Study on Machine Learning Techniques for Botnet Detection

L. Silva, L. Utimura, K. Costa, M. Silva, and S. Prado

***Abstract*—This paper presents a study on the application of machine learning techniques for botnet detection, compromised computer networks controlled by an attacker in order to perform malicious activities, such as distributed denial-of-service attacks (DDoS), data theft and others. The study aims to evaluate the efficiency of commonly used classifiers in the literature for botnet traffic classification and, to this end, we compare the results obtained from each classifier using two different approaches for feature selection, the first one taking into account the most frequently used features in problems of this nature, based on previous works, and the second one taking into account features selected by the Recursive Feature Elimination algorithm, a relatively unexplored feature selection method in the botnet detection area.**

***Index Terms*—Botnet, Machine learning, Recursive feature elimination.**

I. Introdução

UMA das ameaças mais graves na área de cibersegurança é o uso coordenado de uma grande quantidade de máquinas para ataques, as chamadas *botnets*. Uma *botnet* é uma rede de computadores infectados que, sem o consentimento de seus donos, têm seus recursos utilizados por um atacante (denominado *botmaster*) para a execução de atividades maliciosas [1]. Na prática, o *botmaster* envia seus comandos de ataque para uma estrutura central denominada "Servidor de Comando e Controle (C&C)", responsável por transmitir a mensagem do *botmaster* para todas as máquinas infectadas, dando início às atividades maliciosas.

Existem várias classificações de *botnets* dependendo do protocolo utilizado pelo Servidor de C&C. Da mesma forma, também existem várias classificações de *botnets* que não utilizam um Servidor de C&C, isto é, que trabalham de maneira descentralizada através de protocolos *peer-to-peer* (P2P) [2].

Detectar o tráfego de *botnets* é um grande desafio pois, como elas utilizam protocolos de aplicações já existentes, a detecção de sua presença na rede acaba sendo não trivial, e a classificação do seu tráfego se torna ainda mais desafiadora devido ao uso de encriptação no conteúdo dos pacotes de rede [3].

Embora os *honeypots* – dispositivos de segurança projetados para atrair atividades maliciosas para si, a fim de estudá-las [4] – sejam eficientes para analisar o comportamento de uma *botnet* e suas características de funcionamento, eles não são capazes de detectá-las. Tendo isto em vista, foram desenvolvidas algumas técnicas para a detecção de *botnets*, baseadas em assinatura, anomalia, comportamento ou rede [5].

Apesar das *botnets* já existirem há muito tempo, o problema se torna ainda mais grave com o avanço da tecnologia da Internet das Coisas (*Internet of Things* – IoT). Muitos dispositivos de IoT são desenvolvidos sem a devida atenção à segurança do aparelho, o que os torna vulneráveis a ataques e, consequentemente, ao comando de *botnets* [6]. Um caso relativamente recente de ataque que partiu da exploração de dispositivos IoT foi a descoberta da *botnet* Mirai, em 2016. Após a infecção de um dispositivo inicial, o *malware* desta *botnet* realizava uma busca na rede por dispositivos IoT que pudessem ser facilmente invadidos e transformados em *bots*, devido às fracas configurações de segurança presentes nestes aparelhos [7].

Após sua descoberta, o código fonte da *botnet* foi disponibilizado na Internet, o que levou ao surgimento de vários projetos variantes da Mirai. Estima-se que ela foi capaz de controlar aproximadamente meio milhão de dispositivos IoT e responsável por um dos maiores e mais catastróficos ataques DDoS (*Distributed Denial of Service*) [8].

Tendo em vista essa nova tendência de *botnets*, faz-se necessário o estudo de métodos eficientes para detectá-las, a fim de garantir que ataques dessa natureza possam ser evitados. Neste trabalho, foram levados em consideração os tráfegos de *botnets* que utilizam o protocolo *Internet Relay Chat* (IRC). Este protocolo de aplicação, popularmente utilizado por programas de grandes salas de bate-papo, é historicamente predominante entre os Servidores de C&C das *botnets* [9], apesar do seu uso ter diminuído nos últimos anos em favor do Protocolo de Transferência de Hipertexto (*Hypertext Transfer Protocol* – HTTP) [10].

De modo a contribuir para o desenvolvimento de soluções cada vez mais eficientes no combate de *botnets*, este estudo visa avaliar a eficiência de diversos classificadores comumente empregados na literatura para a detecção de *botnets*. Por se tratar de um problema que envolve a área de Aprendizado de Máquina e, sabendo-se que diversas características de fluxos de tráfego de *botnets* podem ser consideradas para resolver um mesmo problema, este trabalho se compromete, também, em comparar os resultados obtidos de cada classificador por meio de duas abordagens: (i) uma levando em consideração um conjunto de características popularmente exploradas em trabalhos previamente desenvolvidos na área, realizando buscas por força-bruta para identificar possíveis subconjuntos ótimos, e (ii) outra levando em consideração características selecionadas exclusivamente através do algoritmo de *Recursive*

L. F. B. Silva, Universidade Estadual Paulista "Júlio de Mesquita Filho", Bauru, São Paulo, Brasil, luis.bueno96@gmail.com.

L. N. Utimura, Universidade Estadual Paulista "Júlio de Mesquita Filho", Bauru, São Paulo, Brasil, luan.utimura@gmail.com.

K. A. P. Costa, Universidade Estadual Paulista "Júlio de Mesquita Filho", Bauru, São Paulo, Brasil, kelton.costa@gmail.com.

