Ernane Rosa Martins
(Organizador)

Pesquisa Operacional e sua Atuação Multidisciplinar

Atena Editora
Ano 2019

Ernane Rosa Martins

(Organizador)

Pesquisa Operacional e sua Atuação Multidisciplinar

Atena Editora
2019

2019 *by Atena Editora*

Editora Executiva: Profª Drª Antonella Carvalho de Oliveira
Diagramação: Karine de Lima
Edição de Arte: Lorena Prestes
Revisão: Os Autores

Atena Editora
Ano 2019

Dados Internacionais de Catalogação na Publicação (CIP)
(eDOC BRASIL, Belo Horizonte/MG)

P474 Pesquisa operacional e sua atuação multidisciplinar [recurso eletrônico] / Organizador Ernane Rosa Martins. – Ponta Grossa, PR: Atena Editora, 2019.

Formato: PDF
Requisitos de sistema: Adobe Acrobat Reader
Modo de acesso: World Wide Web
Inclui bibliografia
ISBN 978-85-7247-478-8
DOI 10.22533/at.ed.788191107

1. Pesquisa operacional. I. Martins, Ernane Rosa.
CDD 658.51

Elaborado por Maurício Amormino Júnior – CRB6/2422

Atena Editora
Ponta Grossa – Paraná - Brasil
www.atenaeditora.com.br
contato@atenaeditora.com.br

Atena Editora
Ano 2019

APRESENTAÇÃO

A Pesquisa Operacional (PO) utiliza a matemática, a estatística e a computação para auxiliar na solução de problemas reais, com foco na tomada das melhores decisões nas mais diversas áreas científicas e de atuação humana, buscando otimizar e melhorar suas performances. Através do uso de técnicas de modelagem matemática e eficientes algoritmos computacionais, a PO vem cada vez mais atuando na análise dos mais variados aspectos e situações de problemas complexos em demandas de inúmeras áreas, principalmente por conta de sua flexibilidade de aplicação e interação multidisciplinar, permitindo a tomada de decisões efetivas e a construção de sistemas mais produtivos.

Esta obra reúne importantes trabalhos que envolvem o uso de PO, realizados em diversas instituições de ensino do Brasil, abordando assuntos atuais e relevantes, tais como: modelos matemáticos; otimização multiobjectivo; heurísticas; algoritmos; otimização geométrica; metodologia SODA; soft systems methodology; strategic choice approach; procedimentos metodológicos de análise estatística; jogos cooperativos; algoritmos genéticos; método VIKOR; regressão linear múltipla; algoritmos de aprendizado de máquina; análise de decisão multicritério e composição probabilística de preferências.

A importância desta coletânea está na excelência dos trabalhos apresentados e na contribuição dos seus autores em temos de experiências e vivências. A socialização destes estudos no meio acadêmico, permite ampla análise e inúmeras discussões sobre diversos assuntos pertinentes referentes a atuação multidisciplinar da PO. Por fim, agradeço a todos que contribuíram na construção desta belíssima obra e desejo a todos os leitores, boas reflexões sobre os assuntos abordados.

Ernane Rosa Martins

SUMÁRIO

CAPÍTULO 11

AVALIAÇÃO DE MODELOS COMPUTACIONAIS DE APRENDIZADO DE MÁQUINA PARA DETECÇÃO REATIVA E PREVENTIVA DE BOTNETS

Vinicius Oliveira de Souza
Universidade Federal do Estado do Rio de Janeiro (UNIRIO), Programa de Pós-Graduação de Informática
Rio de Janeiro – RJ
vinicius.souza@uniriotec.br

Sidney Cunha de Lucena
Universidade Federal do Estado do Rio de Janeiro (UNIRIO), Programa de Pós-Graduação de Informática
Rio de Janeiro – RJ
sidney@uniriotec.br

Em: L Simpósio Brasileiro de Pesquisa Operacional, 2018, Rio de Janeiro.
Anais do SBPO - ISSN 1518-1731

RESUMO: É alarmante o aumento do número de ataques cibernéticos nos últimos anos. Para realizar essas atividades, geralmente são usadas *botnets*, que são redes de máquinas infectadas, controladas remotamente. Na detecção de *botnets* existem abordagens reativa e preventiva, sendo a primeira mais empregada. Esta abordagem implica em maiores riscos, uma vez que o ataque precisou ter início para ser detectado. O objetivo deste trabalho é avaliar modelos de aprendizado de máquina supervisionado e não supervisionado para a detecção de *botnets*, tanto na forma reativa quanto na preventiva, através do tráfego de controle que comanda uma *botnet*. Para tal, foram extraídos os atributos que melhor representam a atividade de rede, de maneira a alimentar uma seleção de modelos de aprendizado de máquina. A partir da análise dos resultados, identificou-se as características do tráfego de rede e os algoritmos com melhor desempenho nos cenários experimentados, comparando-se a eficácia na detecção preventiva e reativa.

