Metodologia de detecção de *botnets* utilizando aprendizado de máquina

Daisy C. A. Silva, Sérgio S. C. Silva, Ronaldo M Salles

***Abstract*— The increasing number of cyber attacks against public and private organizations, as well as common users, have been reported in recent years, raising concerns about the use of such attacks in cyber warfare. In some of these actions were used structures known as botnets. This article aims to analyze the efficiency of machine learning algorithms in the detection of botnet. In this context, a methodology for the detection of botnet was structured, where after the capture and the pre-processing of the network traffic, machine learning algorithms are applied that classify the data in malicious or non malicious according to the Knowledge acquired during the training phase.**

***Resumo*— O crescente número de ataques cibernéticos contra organizações públicas e privadas, além de usuários comuns, tem sido relatados nos últimos anos, trazendo à tona a preocupação do emprego desses ataques em ações de Guerra Cibernética. Em algumas dessas ações foram empregadas estruturas conhecidas como *botnets*. Este artigo tem o objetivo de analisar a eficiência dos algoritmos de aprendizado de máquina na detecção de *botnet*. Nesse contexto, foi estruturada uma metodologia para detecção de *botnet*, onde após a captura e o pré-processamento do tráfego de rede, são aplicados algoritmos de aprendizado de máquina que classificam os dados em maliciosos ou não maliciosos de acordo com o conhecimento adquirido na fase de treinamento.**

I. Introdução

A importância do emprego das redes de computadores cresceu significativamente ao longo dos últimos anos. Atualmente não é possível pensar em desenvolver qualquer tipo de aplicação, seja para computador, celular, televisão, ou quaisquer outros dispositivos, que não utilize algum tipo de rede. Essa crescente dependência do uso das redes passou a despertar o interesse dos usuários mal intencionados que passaram a aproveitar as vulnerabilidades intrínsecas dos sistemas. O aumento das atividades maliciosas se deve, entre outras coisas, à ineficiência das atuais soluções em identificar, reduzir e interromper por completo esse tipo de atividade [1].

Concomitante ao crescimento das atividades maliciosas, houve o aumento da diversidade de ações maliciosas que são realizadas por usuários mal intencionados. Para realizar muitas dessas ações maliciosas passaram a ser empregadas estruturas denominadas *botnets*, que podem ser definidas como uma rede de máquinas infectadas por um *software* malicioso (*bot*) e controlada remotamente por um ou mais atacantes (*botmaster*) [2]. As *botnets* tornaram-se uma das maiores ameaças na Internet e especialistas acreditam que entre 16% a 25% dos usuários na Internet pertencem a uma dessas redes, sem ter conhecimento [3].

Diversos mecanismos de detecção que empregam aprendizado de máquina já foram propostos na literatura, como visto na Tabela I [4]. Apesar dos algoritmos propostos apresentarem bons resultados, os métodos empregam algoritmos específicos associados a apenas um tipo de *bot*, o que torna difícil realizar uma comparação precisa da eficiência das técnicas de aprendizado de máquina na detecção de *botnets*.

TABELA I
MÉTODOS DE DETECÇÃO

Ano	Métodos	Algoritmo	Bots	Protocolo	Métrica
2006	Livadas et al. [5]	C 4.5 Naive Bayes Bayesiano Network	Kaiten bot	IRC	Taxa FP Taxa FN
2008	Strayer et al. [6]	C 4.5 Naive Bayes Bayesiano Network	Kaiten bot	IRC	Taxa FP Taxa FN
2008	Masud et al. [7]	SVM Naive Bayes C 4.5 Boosted	SDBot Rbot	IRC	Acurácia Taxa FP Taxa FN
2010	Liao et al. [8]	C 4.5 Naive Bayes Bayesiano Network	Storm bot	P2P	Acurácia Taxa FP Taxa FN
2011	Saad et al.	SVM ANN k-NN Naive Bayes	Storm bot Waledac	P2P	Taxa TP Acurácia
2012	Bilge et al. [9]	C 4.5 SVM Floresta Aleatória	Storm bot	P2P	Taxa TP Taxa FP

Neste contexto, esse artigo apresenta uma proposta de metodologia para a detecção de *botnet* utilizando aprendizado de máquina. Essa metodologia está dividida em 2 fases. Na primeira fase, foi realizado o levantamento de *features* relevantes para a detecção de *bot* utilizando técnicas de seleção de *features*. Na segunda fase, foi analisado o desempenho dos algoritmos classificadores de aprendizado de máquina na detecção de tráfego malicioso, utilizado métricas como acurácia, precisão, sensitividade, dentre outras. E por fim, o resultado dessa análise será adequado à arquitetura de sistema integrado de defesa cibernética [10].