M. A. Z. M. Silva, Universidade Estadual Paulista "Júlio de Mesquita Filho", Bauru, São Paulo, Brasil, marcia.zanoli@unesp.br.

S. G. D. Prado, Universidade Estadual Paulista "Júlio de Mesquita Filho", Bauru, São Paulo, Brasil, simone.prado@unesp.br.

Feature Elimination, um método relativamente pouco explorado na área de detecção de *botnets* e, consequentemente, uma das principais contribuições deste trabalho. Ademais, são destacadas as características que demonstraram um maior impacto positivo no desempenho dos classificadores conforme observado nas duas abordagens descritas anteriormente.

Este artigo está estruturado da seguinte forma: a Seção II apresenta um breve resumo sobre os trabalhos relacionados. A Seção III descreve, em maiores detalhes, as técnicas estudadas para a realização deste trabalho. A Seção IV descreve o método de pesquisa, a base de dados utilizada, o pré-processamento dos dados e a execução dos algoritmos. As Seções V e VI apresentam os experimentos realizados e os resultados obtidos, respectivamente. Por fim, a Seção VII apresenta as considerações finais deste trabalho.

II. Trabalhos Relacionados

A detecção de *botnets* através do uso de técnicas de Aprendizado de Máquina é uma área que tem sido bastante explorada nos últimos anos no meio acadêmico. Dentre os principais aspectos que diferenciam os trabalhos já desenvolvidos no meio, estão as técnicas utilizadas, as características consideradas e, também, o foco da detecção ou classificação. A seguir, é apresentado um breve resumo sobre os trabalhos relacionados que mais se aproximam do enfoque do nosso trabalho, a fim de destacar possíveis contribuições.

Bapat et al. [10] utilizou uma variação leve da técnica de Regressão Logística para identificar as características de fluxo de tráfego de rede mais eficientes para a tarefa de classificação de fluxos de *botnet* com base em 8 famílias de redes diferentes. Nosso trabalho se diferencia desse ao abranger a utilização de outras técnicas de aprendizado de máquina e, no que diz respeito à seleção de características, ao invés de utilizar a regressão de lasso, levar em consideração tanto as características mais populares utilizadas em trabalhos prévios da área quanto as características exclusivamente selecionadas pela técnica *Recursive Feature Elimination*.

Chen et al. [11] propôs um sistema de detecção de *botnets* que provou-se eficaz na identificação de *botnets* P2P. Essa abordagem extrai uma versão convolucional de características baseadas em fluxo eficazes e treina um modelo de classificação usando uma rede neural artificial *feed-forward*. Os resultados experimentais mostraram que o modelo obteve 94,7% de acurácia nos conjuntos de dados de *botnets* P2P conhecidas, chegando até 98,6% com um estágio de *confidence testing*, que classifica fluxos de confiança insuficiente na rede neural. Nosso trabalho se diferencia desse ao trabalhar com *botnets* de protocolo IRC, ao invés de P2P, e utilizar técnicas tradicionais de Aprendizado de Máquina no lugar de técnicas de Aprendizagem Profunda. Além disso, enquanto em Chen et al. [11] foram consideradas as características baseadas em fluxo de maior impacto segundo trabalhos correlatos, em nosso trabalho foi aplicada uma abordagem adicional que faz uso de uma técnica de seleção de características relativamente pouco explorada na área, para fins de comparação.

Mahardhika et al. [12] propôs um estudo comparativo da eficiência das técnicas *Naive Bayes*, Árvores de Decisão, *K-Nearest Neighbors* e Regra de Indução para a tarefa de classificação de fluxos de *botnets*. Em conjunto com a técnica de Validação Cruzada K-Fold ($K = 10$), foi observado um valor máximo de acurácia de 98,8%. Nosso trabalho se assemelha a esse por ser um estudo comparativo da eficiência de vários algoritmos – alguns, inclusive, utilizados nesse estudo, como as Árvores de Decisão e o *Naive Bayes* – e, também, por utilizar a técnica de Validação Cruzada. No entanto, nosso trabalho se diferencia desse ao empregar, em uma das abordagens, uma técnica para a seleção de características que potencializou o desempenho das técnicas utilizadas.

Kant et. al. [13] desenvolveu um método de identificação de *botnets* reunindo medidas de fluxo de sistema incorporadas utilizando a abordagem *Open Flow*. Neste trabalho, foram comparadas as técnicas de Redes Neurais Convolucionais com algoritmos mais tradicionais da área, sendo eles, Máquina de Vetores de Suporte, *Naive Bayes* e Florestas Aleatórias. Nosso trabalho, apesar de utilizar algumas das técnicas que esse trabalho também estudou, contou com uma abordagem diferente para trabalhar com os fluxos de *botnets*, discutida em maiores detalhes nas próximas seções.

III. Aprendizado de Máquina

O Aprendizado de Máquina (*Machine Learning* – ML) é, em sua essência, a extração de conhecimento a partir de bases de dados. Por envolver áreas como a Estatística, a Inteligência Artificial e a Ciência da Computação, o ML é popularmente visto como uma subárea que tange tópicos como a análise preditiva e o aprendizado estatístico [14] e que compreende métodos computacionais para a aquisição de novos conhecimentos, novas habilidades e novas maneiras de organizar o conhecimento existente [15].

De modo geral, o principal objetivo do ML é a construção de novos algoritmos e o aprimoramento de algoritmos existentes para que eles possam aprender a partir de bases de dados, visando a construção de modelos generalizáveis capazes de realizar predições precisas e/ou encontrar padrões em dados desconhecidos [14]. Na prática, o aprendizado desses algoritmos pode seguir diversas abordagens, sendo as mais populares a de aprendizado *supervisionado* e a de aprendizado *não supervisionado*. Outras abordagens comumente empregadas dignas de menção são a de aprendizado *semi-supervisionado* e a de aprendizado *por reforço*.