PALAVRAS-CHAVE: Botnet, Aprendizado de Máquina, DDoS.

ABSTRACT: It is alarming the increase in the number of cyber attacks in the last years. For such, botnets, which are networks of remotely controlled infected machines, are generally used. In the detection of these botnets, there are reactive and preventive approaches, where the first is mostly used. This approach entails greater risks, since the attack needs to start in order to be detected. The aim of this work is to evaluate supervised and unsupervised machine learning models for botnet detection both in reactive and preventive approches, by inspecting the control traffic that rules the botnet. For that, attributes that best represent network activity were extracted in order to feed a selection of machine learning models. From the

analysis of the results, we identified the features of network traffic and algorithms that better performed for the experimental scenarios, comparing the efficacy on preventive and reactive detections.
KEYWORDS: Botnet, Machine Learning, DDoS.

1 I INTRODUÇÃO

É indiscutível a crescente importância dos serviços que vêm sendo fornecidos pela Internet, aumentando a dependência do uso da rede. Em consequência, o tráfego de dados cresce de modo exponencial. Com esse incremento no uso, também houve crescimento no número de incidentes de segurança. Dentre as atividades maliciosas que geram esses incidentes, estão o envio de *e-mail* não autorizado (*spam*), o furto de informações, o *click fraud*, que é a fraude de pesquisas ou de outras atividades onde se é contabilizado o clique de *link*, além dos perigosos ataques distribuídos de negação de serviço (*DDoS*), que são ataques de negação de serviço (*DoS*) realizados a partir de origens distribuídas, o que significativamente aumenta sua criticidade e dificulta seu bloqueio. O objetivo dos ataques *DDoS* é interromper um serviço através do esgotamento de recursos da vítima, de forma que esta não consiga responder às solicitações legítimas. Nessas atividades maliciosas, costumam ser empregadas *botnets*, que são redes de máquinas infectadas (*Bots*) controladas por um ou mais atacantes, chamados *botmasters*.

O crescente número de dispositivos conectados à internet, e suas vulnerabilidades de *software*, criam um ambiente propício para a disseminação dessas redes maliciosas. No Brasil, em 2017 o número total de incidentes reportados foi 29% maior, totalizando 833.775, sendo que 220.188 são referentes a ataques de negação de serviço, muitos desses ataques realizados através de dispositivos de Internet das Coisas (IoT). Esse número foi quatro vezes maior que as notificações de DoS recebidas em 2016 [CERT, 2018]. O Brasil ainda é o país da América Latina que mais sofre ataque DDoS, com 54% dos casos citados [Arbor, 2017]. Estimativas mostram que, no ano de 2016, as empresas tiveram um prejuízo global de US$ 280 bilhões devido a crimes digitais. Esse número ainda pode aumentar, já que, nessa pesquisa, 15% das empresas participantes tinham sofrido ataques em 2015 e 21% em 2016 [Thornton, 2017].

Torna-se primordial, portanto, garantir segurança para que as redes operem normalmente e continuem a propiciar mais serviços ao público. Todavia, diversos problemas são enfrentados para se atingir este objetivo. Como exemplo, tem-se a evolução constante dos mecanismos das *botnets* ao longo do tempo, mudando sua arquitetura e protocolos utilizados, e também a dificuldade para caracterizar e diferenciar certos padrões de tráfego normal, como nos momentos de pico de acesso, de um padrão de tráfego de ataque. Há, portanto, semelhanças estatísticas entre certos tráfegos maliciosos e alguns padrões de tráfego legítimo, devido ao perfil de alguns serviços [Ficco e Rak, 2015]. Além disso, os programas antivírus nem sempre

são capazes de identificar e remover um *bot*, devido à sofisticação desses *malwares*. Tais fatores tornam a detecção de uma *botnet* uma tarefa árdua.