II. Metodologia

A metodologia proposta para detecção de botnet usando aprendizado de máquina está dividida em 2 fases, conforme ilustrado na Figura 1. A primeira fase tem como objetivo a seleção de *features* relevantes para a detecção de *botnet*. A segunda fase apresenta os algoritmos de aprendizado de máquina que obtiveram um melhor desempenho na detecção de *botnet*. Foi utilizada, nos experimentos, a base de dados da *Czech Technical University* (CTU) [11]. Técnicas de pré-processamento de dados, de aprendizado e avaliação foram empregados com o objetivo de analisar o desempenho dos algoritmos de aprendizado de máquina supervisionados na detecção de *botnets*. Os *frameworks* WEKA e LibSVM foram

empregados nos experimentos por possuírem uma série de algoritmos para mineração de dados.

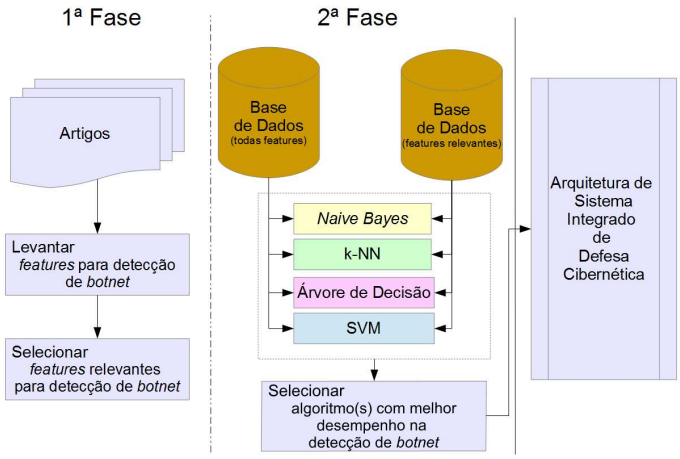

Fig. 1. Fluxograma das fases da metodologia.

A. 1ª *fase*

A primeira fase do trabalho tem como objetivo levantar quais são as *features* importantes na identificação de um tráfego malicioso.

1) Levantamento de Features*:* Foi realizado um levantamento de *features* que já foram empregadas na detecção de *botnets*. As *features* foram catalogadas e categorizadas de acordo com os diversos grupos relacionados abaixo [12]. Apesar de ser possível extrair um grande número de *features* do tráfego de rede, o verdadeiro valor dessas características para o reconhecimento de uma rede de *bots* ainda não é claro. A relação de *features* catalogadas de acordo com o grupo é dada na Tabela II.

TABELA II
LEVANTAMENTO DE *features*.

Grupo	*feature*
Baseado em fluxo	IP de origem porta de origem IP de destino porta de destino Protocolo
Baseado em *byte*	Total de *bytes* Média do tamanho do pacote Número total de pacotes com o mesmo comprimento Desvio padrão de carga útil do tamanho do pacote
Baseado em tempo	Média de *bits* por segundo Média de pacotes por segundo em uma janela de tempo Média de pacotes por segundo Média do tempo de chegada dos pacotes
Baseado em comportamento	Número de reconnects Duração do fluxo Tamanho do primeiro pacote *PageRank* Consultas negativas
Baseado em pacotes	Total de pacotes trocados Total de pacotes nulos trocados Número de pequenos pacotes trocados Razão entre o número de pacotes recebidos sobre pacotes de saída Percentagem de pequenos pacotes trocados Tamanho da comunidade Desvio padrão das consultas DNS Média de consultas DNS Média de pacotes por segundo

2) Seleção de Features*:* A seleção de *features* é uma etapa da fase de pré-processamento da descoberta de conhecimento em uma base de dados. As *feature*s são como funcionalidades ou como características, que no caso deste trabalho são as variáveis extraídas do tráfego de dados que compõe a base de dados estudada. Em geral, espera-se que todas as *features* sejam relevantes, porém nem sempre é possível garantir isso. Além disso, algumas *features* são redundantes e assim podem ser eliminadas. Como o próprio nome já diz, o objetivo é escolher um subconjunto de *features* ou criar outras *features* que substituam um conjunto deles a fim de reduzir a dimensão da base de dados. Com essa redução de dimensão, reduz-se a complexidade da base de dados e assim, o tempo de processamento para extrair algum conhecimento. Além disso, *features* desnecessárias podem causar ruído no resultado final e isto pode ser evitado com a aplicação de técnicas de seleção de *features*.