No aprendizado supervisionado, um *supervisor* fornece o rótulo categórico (ou custo) para cada padrão no conjunto de treinamento, de modo que o objetivo do problema passa a ser a redução do somatório de custos para estes padrões [16]. Dependendo da natureza do problema, os algoritmos de aprendizado supervisionado podem ser interpretados como *classificadores* ou *regressores*.

Em problemas de classificação, o objetivo é predizer um rótulo de classe, que é intrisecamente uma escolha de uma lista predefinida de possibilidades. Um problema dessa natureza seria, por exemplo, identificar frutas presentes em uma determinada imagem. Os problemas de classificação podem ser tanto binários, isto é, envolver apenas duas classes, como multiclasse, envolver mais de duas classes. Já nos problemas

de regressão, o objetivo é predizer um número contínuo com um valor. Neste caso, predizer o salário anual de um indivíduo com base em seu nível de educação, idade e local de residência seria um exemplo de problema de regressão [14].

No aprendizado não supervisionado, não existe um supervisor para instruir o algoritmo de aprendizado [14]. Em problemas de aprendizado não supervisionado, o objetivo pode ser a descoberta de grupos de exemplos similares nos dados (clusterização), determinar a distribuição dos dados dentro do espaço de entrada (estimativa de densidade) ou projetar os dados de um espaço de n-dimensões para outro espaço de duas ou três dimensões, para fins de visualização [17].

Neste trabalho, o foco é voltado para o estudo de técnicas de aprendizado de máquina supervisionadas para a detecção de *botnets*. A seguir, são apresentadas breves explicações sobre os métodos utilizados.

A. Naive Bayes

Naive Bayes (NB) são um conjunto de algoritmos de aprendizados supervisionados baseados no Teorema de Bayes. Esses algoritmos partem do pressuposto de que existe uma independência condicional entre todos os atributos (ou características) dos exemplos, dado o contexto da classe [18]. Por esse motivo, eles são considerados os mais simples dos modelos probabilísticos bayesianos [19].

Apesar da suposição "inocente", os classificadores NB também são considerados um dos mais eficientes e efetivos algoritmos da área de ML, com destaque as suas aplicações na área de classificação de textos [20].

O que diferencia os tipos de algoritmos *Naive Bayes* é o modo como a função de probabilidade condicional (verossimilhança) é estimada, isto é, qual a distribuição utilizada. Normalmente, as distribuições mais utilizadas são a Multinomial, a Gaussiana e a Bernoulli [21].

B. Máquina de Vetores de Suporte

O algoritmo de Máquina de Vetores de Suporte, em inglês *Support Vector Machine* (SVM), foi criado na década de 90 por Cortes e Vapnik [22] e, desde então, tem recebido muita atenção da comunidade de ML por conta do seu grande poder de generalização em diversos domínios e robustez a dados de alta dimensionalidade [23], [24].

A ideia básica do SVM consiste em encontrar um hiperplano ótimo que seja capaz de separar os dados das classes envolvidas com uma determinada margem máxima. Quando o problema em questão é não linear e, consequentemente, não é possível estabelecer um hiperplano ótimo entre as classes no próprio espaço de entrada, o SVM utiliza funções *kernel* para mapear, de forma econômica computacionalmente, os dados do espaço de entrada para um espaço de alta dimensionalidade onde, finalmente, as classes passam a ser linearmente separáveis. Trata-se de uma estratégia popularmente conhecida na literatura como *kernel trick*.

C. Árvores de Decisão

O classificador Árvore de Decisão é uma das possíveis abordagens para a tomada de decisão multiestágio. A ideia básica por trás de qualquer abordagem multiestágio é quebrar uma decisão complexa em uma série de decisões mais simples esperando que a solução final, obtida por meio dessa estratégia, se assemelhe com a solução esperada para o problema em questão [25].

Na literatura, tal estratégia é chamada de *dividir e conquistar*. Para definir os testes e divisões, as Árvores de Decisão trabalham com uma abordagem denominada *splitting* onde, de maneira recursiva, os dados são divididos em regiões menores de acordo com as classes, buscando também minimizar a entropia, ou seja, a aleatoriedade das decisões tomadas.

Uma árvore de decisão possui três tipos de nós:

- **Nó raiz:** O nó inicial da árvore, por onde começa o processo de divisão;
- **Nós de decisão:** Os nós que contêm cada uma das decisões feitas pelo algoritmo, definem através de comparações qual o caminho a ser percorrido pela árvore;
- **Nós folha:** Os nós presentes no final da árvore. Representam o resultado final de uma combinação de decisões ou eventos.

Primeiramente, o algoritmo busca uma característica inicial para iniciar as divisões e ser o nó raiz, que no caso é a que pode dividir da melhor maneira possível o conjunto de dados de modo que as classes fiquem bem separadas, podendo inclusive dividir em mais do que duas regiões. Após a divisão, são adicionados nós de teste à árvore, e novas características são selecionadas para a divisão do próximo passo.

Existem duas condições de parada para os algoritmos: 1) as características possíveis para o problema se esgotarem ou 2) até que todas as instâncias das sub-regiões sejam pertencentes a uma mesma classe [26].