As *botnets* possuem um ciclo de vida, com fases definidas desde o momento em que a vítima é infectada até essa fazer parte da *botnet*, e assim executar as atividades maliciosas que lhe foram destinadas [Silva et al., 2013]. Estas fases são semelhantes, independente do tipo de *botnet*, e são indicativos de que a atividade maliciosa de uma *botnet*, tanto na fase de controle quanto na fase de ataque em si, possuem padrões comportamentais que podem ser sistematicamente aprendidos para fins de detecção. Em função disto, técnicas de aprendizado de máquina vêm sendo cada vez mais aplicadas nas pesquisas voltadas para a detecção de *botnets* [Beigi et al., 2014]. Através desta abordagem, é possível extrair, através de exemplos de padrões comportamentais, um modelo de classificação de dados capaz de reconhecer fluxos de pacotes que estão associados ao tráfego malicioso proveniente destes *malwares*.

O presente trabalho busca analisar uma seleção de modelos computacionais de aprendizado de máquina para a detecção de *botnets*. Através desta análise, objetiva-se verificar qual subconjunto de características do tráfego de rede que, associado a determinados algoritmos de aprendizado de máquina, possibilita detectar *botnets* de forma preventiva - neste caso, através do tráfego de C&C -, mas também de forma reativa, quando as atividades maliciosas dos *bots* passam a ser executadas. Para isso, foram utilizadas as técnicas de seleção de atributos conhecidas como ranqueamento, *CFS* e *Wrapper*. Estas técnicas são usadas com o propósito de reduzir o gasto computacional dos algoritmos de aprendizado de máquina, a partir de uma seleção de atributos que garanta um bom desempenho do sistema de detecção. Após os atributos selecionados, foram avaliados e comparados o desempenho dos seguintes modelos computacionais: Redes Neurais Artificiais do tipo *Perceptron* Multicamadas, Rede Bayesiana, Árvore de Decisão do tipo J48, *Random Forest* e *K-Means*. Estes modelos foram usados para a detecção de fluxos de pacotes decorrentes de atividades de *botnets* registradas em *datasets* representativos de três diferentes cenários de rede da *Czech Technical University*. Como resultado, verificou-se quais desses métodos são mais eficientes, tanto na detecção preventiva quanto na reativa.

A organização do texto prossegue com a Seção 2, que detalha melhor o problema. A Seção 3 apresenta os modelos e as técnicas utilizadas neste trabalho. A Seção 4 apresenta os resultados obtidos. A Seção 5 apresenta os trabalhos relacionados e, por fim, a Seção 6 traz as conclusões e trabalhos futuros.

2 | APRESENTAÇÃO DO PROBLEMA

A arquitetura de uma *botnet* envolve os seguintes componentes [Silva et al., 2013]: o *Bot*, que é o *malware* instalado na máquina da vítima, que permite a execução de ações maliciosas; o *Botmaster*, que é o indivíduo (*hacker*) que possui controle da *botnet*; o Centro, ou Canal, de Comando e Controle (C&C), que é o meio através do

qual os *botmasters* enviam comandos aos *bots*, podendo ser um servidor central, por exemplo utilizando o protocolo IRC, ou distribuído, fazendo uso de redes *peer-to-peer* (P2P); e a *Botnet* em si, que é a rede de *bots* conectadas ao Centro de Comando e Controle, aguardando ordens do *botmaster*.

Para a detecção de ataques e atividades maliciosas, geralmente originadas de uma *botnet*, existem os chamados Sistemas de Detecção de Intrusão (*Intrusion Detection Systems*, ou IDS). Os IDS são divididos conforme seu posicionamento na rede e de acordo com o método usado para a detecção. Os *host-based* IDS (HIDS) são individualmente posicionados nos servidores e tratam apenas o tráfego direcionado àquele servidor específico, enquanto que os *network-based* IDS (NIDS) são posicionados na rede e inspecionam todo o tráfego que passa pela rede em questão, quaisquer que sejam os destinos dos fluxos de dados. Quanto ao método de detecção, há os IDS baseados em assinaturas e os baseados em anomalia. Sistemas baseados em assinaturas utilizam padrões conhecidos dos campos de cabeçalho (*header*) e de conteúdo (*payload*) dos pacotes de dados transmitidos na rede, provenientes de fluxos característicos de ataques. No entanto, esses sistemas tornam-se ineficazes para o problema à medida que o tempo passa, pois as *botnets* evoluem constantemente. Sistemas baseados em detecção de anomalias mostram-se mais promissores, pois a detecção da anomalia caracteriza-se pelo processo de encontrar padrões em um conjunto de dados de maneira a permitir avaliar quando o comportamento do tráfego da rede foge ao normal ou ao esperado. Nesse sentido, a detecção baseada em anomalias tem sido o principal alvo de pesquisas voltadas para técnicas de detecção de *botnets* [Silva et al., 2013]. Para tal, diversas técnicas de mineração de dados e aprendizado de máquina vêm sendo pesquisadas e usadas.