Neste artigo, a técnica de seleção de *features wrapper* foi empregada com o objetivo de obter o melhor subconjunto de *features*. O método *greedy* foi utilizado aplicando o *forward*, onde *features* são adicionadas progressivamente retornando o melhor subconjunto encontrado. E para avaliar a eficácia das *features* levantadas foi analisado a métrica de precisão de classificação: *acurácia* ou taxa de detecção. O algoritmo de classificação árvore de decisão foi o escolhido por apresentar uma abordagem de aprendizagem amplamente utilizada que demonstrou boa precisão em estudos de detecção de *botnets* [13] [14].

B. 2ª *fase*

Esta segunda fase tem o objetivo de analisar o desempenho dos algoritmos de aprendizado de máquina na detecção de *botnets*.

O critério de seleção dos algoritmos de aprendizado de máquina foi de acordo com cada paradigma de aprendizado. O paradigma estatístico utiliza métodos estatísticos para encontrar uma boa aproximação do conceito induzido, algoritmo *Naive Bayes*. No caso do paradigma simbólico, busca aprender construindo representações simbólicas, como a árvore de decisão, algoritmo J48. E o paradigma de aprendizado de máquina baseado em exemplos classifica exemplos nunca vistos por meio de exemplos similares conhecidos, algoritmo *k-Nearest Neighbors* (k-NN).

Para avaliar o desempenho de cada um desses algoritmos foi analisado o resultado das seguintes métricas: acurácia, precisão, sensitividade, eficiência, área da curva (ROC) e tempo de processamento do modelo. Uma das métricas mais usadas na avaliação de algoritmos de classificação é a acurácia. Porém, no caso de base de dados desbalanceadas, a acurácia pode não fornecer informação adequada sobre a capacidade de discriminação de um classificador em relação a um dado grupo específico de interesse.

Nesse contexto, outras métricas foram adotadas com o objetivo de fornecer avaliações mais adequadas para aplicações desbalanceadas [15]. Dentre outras métricas mais usadas, destacam-se: precisão, sensitividade, especificidade e eficiência. Porém essas métricas também não são tão eficientes

na avaliação de classificadores em cenários desbalanceados. E essa limitação, no entanto, pode ser superada através da métrica área da curva ROC (*receiver operator characteristic curve*). Essa técnica representa a relação entre a sensibilidade e a especificidade, isto é, é um gráfico de sensibilidade (taxa de verdadeiros positivos) versus especificidade (taxa de falsos positivos) [16].

Outro aspecto verificado foi os possíveis métodos para particionar a base de dados em base de treinamento e teste. Primeiramente, o *holdout* foi empregado. Esse método divide a base de dados inicial em 2/3 para treinamento e 1/3 para teste. Os dados foram selecionados de forma aleatória, para minimizar o risco de inserir viés na formação das bases. Essa proporção foi empregada por ser uma boa prática segundo [17]. Porém os cenários mais desbalanceados nÃ£o apresentaram dados da classe *botnet* na base de teste. Por essa razão, resolveu-se usar a validação cruzada (*cross-validation*). Na prática, a validação cruzada (cross-validation) com *k* igual a dez (10) vem sendo o método mais utilizado e o escolhido neste artigo.

III. Experimentos

A. Ambiente de trabalho

Os experimentos foram realizados no Laboratório de Defesa Cibernética do IME (LabDCiber-IME). Esse laboratório foi criado em 2012 com o objetivo de fornecer uma estrutura adequada para o desenvolvimento de pesquisa científica na área cibernética (ex. criptografia, segurança da informação e segurança de redes). Para realizar os experimentos, foram utilizados os seguintes recursos computacionais do laboratório: 06 Servidores DELL R910 (4 Xeon E7-4800 com 10 núcleos e 512 GB RAM, cada servidor).