Apesar destes métodos serem bem vantajosos devido à fácil compreensão e intepretação, eles infelizmente não são muito robustos por serem muito suscetíveis a perdas de precisão causadas por pequenas mudanças nas características, além de serem mais suscetíveis ao super treinamento.

D. Florestas Aleatórias

Florestas Aleatórias é um método baseado em Árvores de Decisão. O algoritmo funciona com base em um número n, que determina quantas árvores são criadas, e para cada árvore n é selecionado de maneira aleatória um subconjunto de p características que esta árvore utilizará, considerando um número t de total de características [26]. Este procedimento é muito eficiente, pois ao fazer com que as árvores sejam criadas com um número reduzido de características, elimina-se a possibilidade de que uma que seja muito mais forte do que as outras influencie tanto no resultado, de maneira que as árvores geradas sejam bem diferentes umas das outras e possuam um valor de variância mais alto entre elas [27].

Ao fim deste processo, as predições são feitas através de um sistema de votação. Os dados são passados por todas as árvores da floresta onde cada uma irá predizer uma classe e, enfim, os votos serão contados e o resultado será definido pela classe com maior número de predições.

E. AdaBoost

AdaBoost, abreviação de *Adaptive Boosting*, é um algoritmo formulado por Freund e Schapire [28], que busca melhorar a performance de outros classificadores. No contexto desse trabalho, será abordado a utilização deste algoritmo em conjunto com Árvores de Decisão. O *AdaBoost* possui semelhanças com o algoritmo de Florestas Aleatórias, visto que ele busca trabalhar com um conjunto de classificadores de forma a obter um resultado mais confiável, porém se diferencia ao obter no final de seu processo, um modelo ao invés de utilizar a abordagem de votação [29].

F. Recursive Feature Elimination

Recursive Feature Elimination (RFE) é um algoritmo recursivo que busca encontrar um subconjunto ótimo de características para um classificador específico, através do processo de atribuição de um ranking de importância de cada uma delas.

O algoritmo começa sua execução com todas as características do conjunto passado, cria um modelo com base nelas, e calcula a importância de cada uma para a execução e performance do modelo [30]. Depois disso, o algoritmo elimina a característica que teve o menor valor de importância e repete o processo recursivamente.

Ao final da execução, o algoritmo seleciona as n características mais bem colocadas em seu ranking, obtendo assim o melhor conjunto possível, sendo n um número pré-definido.

IV. Método de Pesquisa

Para a realização deste trabalho, foi utilizada a seguinte metodologia de pesquisa:

(a) Inicialmente, foi feito um levantamento bibliográfico do estado da arte da área de detecção de *botnets* do protocolo IRC, principalmente no que diz respeito à aplicação de técnicas de ML para este fim;
(b) Com base neste estudo inicial, foram selecionados as técnicas mais utilizadas na literatura para essa tarefa, a fim de incorporá-las neste trabalho;
(c) Em seguida, foi feito o pré-processamento da base de dados escolhida para este trabalho, executando todos os passos necessários para que ela fosse processada pelas técnicas de ML;
(d) Após o pré-processamento da base de dados, foi feita a execução de todos as técnicas em duas abordagens, uma realizando a busca por características de maior relevância com uma estratégia de força bruta, e outra com o algoritmo RFE;
(e) Por fim, foi feito o levantamento dos resultados obtidos em ambas abordagens.

Nesta seção, descrevemos mais a fundo os passos de desenvolvimento de cada fase.

A. Base de Dados

Para o desenvolvimento deste artigo foi utilizada uma base de dados de *botnets* fornecida pela *University of New Brunswick* [31]. Esta base foi construída a partir de outros três conjuntos de dados da área, buscando possuir não só uma grande variedade de protocolos disponíveis, mas também apresentar vários tipos de diferentes arquiteturas de rede dentro de cada protocolo de comunicação. No contexto do protocolo IRC, se fazem presentes as seguintes *botnets*: Neris, Rbot, Menti, Murlo e Tbot. Destas, se destacam a Neris e Rbot por possuírem um maior volume de dados, consistindo de 12% e 22% dos fluxos de dados no conjunto de treino, respectivamente.

Esta base vem sendo utilizada em vários estudos recentes da área [12], [11] e, portanto, sua utilização é de extrema relevância. Foram fornecidos três arquivos no formato *Packet Capture* (PCAP), sendo um voltado para a fase de treinamento e dois para testes.

Todo o desenvolvimento deste trabalho, assim como seus resultados, foi baseado somente no arquivo de conjunto de treino fornecido e balanceado para manter uma proporção de 50% de fluxos maliciosos e 50% de fluxos lícitos. Além disso, para as abordagens que não fizeram o uso da Validação Cruzada (todas exceto a RFE), foi utilizada a técnica de *hold-out* para dividir a base de dados, com proporções de 70% para o conjunto de treino e 30% para o conjunto de teste.

Para que os estimadores pudessem trabalhar no conjunto de dados, foi preciso primeiro passar o arquivo de pacotes por um gerador de fluxos bidirecionais, um programa capaz de analisar todos os pacotes e extrair características dos fluxos de rede presentes nele. Um fluxo de rede pode ser entendido como um conjunto de pacotes que foram transmitidos entre um IP de origem e um IP de destino, em um certo intervalo de tempo [32].

Para realizar a transformação de pacotes de rede para conjuntos de fluxos bidirecionais, foi utilizado o programa *flowtbag*, uma ferramenta *open-source* desenvolvida na linguagem Go, disponível em um repositório do *GitHub* [33].