Existem ainda duas outras abordagens no contexto da detecção de uma *botnet*: a reativa e a preventiva. A primeira é a mais empregada [Freiling et al., 2005], porém possui desvantagens, pois é necessário grande poder computacional para analisar uma grande quantidade de informações [Freiling et al., 2005] que não para de crescer. A outra desvantagem é quanto ao tempo para a detecção de uma ameaça. Até o momento da detecção, usuários já foram prejudicados, pois a atividade maliciosa já está em andamento. Sistemas atuais, como o *Security Information and Event Ma- nagement* (SIEM), detectam 85% das intrusões depois de semanas de ocorrência [Ponemon, 2015] e a reação às ameaças é lenta, em média 123 horas depois de ocorridas [Clay, 2015]. Já a abordagem preventiva possui diversas vantagens, pois, ao invés de tentar detectar o ataque, investiga-se a causa raiz para, então, desativar a *botnet* [Freiling et al., 2005]. Entretanto, detectar o tráfego de C&C é também um desafio, pois ele se assemelha ao tráfego normal e possui baixo volume. Por exemplo, podem não haver muitos *bots* na rede monitorada e, além disso, a comunicação pode ser criptografada.

3 I METODOLOGIA DE AVALIAÇÃO

Nesta seção são apresentadas as bases de dados usadas para a avaliação dos modelos, os mecanismos de seleção de atributos usados e os modelos analisados. Por fim, são apresentadas as métricas de desempenho usadas para avaliar estes modelos.

3.1 Base de dados

Encontrar, ou gerar, bases de dados (*datasets*) realistas que reflitam o problema em questão não é tarefa trivial. Essa base deve conter, além do tráfego não malicioso, *botnets* realizando atividades maliciosas e também se comunicando através do canal de comando e controle *C&C*. Esta base deve ainda ser capaz de refletir as proporções entre tráfego legítimo e tráfego malicioso que representam situações reais de uma *botnet* em ação, em geral caracterizadas por um desbalanceamento entre estes tráfegos. Muitas dessas bases se originam de *honeypots*, que são máquinas aparelhadas para se comportarem como potenciais vítimas e, assim, dedicadas a coletar tráfego malicioso. Nesses cenários, o tráfego legítimo (não malicioso) fica em menor quantidade e acaba por não reproduzir uma situação real.

Neste trabalho serão utilizados três cenários da base de dados CTU-13 da CTU (*Czech Technical University*) [Garcia et al., 2014]. Trata-se de uma base desbalanceada, conforme a realidade, composta de tráfego de *botnets* realizando diversas atividades maliciosas e também tráfego legítimo. A base é dividida em treze diferentes cenários, cada um com características diferentes. Como o objetivo deste trabalho é detectar diversas famílias de *botnets* em arquiteturas diversas, pré e pós-ataque, foram selecionados os cenários 11, 12 e 13, mostrados na Tabela 1.

Cenário	Protocolo de aplicação	Atividade	Pacotes	Malware (ou Bot)
11	IRC	DDoS	6.337.202	Rbot
12	P2P	Sincronização	13.212.268	NSIS.ay
13	HTTP	SPAM e PortScan	50.888.256	Virut

Tabela 1: Características dos cenários do *dataset* CTU-13 utilizados no trabalho

Essas bases são compostas por dados de fluxos do tipo NetFlow [Claise, 2004], que é um recurso frequentemente encontrado em roteadores e *switches* para sumarizar as informações do tráfego que passa por uma rede. Isso é feito a partir da descrição dos fluxos de pacotes de dados existentes na rede, onde um fluxo é definido como uma sequência de pacotes que possuem os mesmos valores para o endereço IP de origem, o endereço IP de destino, o número identificador do protocolo de aplicação de origem (número da porta de origem), o número da porta de destino e o número identificador do protocolo usado no transporte das informações (por exemplo, identificando ser TCP ou UDP). Geralmente considera-se que um fluxo se encerrou quando se detecta o fechamento de uma sessão TCP, para os fluxos que fazem uso deste protocolo, ou

quando há ausência de pacotes neste fluxo por mais de 30 segundos. Para cada fluxo IP caracterizado por sua respectiva tupla, são registradas diversas informações relevantes, como o número de pacotes contido no fluxo, o total de *bytes* enviados, o tempo de início e a duração do fluxo, dentre outras.