B. Base de dados

Neste artigo, a base de dados primeiramente foi pré-processada com o objetivo de sofrer uma redução na sua dimensionalidade reduzindo *features* irrelevantes e extraindo novas a partir da combinação das *features* existentes. As *features Source IP Address*, *Destination IP Address*, *Source Port* e *Destination Port* foram ignoradas devido à sua ineficiência na detecção de *botnet*. Embora reconheçamos seu valor em casos específicos, como em *blacklist*, no contexto desta classificação, elas impedem uma análise mais precisa na seleção das *features* mais relevantes. Além da redução, três *features* foram calculadas: média do tamanho do pacote (total de *bytes* / total de pacotes), média de pacotes por segundo (total de pacotes/duração) e média de *bits* por segundo (total de *bytes* * 8 / duração).

A base de dados da *Czech Technical University* (CTU) [11] possui as seguintes classes: tráfego normal, *background* e *botnet*. Na classe normal encontra-se o tráfego não malicioso, na classe *botnet*, o tráfego maliciosos e na classe *background* o tráfego indeterminado. Outra característica desta base é que esta dividida em treze cenários, e cada um deles contém algum comportamento de *malware*, totalizando no mínimo 13 tipos de *bots*.

Após o pré-processamento, cada cenário sofreu uma desdobramento em janelas de tempo devido à vasta quantidade de dados existentes a serem processado. Como em [11], alguns cenários produziram mais de 395 mil pacotes por minuto. Uma pequena janela de tempo permite um melhor processamento dessas informações. Um outro motivo para esse desdobramento é que *botnets* tendem a ter um comportamento temporal, ou seja, a maior parte das ações se mantém inalteradas por muito tempo. As janelas de tempo foram compostas por um intervalo de 3 minutos (180s). Esta separação foi feita com base na *featuresStartTime*. E segundo [18], a precisão mais elevada é alcançada quando a janela de tempo é de 180s. Isto mostra que um padrão único é seguido pelas redes de *bots* quando o tamanho da janela de tempo é de 180s.

C. 1^a *fase*

O objetivo da 1^a fase é identificar as *features* relevantes para a detecção de *bots*. Conforme descrito na Subseção II-A.2 a técnica empregada em cada cenário foi o *wrapper*. O resultado de cada cenário encontra-se apresentado na Tabela III. Para cada cenário, um subconjunto de *features* foi selecionado com as mais relevantes na detecção de *botnet*.

TABELA III
RESULTADOS DA 1^a FASE - SELEÇÃO DE *features*

Cenário	***Subconjunto de features***	**Cenário**	***Subconjunto de features***
1	Status Total bytes	8	Duração Total bytes Média bits/segundo
2	Status Média bits/segundo Média pc/segundo	9	Duração Total bytes Total Src bytes
3	Duração Total bytes Média bits/segundo	10	Status Total bytes
4	Total bytes Status Total Src bytes	11	Status Total bytes Total Src bytes Média pc/segundo
5	Duração Total bytes Média bits/segundo	12	Duração Total bytes Média bits/segundo
6	Total bytes Média pc/segundo Média bits/segundo	13	Status Média bits/segundo Média tam pacote
7	Duração Total bytes Média bits/segundo		

Ao analisar os resultados, em todos os experimentos, as seguintes *features* se sobressaíam: duração, estado, total de *bytes*, média do tamanho do pacote, média de pacotes por segundo e média de *bits* por segundo. Como parte da análise, foram comparados os resultados com o trabalho apresentado por [12]. No caso de [12], foi analisado um conjunto abrangente e diversificado de *bots* apresentando uma elevada taxa de detecção de *botnet*. As *features* selecionadas foram: duração, média de *bits* por segundo, média do tamanho do pacote e taxa do número de pacotes de entrada com os de saída.

Para os experimentos apresentados por [12], os autores utilizaram o *trace* completo, que possibilita obter qualquer tipo de *feature*. Já neste trabalho, foram utilizados apenas os registros de fluxo de tráfego do *trace* que, por sua natureza, apresentam menos informações. Mesmo com a redução das informações disponíveis para a análise, percebe-se uma similaridade nos resultados. Dentre as quatro *features* selecionadas pela abordagem de [12] como as mais relevantes para a detecção de *bots*, três destas também foram identificadas nos experimentos realizados. Convém ressaltar que a *feature* taxa do número de pacotes de entrada com os de saída não pode ser obtida através de apenas uma análise de fluxo.