B. Pré-Processamento de Dados

Depois de gerados os arquivos em formato CSV, foi preciso fazer uma análise e processamento dos dados de forma que eles fossem usados pelos estimadores. Para isto, foram utilizadas funcionalidades do *Pandas*, uma biblioteca *Python* que apresenta funções para se trabalhar com dados estruturados, desde seu armazenamento e leitura, até a execução de operações em linhas e colunas.

Neste contexto, foram utilizadas funções para filtragem seletiva dos fluxos para eliminar dados indesejados, além de utilizar ferramentas para fazer a criação de novas características.

O programa gerador utilizado faz o processamento de conexões que utilizam o protocolo TCP (*Transmition Control Protocol*) e UDP (*User Datagram Protocol*), porém como o foco do projeto é analisar o fluxo de *botnets* do protocolo IRC, que por sua vez utilizam o protocolo TCP para transporte [34], foi feita uma filtragem para que somente os protocolos desse tipo fossem mantidos na base de dados, utilizando como base o fato de que os fluxos de protocolo TCP eram representados pelo número 6, enquanto o número 17 representava os fluxos UDP. Dessa maneira, foi necessário somente filtrar os fluxos com número de protocolo igual a 6.

Depois de eliminado os fluxos que não iriam ser utilizados, foi feita a rotulação dos dados, atribuindo para cada uma das linhas da tabela um valor que indica se aquele fluxo é de *botnet* ou não. Para essa tarefa foram utilizados dois recursos: uma lista de IPs fornecida juntamente com a base de dados que informa o IP usado pelas *botnets*, e uma tabela pertencente ao estudo desenvolvido em conjunto com a base, que informa quais são as *botnets* presentes em cada conjunto.

Com essas informações, foi possível então desenvolver um método com a biblioteca Pandas que iterasse por toda a tabela verificando se os IPs dos fluxos correspondiam àqueles dos ataques.

Além disso, foram também eliminados todos os fluxos de *botnets* que não fossem do protocolo IRC, evitando que houvesse interferência dessas informações nos resultados, tendo em vista que apesar de utilizarem protocolos diferentes, as redes ainda podem compartilhar certos tipos de comportamentos gerais, o que poderia acarretar predições equivocadas.

Foi necessário também criar algumas colunas de dados que não foram geradas automaticamente pelo programa. Como citado na Seção anterior, o programa gerador divide as características em dois tipos, *forward* e *backward*, o que permite uma maior flexibilidade ao se trabalhar com os dados, porém exige que alguns cálculos sejam feitos para se gerar certas características mais gerais.

C. Execução dos Algoritmos

Contando todas as colunas válidas do gerador e as características geradas manualmente e desconsiderando a coluna de rótulos, têm-se 39 colunas de características.

No entanto, não é viável executar os estimadores com todas elas, mas sim tentar encontrar um subconjunto destas que melhor caracterize o problema. Para este fim, foram aplicadas duas abordagens:

- **Busca por força bruta segundo dados de pesquisas anteriores:** consiste em utilizar somente as características predominantes em trabalhos da área (mostradas na Tabela I) e, através de uma busca por força bruta, testar todas as combinações de características e algoritmos a fim de encontrar o melhor conjunto de características para cada classificador;
- **Busca utilizando um algoritmo específico de seleção de características:** consiste em utilizar o algoritmo *Recursive Feature Elimination* em todo o conjunto de características para observar o quão eficaz são os dados utilizados historicamente na área de detecção de *botnets* para a tarefa de classificação e, também, identificar se algumas das características fornecidas pelo gerador são eficazes nesse processo, através da observação das características escolhidas pelo algoritmo. Para esta tarefa, foram utilizadas as características presentes nas Tabelas I e II. Outro ponto dessa abordagem é verificar se as características utilizadas nos estudos anteriores possuem grande representatividade dado todo o conjunto de características, ou seja, se elas serão escolhidas pelo algoritmo.

Para a execução dos algoritmos, foi utilizada a biblioteca *Scikit-learn*, que apresenta todos os métodos propostos pelo trabalho implementado, além da técnica de *Recursive Feature Elimination* e outras funcionalidades úteis para a execução do trabalho, por exemplo, a obtenção de métricas.

TABELA I
CARACTERÍSTICAS UTILIZADAS NAS DUAS ABORDAGENS

Característica	Descrição
duration	Duração do fluxo
bps	Média de *bytes* enviados por segundo
variat	Variância média de tempo entre chegada de pacotes
pps	Média de pacotes enviados por segundo
totalpackets	Total de pacotes enviados
avgpayloadlength	Média de tamanho de *payload*
bpp	Média de *bytes* por pacote
avgiat	Média de tempo entre chegada de pacotes
totalbytes	Total de *bytes* enviados
iopr	Relação entre o número de pacotes enviados e recebidos
pctpacketpushed	Porcentagem de pacotes enviados