Dentro de cada cenário do *dataset* CTU-13, foram atribuídos um dos seguintes rótulos (*labels*) para cada fluxo: rótulo *Normal*, atribuído ao tráfego sabidamente não malicioso, cujos fluxos se originaram de equipamentos para os quais havia uma supervisão cuidadosa; o rótulo *Botnet*, sinalizando os fluxos oriundos dos *IPs* sabidamente infectados; e o rótulo *Background*, usado para todo o tráfego restante. Importante ressaltar que, em todos os cenários, a classe associada ao rótulo *Botnet* possui uma quantidade de fluxos muito menor que a das demais classes, o que dificulta a detecção. Desta forma, cada fluxo contido nestas bases possui os seguintes atributos: StartTime, Dur, Protocol, SrcAddr, Sport, Dir, DstAddr, Dport, State, sTos, dTos, TotPkts, TotBytes, SrcBytes e Label.

3.2 Seleção e extração de *características*

Num processo de aprendizado de máquina, o número de características que devem ser processadas para fins de classificação dos dados pode aumentar significativamente o tempo de processamento sem que necessariamente melhore a capacidade de detecção do modelo. Isto é algo que pode impactar severamente um sistema de detecção de ataques. Por isso, é importante usar técnicas que reduzam o número de características que serão processadas pelo modelos sem que, com isso, se degrade a capacidade de classificação.

Para esta fase, buscou-se pesquisas sobre seleção de características para detecção de *botnets*. Em [Beigi et al., 2014], apontou-se as seguintes características: duração do fluxo, média de bits/s, média do tamanho do pacote e razão do número de pacotes de entrada com os de saída. Já em [Silva et al., 2017], indicou-se como características: duração do fluxo, estado, total de bytes transmitidos, média do tamanho do pacote e média de bits/s. Em [Alejandre et al., 2017], selecionou-se: número de pacotes da origem ao destino, média do tamanho do pacote, tamanho do *payload*, total de bytes transmitidos, tamanho do primeiro pacote da conexão, número de pacotes nulos e desvio padrão do tamanho do *payload*.

Após essa busca na literatura, foi realizado um pré-processamento no *dataset*, para eliminar dados desnecessários e calcular novos atributos. Nesta etapa, fluxos com duração igual a zero foram eliminados. Também foram eliminados os atributos afetados pelas técnicas de evasão das *botnets*, que trocam essas informações periodicamente: *SrcAddr, Sport, DstAddr, Dport e Protocol*. A partir daí, novos atributos foram calculados em função das características recomendadas pelos trabalhos em [Beigi et al., 2014; Silva et al., 2017; Alejandre et al., 2017]: MedPktSize (total de bytes / total de pacotes), MedPktSecond (total de pacotes / duração) e MedBitsSecond (total de bytes * 8 / duração). Além disso, também foi realizada a padronização do atributo

label, reduzindo para apenas duas possibilidades, *Normal* e *Botnet*, onde os fluxos rotulados como *Background* foram associados ao rótulo *Normal*.

Após essa etapa, mesmo considerando a redução de características já conseguidas, foram novamente utilizados algoritmos de seleção de atributos para se chegar às melhores características. Foram empregadas as técnicas *Ranker*, *Wrapper* e *CFS*, explicadas a seguir.

O algoritmo *Ranker* faz um ranqueamento dos atributos de acordo com a relevância, avaliando o ganho de informação em relação à classe. O *Wrapper* [Kohavi e John, 1997] usa estimativas oriundas de outros algoritmos de aprendizado de máquina, aplicando o algoritmo escolhido em subconjuntos dos atributos até encontrar as melhores características, dada uma medida de desempenho. Já o algoritmo CFS, *Correlation-based Feature Subset Selection* [Hall, 1999], avalia o valor de um subconjunto de atributos considerando a capacidade preditiva individual de cada recurso juntamente com o grau de redundância entre eles. Daí, são escolhidos os subconjuntos de recursos que estejam altamente correlacionados com a classe, tendo baixa intercorrelação entre eles.