D. 2^a *fase*

O objetivo dessa fase é avaliar o desempenho dos algoritmos de aprendizado de máquina na detecção de *bots*. Conforme descrito na Subseção II-B, os algoritmos de aprendizado de máquina Naive Bayes, J48, SVM e k-NN foram analisados seu desempenho de acordo com os resultados obtidos pelas métricas: acurácia, precisão (*precision*), sensitividade (*recall*) e eficiência (*F-measure*), curva ROC e tempo de processamento do modelo.

O gráfico 2 apresenta os resultados da métrica Acurácia, porém essa métrica não é suficiente para avaliar um classificador na detecção de *botnet*. Devido a particularidade do conjunto de dados em possuir cenários desbalanceados, observa-se uma baixa ou nula detecção da classe *botnet* como apresentado no caso do cenário 7. Conforme a tabela IV, o algoritmo k-NN obteve 98,74% de acerto na métrica acurácia, porém 0% de acerto na métrica precisão da classe botnet, mostrando que a proporção escolhida para treinamento, mesmo de uma forma aleatória, pode conter um número mínimo ou nulo de casos marcados como *botnet*.

TABELA IV
RESULTADOS DA 2^a FASE - CENÁRIO 7 - MÉTRICA ACURÁCIA E PRECISÃO

Cenário	Algoritmo	Total instâncias (geral)	Total instâncias (classe botnet)	Métricas	
				Acurácia	Precisão
7	k-NN	114.078	63 (0,06%)	98,74%	0%

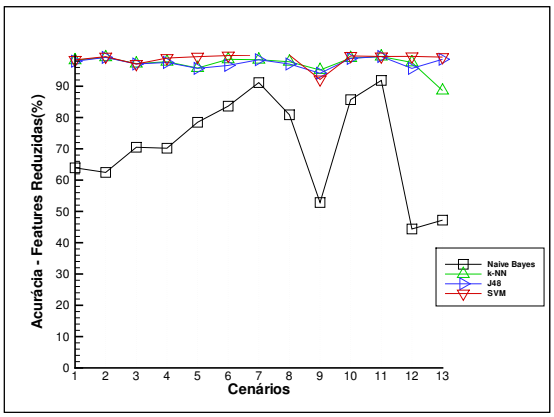

Fig. 2. Resultado da Métrica Acurácia

Dentre as outras métricas, a curva ROC apresentou resultados melhores e mais satisfatórios. Essa é uma métrica interessante para tarefas com classes desproporcionais. Nela, mede-se a área sob uma curva formada pelo gráfico entre a taxa de exemplos positivos, que realmente são positivos, e a taxa de falsos positivos.

IV. ANÁLISE DOS RESULTADOS

Os resultados experimentais demonstraram que o algoritmo *Naive Bayes* obteve um dos piores resultados em todas as métricas. O algoritmo J48 não obteve bons resultados nos cenários mais desbalanceados, com uma baixa taxa de acerto na classe *botnet*. Como por exemplo o cenário 7, que possui apenas 0,06% de tráfego malicioso, o algoritmo J48, apesar de uma taxa de acurácia de 98,47%, obeteve 0% na métrica precisão, sensitividade e eficiência. O algoritmo SVM obteve bons resultados em cenários desbalanceados, porém com um alto tempo de processamento do modelo na fase de treinamento do algoritmo, aumentando assim os custos computacionais e inviabilizando sua utilização na detecção de *botnet* em tempo real. O algoritmo k-NN obteve destaque em todas as métricas e acertou o maior número de instâncias maliciosas em cada cenário, além de ser o mais rápido.