TABELA II
CARACTERÍSTICAS ADICIONAIS UTILIZADAS SOMENTE NA SEGUNDA ABORDAGEM

Característica	Descrição
totalfpackets	Pacotes enviados da origem para o destino
totalfvolume	Bytes enviados da origem para o destino
totalbpackets	Pacotes enviados do destino para a origem
totalbvolume	Bytes enviados do destino para a origem
minfpktl	Tamanho mínimo de pacote enviado da origem para o destino
meanfpktl	Média de tamanho de pacote enviado da origem para o destino
maxfpktl	Tamanho máximo de pacote enviado da origem para o destino
stdfpktl	Desvio padrão de tamanho de pacote enviado da origem para o destino
minbpktl	Tamanho mínimo de pacote enviado do destino para a origem
meanbpktl	Média de tamanho de pacote enviado do destino para a origem
maxbpktl	Tamanho máximo de pacote enviado do destino para a origem
stdbpktl	Desvio padrão de tamanho de pacote enviado do destino para a origem
minfiat	Mínimo de intervalo entre chegada de pacotes da origem para o destino
meanfiat	Média de intervalo entre chegada de pacotes da origem para o destino
maxfiat	Máximo de intervalo entre chegada de pacotes da origem para o destino
stdfiat	Desvio padrão de intervalo entre chegada de pacotes da origem para o destino
minbiat	Mínimo de intervalo entre chegada de pacotes do destino para a origem
meanbiat	Média de intervalo entre chegada de pacotes do destino para a origem
maxbiat	Máximo de intervalo entre chegada de pacotes do destino para a origem
stdbiat	Desvio padrão de intervalo entre chegada de pacotes do destino para a origem
minactive	Mínimo de tempo que o fluxo ficou ativo
meanactive	Média de tempo que o fluxo ficou ativo
maxactive	Máximo de tempo que o fluxo ficou ativo
fpshcnt	Número de vezes que a flag PSH foi enviada da origem para o destino
bpshcnt	Número de vezes que a flag PSH foi enviada do destino para a origem
totalfhlen	Bytes utilizados por headers da origem para o destino
totalbhlen	Bytes utilizados por headers do destino para a origem

V. Experimentos

Todos os experimentos feitos para este trabalho foram efetuados em um notebook Lenovo G40-80, processador Intel Core i5-5200U de 2.20 GHz, 8 GB de memória RAM, em um sistema operacional Ubuntu 16.04.

Na primeira abordagem, para obter as combinações de características do conjunto, foi utilizada a biblioteca *itertools*, sendo definido um número mínimo de características por conjunto de 4. O resultado gerado é uma lista contendo todas as combinações, neste caso contendo 1816 possibilidades. Depois, todas as combinações foram processadas por todos os estimadores, e as acurácias dos modelos obtidas.

Já para a segunda abordagem, o *Scikit-learn* implementa o RFE em conjunto com a técnica de Validação Cruzada, para todos os números possíveis de parâmetros. Dessa maneira, é possível identificar não só um número fixo de características, mas também encontrar qual o número ideal. Para utilizá-lo, passa-se como argumento uma instância do classificador desejado, e depois aplica-se a função *fit* com um conjunto de treino, para que ele encontre o subconjunto ótimo.

Depois de encontrado, é possível acessar tanto o estimador quanto as características do conjunto de maneira ranqueada. Todas as características escolhidas são atribuídas o valor 1, e dessa maneira é possível tanto definir quais as características ótimas quanto já executar predições com o estimador que a estrutura fornece.

VI. Resultados

Os resultados de acurácia obtidos pelos modelos executados nos experimentos são mostrados na Tabela III.

TABELA III
Acurácia dos modelos nas duas abordagens

Classificador	Acurácia Abordagem 1	Acurácia Abordagem 2
Árvores de Decisão	0.996604	0.994775
Florestas Aleatórias	0.997910	0.995559
AdaBoost	0.993208	0.996342
NB Multinomial	0.894723	0.950365
NB Bernoulli	0.869905	0.869905
SVC Linear	0.958986	0.996604

A Tabela IV apresenta os resultados referentes às frequências observadas das características mais investigadas em estudos anteriores, para ambas as abordagens. Já a Tabela V mostra as características adicionais da segunda abordagem que mais se fizeram frequentes nos experimentos.

Com base nos resultados, é possível afirmar que os classificadores testados foram bem-sucedidos em suas tarefas, obtendo altos níveis de acurácia. Dentre eles destacam-se os métodos baseados em Árvores, que obtiveram os melhores resultados, seguidos pelos métodos de Máquina de Vetores de Suporte, e por último pelos métodos *Naive Bayes*.

A inconsistência do *Naive Bayes* pode estar relacionada à própria suposição da técnica de que todas as características utilizadas são independentes entre si. Por outro lado, tratando-se da consistência das Florestas Aleatórias e do AdaBoost,

TABELA IV
Frequência de utilização das características

Característica	Frequência Abordagem 1	Frequência Abordagem 2
pctpacketspushed	6	5
iopr	4	4
totalbytes	4	1
avgiat	4	4
bpp	4	2
avgpayloadlength	4	3
totalpackets	3	1
pps	3	2
variat	3	2
bps	2	2
duration	2	1

TABELA V
Frequência de utilização das características da abordagem 2

Característica	Frequência
maxbiat	6
meanfiat	5
maxfiat	5
bpshcnt	4
minfiat	4
stdfiat	4
meanbiat	4
stdbiat	4

uma possível justificativa para o equilíbrio dessas técnicas pode estar relacionada ao fato de ambas utilizarem um comitê de estimadores para obter os seus resultados, e dessa maneira também se fazem mais consistentes do que as Árvores de Decisão por si só. Já o desempenho da técnica de Máquina de Vetores de Suporte pode estar ligado à relação linear observada nas características, tendo em vista que o *kernel* utilizado para os testes foi o linear.

Com relação às duas abordagens de utilização de características, é possível observar que a utilização do RFE para seleção de características impactou diretamente na melhoria da acurácia dos modelos probabilísticos de *Naive Bayes*, no entanto para os outros métodos os resultados foram parecidos. Porém, a utilização do RFE ainda apresenta menor custo computacional, o que a torna mais efetiva.

As características que foram utilizadas nos estudos anteriores se fizeram muito presente nas encontradas pelo RFE, com destaque para algumas que se destacaram em ambas as abordagens, como observado na Tabela IV.

