.993	18.038	**897.396**	29.160
DDoS	2.308	3	298	**1.007.978**

Tabela 7. Matriz de confusão do algoritmo Árvore Adaptiva Hoeffding, utilizando classificação em lotes e 10 características principais.

Árvore Adaptiva Hoeffding	Normal	Botnet	PortScan	DDoS
Normal	**1.005.179**	624	1.063	3.721
Botnet	5.935	**899.495**	87.822	17.335
PortScan	28.330	8.814	**961.046**	12.397
DDoS	1.362	0	128	**1.009.097**

Tabela 8. Matriz de confusão do algoritmo Bayesiano Simples, utilizando classificação em lotes e 10 características principais.

Bayesiano Simples	Normal	Botnet	PortScan	DDoS
Normal	**659.125**	28.337	101.163	221.962
Botnet	198.308	**511.462**	80.883	219.934
PortScan	63.108	10.869	**877.709**	58.901
DDoS	1.902	7	458	**1.008.220**

Tabela 9. Matriz de confusão do algoritmo Bayesiano Simples, utilizando classificação em lotes e todas as características.

Bayesiano Simples	Normal	Botnet	PortScan	DDoS
Normal	**852.183**	12.049	48.338	98.017
Botnet	87.692	**783.629**	33.874	105.392
PortScan	31.936	4.408	**946.401**	27.842
DDoS	1.098	2	196	**1.009.291**

Observando os resultados obtidos, o processamento em fluxo reduziu em até 54,55% a quantidade de falsos-positivos e falsos-negativos para o conjunto de dados utilizando 10 características e em até 66,61% quando utilizando todas as características do conjunto de dados. Este ganho em acurácia pelo processamento em fluxos é resultado da adaptação em tempo real do algorítimo, onde cada dado é primeiro classificado e depois integrado ao conjunto de dados para auxiliar na classificação de dados futuros.

7. Conclusão

Este artigo apresenta um sistema de detecção de intrusão *online* para detecção de ameaças distribuídas de rede utilizando o processamento em fluxos. São utilizados três algoritmos de aprendizado de máquina, tanto no modo de processamento em lotes como no processamento em fluxos. A base de dados é avaliada utilizando todas as características

extraídas, assim como executando uma redução no espaço para apenas 10 características.

Considerando todas as combinações de algoritmos de classificação e número de características, o processamento em fluxos apresentou uma melhora na acurácia da classificação de ameaças em relação ao processamento em lotes. Para o algoritmo bayesiano simples, o método proposto melhorou em até 17,50% a taxa de acerto e reduziu em até 66,61% a taxa de falsos-positivos e falsos-negativos. Em alguns cenários, a redução do número de características demostrou não ser tão eficiente na melhora da acurácia, porém reduziu o tempo de classificação em até seis vezes, comparado com o tempo de classificação utilizando todas as características. Considerando a grande quantidade de dados, a necessidade de uma resposta imediata e modelos de previsão compactos com baixo consumo de memória, a classificação de ameaças *online* requer técnicas de mineração de fluxo de dados para resolver as necessidades e problemas de detecção de intrusão em redes de computadores. Como trabalhos futuros, pretende-se estudar e analisar outros métodos de classificação, testar novos conjuntos de características e outras base de dados.

Referências

Bifet, A. and Gavaldà, R. (2009). Adaptive learning from evolving data streams. In *Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII*, IDA '09, pages 249–260, Berlin, Heidelberg. Springer-Verlag.

Bifet, A., Holmes, G., Pfahringer, B., and Frank, E. (2010). Fast perceptron decision tree learning from evolving data streams. In *Proceedings of the 14th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining - Volume Part II*, PAKDD'10, pages 299–310, Berlin, Heidelberg. Springer-Verlag.

Bifet, A., Holmes, G., Pfahringer, B., Read, J., Kranen, P., Kremer, H., Jansen, T., and Seidl, T. (2011). Moa: A real-time analytics open source framework. In *Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part III*, ECML PKDD'11, pages 617–620, Berlin, Heidelberg. Springer-Verlag.

Buczak, A. L. and Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys Tutorials*, 18(2):1153–1176.

Chen, F., Deng, P., Wan, J., Zhang, D., Vasilakos, A. V., and Rong, X. (2015). Data mining for the internet of things: Literature review and challenges. *International Journal of Distributed Sensor Networks*, 11(8):431047.

CICFlowMeter (2017). Cicflowmeter - a network traffic biflow generator and analyzer. Acessado em: 15-08-2018.

Corrêa, D. G., Enembreck, F., and Silla, C. N. (2017). An investigation of the hoeffding adaptive tree for the problem of network intrusion detection. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 4065–4072.

Desale, K. S., Kumathekar, C. N., and Chavan, A. P. (2015). Efficient intrusion detection system using stream data mining classification technique. In *2015 International Conference on Computing Communication Control and Automation*, pages 469–473.

Gama, J., Sebastião, R., and Rodrigues, P. P. (2009). Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 329–338, New York, NY, USA. ACM.

Gama, J., Sebastião, R., and Rodrigues, P. P. (2013). On evaluating stream learning algorithms. *Mach. Learn.*, 90(3):317–346.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37.

Gartner (2016). Gartner says by 2020, more than half of major new business processes and systems will incorporate some element of the internet of things. Acessado em: 07-11-2018.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30.

Kezih, M. and Taibi, M. (2013). Evaluation effectiveness of intrusion detection system with reduced dimension using data mining classification tools. In *2nd International Conference on Systems and Computer Science*, pages 205–209.

Kholghi, M., Hassanzadeh, H., and Keyvanpour, M. (2010). Classification and evaluation of data mining techniques for data stream requirements. In *2010 International Symposium on Computer, Communication, Control and Automation (3CA)*, volume 1, pages 474–478.

Labs, F. (2017). The hunt for iot: The rise of thingbots. Acessado em: 07-11-2018.

Lashkari, A. H., Zang, Y., Owhuo, G., Mamun, M. S. I., and Gil, G. D. (2017). Cicflowmeter - a network traffic biflow generator and analyzer. Acessado em: 15-08-2018.

Lobato, A. G. P., Andreoni, M., and Duarte, O. C. M. B. (2016). Um sistema acurado de detecção de ameaças em tempo real por processamento de fluxos.

Pecht, M. G. and Kang, M. (2019). *Machine Learning: Fundamentals*. IEEE.

Schuartz, F. C., Fonseca, M., and Munaretto, A. (2017). Sistema distribuído para detecção de ameaças em tempo real utilizando big data. In *XXXV Simpósio Brasileiro de Telecomunicações e Processamento de Sinais - SBrT 2017*.

Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2017). Cicids2017. Acessado em: 15-08-2018.

Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,*, pages 108–116. INSTICC, SciTePress.

Sun, J. (2010). Application of data mining for decision tree model of multi-variety discrete production and manufacture. In *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, pages 724–728.

Symantec (2018). Internet security threat report, volume 23. Acessado em: 07-11-2018.