Neste trabalho, primeiramente utilizou-se o *Wrapper* com o método de busca *Greedy*, heurística bastante popular [Beigi et al., 2014], e aplicando *Forward Selection*. Nesta abordagem, o processo inicia com um atributo e, então, são adicionados mais atributos a cada iteração, e ainda usando, para fins de verificação dos efeitos de classificação, dois diferentes algoritmos: Árvore de Decisão J48 e Rede Bayesiana. Ainda na abordagem *Wrapper*, foi usado também o método de busca adotando o algoritmo genético descrito em [Goldberg, 1989], utilizado no trabalho de [Alejandre et al., 2017], cuja verificação dos efeitos de classificação se baseou no algoritmo Árvore de Decisão J48.

Analisando os resultados da aplicação desses algoritmos nos três cenários selecionados, as características que mais se destacaram foram: *SrcBytes*, *State*, *MedPktSize*, *TotBytes* e *MedBitsSecond*. Esse subconjunto apresentou péssimo desempenho em alguns algoritmos, então foi removida empiricamente a característica *State*, uma vez que a mesma não fornece uma relação direta com fluxos IPs que possam estar ou não ligados a tráfego malicioso. Isto levou a um aumento significativo no desempenho da detecção. Comparando com os trabalhos de [Alejandre et al., 2017], [Silva et al., 2017] e [Beigi et al., 2014], ratifica-se a característica *MedPktSize* que aparece em todos. *TotBytes* só não aparece em [Beigi et al., 2014] e *MedBitsSecond* não foi selecionada no trabalho de [Alejandre et al., 2017]. Já o atributo *SrcBytes* aparece, no presente trabalho, como uma nova característica relevante para detectar tráfego de *botnet*.

3.3 Classificação

Após ter as características extraídas, é o momento de utilizá-las nos algoritmos de aprendizado de máquina para que sejam analisados os desempenhos desses na detecção de *botnets*. Nessa etapa foi utilizado o software *Weka* [Azuaje, 2006] para executar os algoritmos. O *Weka* possui código aberto e tem várias heurísticas para mineração de dados, como classificação, regressão, clusterização, regras de associação e visualização.

Os algoritmos de aprendizado de máquina usados neste trabalho fazem uso de classificação supervisionada e não supervisionada. Os algoritmos supervisionados fazem previsão com base em exemplos rotulados utilizados para treinamento e, após achado os padrões, ele usará o modelo para fazer previsões em dados de teste não rotulados. O aprendizado supervisionado pode ser: classificação, que é o utilizado neste trabalho, e regressão, onde a saída gerada é em valores contínuos por exemplo, para prever o preço de uma casa ou a inflação do próximo ano. O aprendizado não supervisionado utiliza um conjunto sem rótulos, onde o objetivo é organizar e classificar os dados seguindo alguma estrutura intrínseca de similaridade - por exemplo, agrupando-os em *clusters*.

O algoritmo de Rede Neural assemelha-se ao funcionamento do cérebro humano, onde cada neurônio é responsável por parte do processamento e o resultado deste processamento é passado ao próximo neurônio, até chegar a uma saída onde é obtido um grau de pertinência para cada classe - a classe de maior grau será a escolhida. Os vetores de peso são calculados durante o treinamento, vetores esses que são as ligações entre os neurônios. Após obtido o espaço amostral de entradas e saídas desejadas, minimiza-se o erro da estimativa de cada parâmetro através de um algoritmo conhecido como *Back Propagation* [Haykin, 2007]. A equação 1 mostra o cálculo realizado para ajustar os pesos sinápticos, onde e é o erro, ou seja, a diferença entre o sinal desejado e a saída da rede.

$$e_j(n) = d_j(n) - y_j(n) \qquad (1)$$

Para determinar a classe, o cálculo realizado por cada camada é determinado pelas seguintes equações:

$$u(i+1) = w(i)a(i) \qquad (2)$$

$$a(i+1) = \sigma(u(i+1)) \qquad (3)$$

$$\sigma(u) = \frac{1}{1+exp^{-\mu\beta}} \qquad (4)$$

Na Equação 2, $a(i)$ é a saída correspondente à camada i e $w(i)$ é o vetor de pesos

da camada *i* à *i* + 1. Na Equação 3, *a*(*i*+1) é a saída da camada seguinte e, na Equação 4, σ(*u*) é a função unipolar *sigmoid* utilizada, onde β é a constante de inclinação e não depende dos valores de entrada. Para valores de *u* maiores que zero, σ(*u*) vale 1, caso contrário ela vale -1. A Figura 1 mostra a rede neural utilizada.