Observa-se ainda que no que tange a abordagem do RFE, de maneira geral características ligadas a IAT (Tempo de Chegada entre Pacotes, em inglês *Inter-Arrival Time Between Packets*) foram amplamente usadas por quase todos os classificadores, enquanto as características de tamanho de fluxo, como total de *bytes*, não obtiveram muita representatividade em ambas as abordagens, assim como a característica duração do fluxo.

É possível observar também que o estudo obteve resultados comparáveis a outros da literatura recente, tanto para estudos como [12] que utilizam técnicas clássicas obtendo 98,8% de acurácia, quanto para [11] que obteve 98,6% de acurácia de detecção utilizando Redes Neurais Artificiais.

VII. Conclusão

Este trabalho apresentou um estudo sobre a utilização de técnicas da área de Aprendizado de Máquina para a detecção de *botnets* baseadas no protocolo IRC, através da análise de fluxos bidirecionais. Estudos deste tipo tornam-se cada vez mais relevantes nesta área tendo em vista o crescimento no uso de aparelhos conectados em rede que podem ser integrados às *botnets*, fator o qual contribui ainda mais para o poder de seus ataques.

A partir dos resultados obtidos, foi possível observar que as duas abordagens apresentadas no trabalho – assim como todos os métodos propostos – obtiveram bons resultados e, também, que determinadas características mostraram-se mais relevantes para a identificação dos fluxos maliciosos, obtendo altos níveis de acurácia.

Ademais, observou-se que o estudo obteve resultados relevantes em termos de acurácia dos modelos, frente a estudos recentes da área de detecção de *botnets*.

Como trabalhos futuros, têm-se a exploração de outras características possíveis de serem utilizadas na área, avaliar a eficiência dos métodos e características descritos para outros protocolos de *botnet* centralizadas como o HTTP ou descentralizadas que utilizam comunicação P2P, implementação de outras técnicas da área, como Redes Neurais Artificiais ou algoritmos de ML não-supervisionado para encontrar padrões na rede, e implementação de um sistema de monitoramento que analise os fluxos em tempo real.

Para fins de contribuição à comunidade científica, este trabalho encontra-se disponível em um repositório público do *GitHub* [35].

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Os autores também gostariam de agradecer à Fapesp 2017/22905-6 e ao CNPq 429003/2018-8.

Referências

[1] S. Miller and C. Busby-Earle, "The role of machine learning in botnet detection," in *Internet Technology and Secured Transactions (ICITST), 2016 11th International Conference for*. IEEE, 2016, pp. 359–364.

[2] G. Vormayr, T. Zseby, and J. Fabini, "Botnet communication patterns," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2768–2796, Fourthquarter 2017.

[3] W. Lu and A. A. Ghorbani, "Botnets detection based on irc-community," in *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*. Citeseer, 2008, pp. 1–5.

[4] C. Seifert, I. Welch, and P. Komisarczuk, "Taxonomy of honeypots," 2006.

[5] S. Asha, T. Harsha, and B. Soniya, "Analysis on botnet detection techniques," in *Research Advances in Integrated Navigation Systems (RAINS), International Conference on*. IEEE, 2016, pp. 1–4.

[6] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for iot security based on learning techniques," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2671–2701, thirdquarter 2019.

[7] G. Kambourakis, C. Kolias, and A. Stavrou, "The mirai botnet and the iot zombie armies," in *Military Communications Conference (MILCOM), MILCOM 2017-2017 IEEE*. IEEE, 2017, pp. 267–272.

[8] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in *2018 IEEE Security and Privacy Workshops (SPW)*, 2018, pp. 29–35.

[9] S. Khattak, N. R. Ramay, K. R. Khan, A. A. Syed, and S. A. Khayam, "A taxonomy of botnet behavior, detection, and defense," *IEEE Communications Surveys Tutorials*, vol. 16, no. 2, pp. 898–924, Second 2014.

[10] R. Bapat, A. Mandya, X. Liu, B. Abraham, D. E. Brown, H. Kang, and M. Veeraraghavan, "Identifying malicious botnet traffic using logistic regression," in *2018 Systems and Information Engineering Design Symposium (SIEDS)*, 2018, pp. 266–271.

[11] S. Chen, Y. Chen, and W. Tzeng, "Effective botnet detection through neural networks on convolutional features," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2018, pp. 372–378.

[12] Y. M. Mahardhika, A. Sudarsono, and A. R. Barakbah, "An implementation of botnet dataset to predict accuracy based on network flow model," in *2017 International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC)*, 2017, pp. 33–39.

[13] V. Kant, E. M. Singh, and N. Ojha, "An efficient flow based botnet classification using convolution neural network," in *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2017, pp. 941–946.

[14] K. Bakshi and K. Bakshi, "Considerations for artificial intelligence and machine learning: Approaches and use cases," in *2018 IEEE Aerospace Conference*. IEEE, 2018, pp. 1–9.

[15] E. Tyugu, "Artificial intelligence in cyber defense," in *Cyber Conflict (ICCC), 2011 3rd International Conference on*. IEEE, 2011, pp. 1–11.

[16] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[17] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[18] L. Jiang, H. Zhang, and Z. Cai, "A novel bayes model: Hidden naive bayes," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 10, pp. 1361–1371, Oct 2009.

[19] A. McCallum, K. Nigam *et al.*, "A comparison of event models for naive bayes text classification," in *AAAI-98 workshop on learning for text categorization*, vol. 752, no. 1. Citeseer, 1998, pp. 41–48.