Após a verificação das métricas precisão, sensitividade e eficiência, nenhum dos algoritmos se sobressaiam dentre eles, não gerando uma analise conclusiva em relação ao algoritmo com o melhor desempenho na detecção de *botnet*. Porém ao analisar o gráfico da curva ROC apresentado na Figura 3 percebe-se que o algoritmo k-NN obteve melhores resultados dentre os outros algoritmos. O algoritmo k-NN se destacou em 12 cenários e o SVM em apenas um cenário (cenário 3). Mesmo assim com um valor da curva ROC do algoritmo SVM aproximado ao valor do k-NN. O algoritmo J48 não se destacou, além de apresentar um dos piores resultados em todos os cenários ao se comparar com o k-NN e o SVM. Pode-se concluir que o algoritmo k-NN obteve os melhores resultados ao se analisar seus valores em todas as famílias de *bot*, nos cenários com alto e baixo percentual de tráfego malicioso, nos cenários pequenos e grandes e dentre os protocolos analisados (IRC, HTTP, P2P).

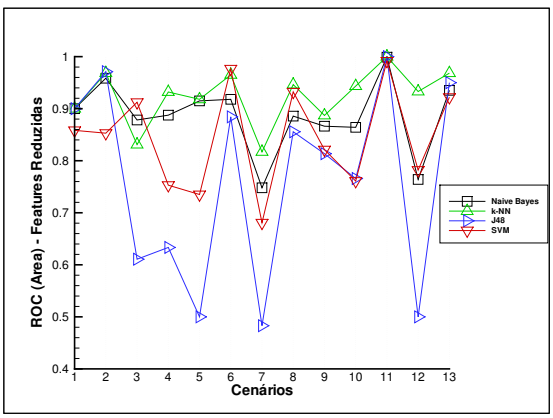

Fig. 3. Área da curva ROC

V. Considerações Finais

A detecção de *botnet* com base no aprendizado de máquina tem sido um tema bastante procurado ultimamente, resultando em inúmeros métodos de detecção baseados em diversas heurísticas, que visam vários tipos de *botnet* usando uma variedade de algoritmos. Este artigo apresenta uma proposta de metodologia para a detecção de *botnet* utilizando aprendizado de máquina, no qual foram analisados o desempenho e a eficiência de algoritmos supervisionados. Essa metodologia está dividida em 2 fases. Na primeira fase, foi realizado o levantamento de *features* relevantes para a detecção de *bot* utilizando técnicas de seleção de *features*. Na segunda fase, foi analisado o desempenho dos algoritmos classificadores de aprendizado de máquina na detecção de tráfego malicioso, utilizado métricas como acurácia, precisão, área da curva ROC, dentre outras.

Na primeira fase desse trabalho, o subconjunto de *features* relevantes gerados foi comparado com abordagem recentemente desenvolvida por [12]. No caso de [12], foi analisado um conjunto abrangente e diversificado de *bots* apresentando uma elevada taxa de detecção. Neste trabalho, o conjunto de dados é mais reduzido, porém os resultados foram similares. Dentre as quatro *features* selecionadas como as mais relevantes na abordagem de [12], três destas *features* estão presentes no subconjunto deste trabalho. Concluindo que mesmo com uma base de dados reduzida é possível obter bons resultados na detecção de *botnet*, facilitando o processamento e análise de tráfego de rede em tempo real.

Na segunda fase foi analisado o desempenho dos algoritmos de aprendizado de máquina na detecção de *botnets*. Com esse objetivo, foi utilizado nos experimentos a base de dados do tráfego de rede da *Czech Technical University* (CTU) [11] reduzida pela seleção de *features* realizada na primeira fase. Ao processar os algoritmos, observou-se que dentre os quatro algoritmos classificadores analisados (*Naive Bayes*, k-NN, J48 e SVM), o algoritmo *Naive Bayes* obteve os piores resultados. Todos os demais algoritmos mantiveram altas taxas de acurácia, demonstrando como essa métrica é altamente suscetível a um conjunto de dados desbalanceado e pode facilmente induzir a uma conclusão errada sobre o desempenho do algoritmo. Nesse contexto, outras métricas foram analisadas, como a precisão, sensitividade, eficiência, área da curva ROC e tempo de processamento do modelo. Dentre as métricas, a área da curva ROC fornece uma melhor análise do desempenho dos algoritmos, observando-se que os algoritmos k-NN e SVM apresentaram os melhores resultados nessa métrica. Ao se comparar os algoritmos k-NN e SVM, o algoritmo SVM obteve bons resultados com classe rara, porém com um tempo de processamento muito elevado. Em contrapartida, o algoritmo k-NN apresentou bons resultados na maioria dos cenários, com baixas taxas de erro e altas taxas de acertos na classe *botnet*, além de ser um algoritmo rápido com o menor tempo de processamento do modelo em todos os cenários. Conclui-se a segunda fase com o destaque para o algoritmo k-NN na análise do tráfego de rede em tempo real.