Rede Bayesiana é um modelo descrito por meio de um grafo acíclico direcionado onde são representadas as relações de causalidade entre as variáveis. Os nós representam as variáveis e os arcos as conexões entre elas, e para representar as dependências são utilizadas probabilidades. Pode-se dizer que é uma representação enxuta de uma tabela de conjunção de probabilidades do universo do problema. As Redes Bayesianas utilizam conhecimento incerto e incompleto através do Teorema de Bayes [Pearl, 2011]. Algumas vantagens dessa abordagem são: (i) a possibilidade de

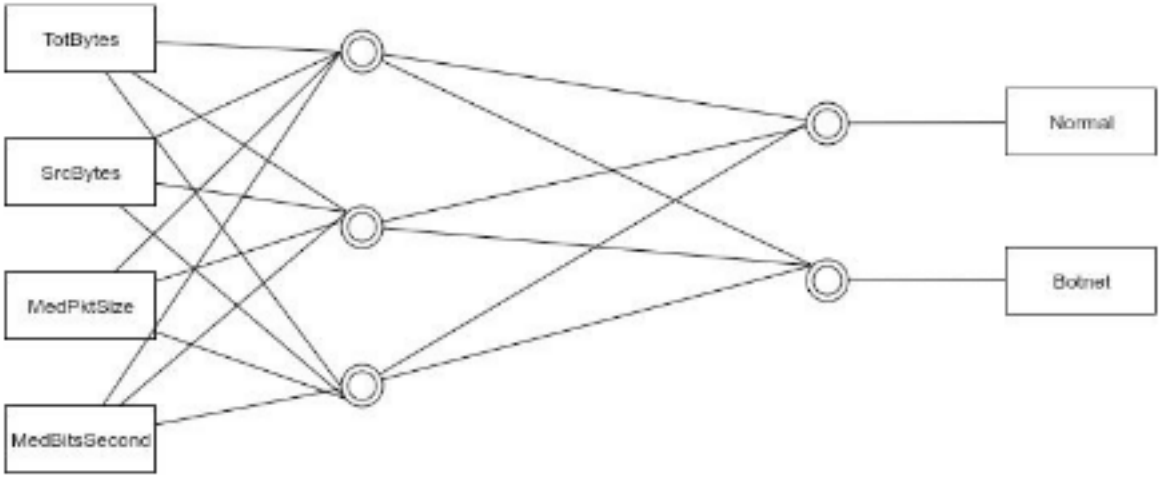

Figura 1: rede neural utilizada

classificar mesmo sem ter o valor de todos os atributos; e (ii) a visão do relacionamento entre todos os atributos envolvidos no problema, assim como independências, dependências e quão forte são esses relacionamentos. Para se achar as probabilidades condicionais, foi usado o algoritmo *SimpleEstimator*, que as estima quando a estrutura é aprendida. Como método de busca, foi usado o K2, que aprende a rede através da técnica *Hill Climbing*. A Figura 2 mostra a rede *bayesiana* utilizada.