[20] H. Zhang, "The optimality of naive bayes," vol. 1, no. 2, p. 3, 2004.

[21] V. Metsis and et al., "Spam filtering with naive bayes – which naive bayes?" in *Third Conference on Email and Anti-Spam (CEAS)*, 2006.

[22] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[23] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Machine Learning: ECML-98*, C. Nédellec and C. Rouveirol, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 137–142.

[24] A. C. Lorena and A. C. De Carvalho, "Evolutionary tuning of svm parameter values in multiclass problems," *Neurocomputing*, vol. 71, no. 16-18, pp. 3326–3334, 2008.

[25] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.

[26] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.

[27] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[28] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119 – 139, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002200009791504X

[29] R. Rojas, "Adaboost and the super bowl of classifiers a tutorial introduction to adaptive boosting," *Freie University, Berlin, Tech. Rep*, 2009.

[30] A. Müller and S. Guido, *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media, Incorporated, 2017. [Online]. Available: https://books.google.com.br/books?id=q5pnAQAACAAJ

[31] UNB, "Botnet dataset," http://www.unb.ca/cic/datasets/botnet.html, University of New Brunswick, 2018. [Online]. Available: http://www.unb.ca/cic/datasets/botnet.html

[32] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An overview of ip flow-based intrusion detection," *IEEE Communications Surveys Tutorials*, vol. 12, no. 3, pp. 343–356, Third 2010.

[33] D. Arndt, "Flowtbag," https://github.com/DanielArndt/flowtbag, 2011.

[34] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, "Usilng machine learning techniques to identify botnet traffic," in *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*. IEEE, 2006, pp. 967–974.
[35] L. F. B. Silva, "Botnet detection," https://github.com/luisfbueno/BotnetDetectionTCC, 2018.

Luis F. B. Silva Possui graduação em Bacharelado em Ciência da Computação (2018) pela Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP Bauru). Possui experiência na área de Ciência da Computação, nas áreas de Otimização Linear, Aprendizado de Máquina, e Segurança de Redes de Computadores.

Luan N. Utimura Possui graduação em Bacharelado em Ciência da Computação (2017) pela Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP Bauru) e Mestrando em Ciência da Computação (2018) pelo Instituto de Biociências, Letras e Ciências Exatas (IBILCE/UNESP São José do Rio Preto). Atualmente suas pesquisas se concentram na área de Redes de Computadores, Segurança de Redes de Computadores e Sistemas de Detecção de Intrusão.

Kelton A. P. Costa Possui graduação em Análise de Sistemas pela Universidade Sagrado Coração - USC (2000), mestrado em Ciência da Computação pelo Centro Universitário Eurípides de Marília - UNIVEM (2004), doutorado pela Universidade de São Paulo - USP (2009), pós-doutorado em Redes de Computadores pelo Instituto de Computação da Universidade Estadual de Campinas - UNICAMP e pós-doutorado pelo Departamento de Computação da Universidade Estadual Paulista Júlio de Mesquita Filho - UNESP. Atualmente é professor da Faculdade de Tecnologia (FATEC-campus Bauru) dos cursos de Tecnologia em Banco de Dados e Tecnologia em Redes de Computadores e também professor do curso de Ciência da Computação e Sistemas de Informação da Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP-campus Bauru) e professor dos cursos de Sistemas de Informação e Análise e Desenvolvimento de Sistemas da Instituição Toledo de Ensino (ITE). É avaliador de cursos de graduação do INEP/MEC, Docente do Programa de Mestrado em Ciência da Computação da UNESP (São José do Rio Preto/Bauru) e possui experiência na área de Ciência da Computação, com ênfase em Arquitetura de Sistemas de Computação e Sistemas Distribuídos, atuando principalmente nos seguintes temas: Gerência em Redes de Computadores, Segurança em Computadores, Sistemas de Detecção de Anomalias e Assinaturas em Redes de Computadores e Análise de Fluxo de Dados em Redes de Computadores.

Marcia A. Z. M. Silva Possui graduação em Licenciatura em Ciências - Habilitação Matemática pela Universidade Federal de São Carlos (1989), mestrado em Ciências da Computação e Matemática Computacional pela Universidade de São Paulo - USP/São Carlos (1994) e doutorado em Agronomia (Energia na Agricultura) pela Universidade Estadual Paulista "Júlio de Mesquita Filho" - UNESP/Botucatu (2002). Atualmente é professor assistente doutor - efetivo - da Universidade Estadual Paulista Júlio de Mesquita Filho - UNESP/Bauru. Tem experiência na área de Ciência da Computação, com ênfase em Otimização, atuando com programação linear e não linear.

Simone G. D. Prado Possui graduação em Bacharelado em Ciência da Computação pela Universidade Estadual Paulista "Júlio de Mesquita Filho" (1990) - Campus de São José do Rio Preto, mestrado em Engenharia Elétrica pela Universidade de São Paulo (1993) - São Carlos, doutorado em Engenharia Elétrica pela Universidade de São Paulo (2005). Desenvolveu pesquisa de pós-doutorado (2014/2015) na Faculdade de Engenharia da Universidade de Porto-Portugal. Desde 1994 é docente no Departamento de Computação da Faculdade de Ciências da Universidade Estadual Paulista Júlio de Mesquita Filho (UNESP) em Bauru-SP. Tem experiência na área de Ciência da Computação, com ênfase em Ontologia, atuando principalmente nos seguintes temas: Informática, Ensino a Distância, Inteligência Artificial, Ontologias, Agentes Inteligentes e Web Semântica.