Ao analisar os resultados, observou-se também, que a acurácia não é uma métrica suficiente na avaliação de um classificador. Foi observado na maioria dos algoritmos uma elevada acurácia, isto é, os algoritmos apresentaram uma alta taxa de acertos. Porém, ao analisar a precisão, observa-se baixos resultados, concluindo que os algoritmos obtiveram uma alta taxa de acertos com as classes não *botnet*, mas uma baixa taxa de acertos para a classe *botnet*. Por isso, é importante que sejam analisadas outras métricas que possibilitam avaliar o desempenho do classificador para cada classe da base de dados, tais como, precisão (*precision*), sensitividade (*recall*) e eficiência (*F-measure*).

Referências

[1] N. Ianelli and A. Hackworth, "EstatÃsticas dos incidentes reportados ao cert.br," 2016, carnegie Mellon University. [Online]. Available: http://www.cert.br/stats/incidentes/

[2] S. S. Silva, R. M. Silva, R. C. Pinto, and R. M. Salles, "Botnets: A survey," *Computer Networks*, vol. 57, no. 2, pp. 378 – 403, 2013, botnet Activity: Analysis, Detection and Shutdown. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128612003568

[3] B. AsSadhan, J. Moura, D. Lapsley, C. Jones, and W. Strayer, "Detecting botnets using command and control traffic," in *Network Computing and Applications, 2009. NCA 2009. Eighth IEEE International Symposium on*, Jul. 2009, intl, pp. 156–162.

[4] M. Stevanovic and J. M. Pedersen, "Machine learning for identifying botnet network traffic," Aalborg University, Tech. Rep., 2013.

[5] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, "Usilng machine learning technliques to identify botnet traffic," in *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*. IEEE, 2006, intl, pp. 967–974.

[6] W. T. Strayer, D. Lapsely, R. Walsh, and C. Livadas, "Botnet detection based on network behavior," in *Botnet Detection*. Springer, 2008, intl, pp. 1–24.

[7] M. M. Masud, T. Al-khateeb, L. Khan, B. Thuraisingham, and K. W. Hamlen, "Flow-based identification of botnet traffic by mining multiple log files," in *Distributed Framework and Applications, 2008. DFmA 2008. First International Conference on*. IEEE, 2008, intl, pp. 200–206.

[8] W.-H. Liao and C.-C. Chang, "Peer to peer botnet detection using data mining scheme," in *International Conference on Internet Technology and Applications*. IEEE Computer Society, 2010, intl, pp. 1–4. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5566407

[9] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "Disclosure: detecting botnet command and control servers through large-scale netflow analysis," in *Proceedings of the 28th Annual Computer Security Applications Conference*. ACM, 2012, intl, pp. 129–138.

[10] S. S. C. Silva and R. M. Salles, "Arquitetura de um sistema integrado de defesa cibernética para detecção de botnets," vol. 1, pp. 303–309, 2012.

[11] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers & Security*, vol. 45, pp. 100 – 123, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404814000923

[12] E. Beigi, H. Jazi, N. Stakhanova, and A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in *Communications and Network Security (CNS), 2014 IEEE Conference on*, 2014, intl, pp. 247–255.

[13] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant, "Botnet detection based on traffic behavior analysis and flow intervals," *Computers & Security*, vol. 39, pp. 2–16, 2013.

[14] P. Sangkatsanee, N. Wattanapongsakorn, and C. Charnsripinyo, "Practical real-time intrusion detection using machine learning approaches," *Computer Communications*, vol. 34, no. 18, pp. 2227–2235, 2011.

[15] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[16] R. C. Prati, G. Batista, and M. C. Monard, "Evaluating classifiers using roc curves," *IEEE Latin America Transactions*, vol. 2, no. 6, pp. 215–222, 2008.

[17] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning From Data*. AMLBook, 2012.

[18] G. Kirubavathi and R. Anitha, "Botnet detection via mining of traffic flow characteristics," *Computers & Electrical Engineering*, vol. 50, pp. 91–101, 2016.