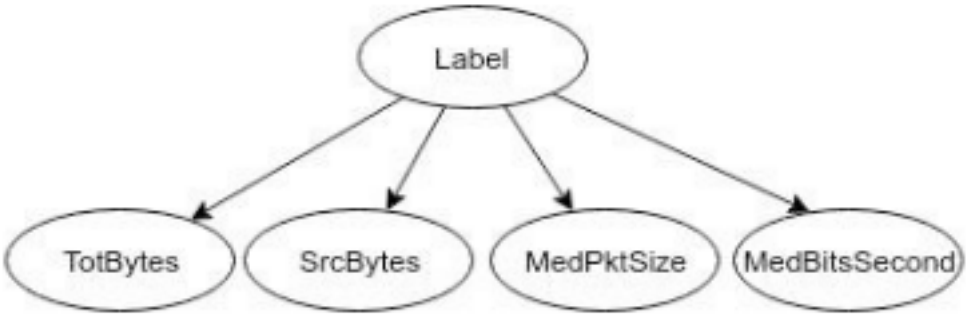

Figura 2: rede bayesiana utilizada

Árvore de Decisão é um dos principais algoritmos de Aprendizado de Máquina [Hall et al., 2011]. É composta por três elementos: o nó, que corresponde a um elemento de decisão, onde o nó raiz, que corresponde à decisão inicial, é geralmente o atributo mais discriminante entre as classes; arestas, que são os possíveis valores resultantes das decisões; e nó folha, que é a resposta, ou seja, a classe na qual os objetos serão classificados. Esse algoritmo possui duas fases: treinamento, onde é construída a árvore com base nos dados rotulados, e classificação, onde os dados dos atributos selecionados são testados a partir da raiz até um dos nós folhas, onde a classe será atribuída. Existem diversos algoritmos que implementam Á rvores de Decisão, como ID3, C4.5 e J48. Esses algoritmos utilizam a abordagem *top-down* para construir a árvore e são recursivos, só terminando quando constroem a menor árvore com maior acurácia. Para este trabalho, será utilizado o J48, que usa uma abordagem "gulosa" para construir a árvore. Já o *Random Forest* é um algoritmo do tipo *ensemble learning*, que cria diversos classificadores e depois agrupa os resultados. Nesse caso, são geradas diversas árvores de decisão que são usadas em conjunto para a classificação [Breiman, 2001].

O *K-means* é um algoritmo de agrupamento dos mais utilizados [Hall et al., 2011] em diversas áreas do conhecimento. Pertence à categoria de aprendizado não-supervisionado, ou seja, trabalha com instâncias não rotuladas com sua classe real. Essas instâncias são dividas em grupos (*clusters*) bem definidos, mais ou menos homogêneos, de acordo com a similaridade entre elas, o que gera um particionamento dessas instâncias. O particionamento mais adequado é aquele que agrupa instâncias semelhantes no mesmo *cluster* e separa as não semelhantes em diferentes *clusters*. O *K-means* busca encontrar a melhor partição do conjunto de instâncias em *k* grupos, onde cada grupo está associado a um centroide. Assim, o parâmetro *k* determina o número de *clusters* nos quais as instâncias serão divididas. Neste trabalho foi utilizado o valor de *k* como sendo igual ao número de classes (duas), de maneira que cada *cluster* pode ser associado a uma classe.

A Tabela 2 mostra os principais parâmetros utilizados nos algoritmos.

Parâmetros dos Algoritmos				
RNA	**Rede Bayesiana**	**J48**	**Random Forest**	**K-Means**
Taxa de aprendizagem = 0.3 Momentum = 0.2 Epochs = 500 Treshold = 20	Simple Estimator, α = 0.5 K2, Número máx. de pais = 1	Fator de confiança = 0.25	*Bag size* = 100% Iterações = 100	K=2, Função de Distância Euclidiana Número máx. de iterações = 500

Tabela 2: Principais parâmetros dos algoritmos de classificação

A partir dos algoritmos citados, pretende-se aqui comparar diversos paradigmas de aprendizado de máquina com o objetivo de verificar qual apresenta melhor

desempenho com relação ao problema. Serão comparados os seguintes paradigmas do aprendizado supervisionado [Monard e Baranauskas, 2003]: o conexionista (Redes Neurais Artificiais Multilayer do tipo Perceptron), o estatístico (Rede Bayesiana) e o simbólico (Árvore de Decisão J48 e *Random Forest*). Já no aprendizado não supervisionado, será avaliado o paradigma de *clustering* através do *KMeans*. Além disso, o algoritmo *Random Forest* também representa o conceito de *ensemble learning*.

3.4 Avaliação

Sabendo que a base é desbalanceada e que algoritmos classificadores de aprendizado de máquina são sensíveis a isso, os modelos gerados devem ser bem avaliados para evitar uma classificação tendenciosa, pois a classe rara pode ser ignorada pelo algoritmo classificador. Um método bastante utilizado para particionar o *dataset* é a divisão da base de forma aleatória, geralmente 2/3 para treinamento e 1/3 para teste, porém isso pode gerar uma classificação otimista. Para esse tipo de base, que é desbalanceada, a validação recomendada é a cruzada. Essa validação divide a base em diversas partes, mantendo uma parte para teste e as outras para treinamento, repetindo este procedimento para todas as partes. A acurácia final será então a média de todas as partes. Foi aqui utilizada a avaliação *10-foldcross-validation*, onde 9/10 da base é utilizado para treinamento e 1/10 para avaliação.

A métrica mais utilizada para avaliação é a acurácia, porém esta não é recomendada para bases desbalanceadas Por exemplo, pode ser conseguida uma acurácia de mais de 90% apenas classificando toda a amostra como sendo da classe majoritária, o que condenaria totalmente o classificador. Diante disso, existem outras métricas adequadas para este cenário: precisão, *recall*, *f-measure* e a área sob a curva (ROC). Essas métricas, juntamente com a acurácia, serão utilizadas neste trabalho, onde a identificação da classe *botnet* será considerada como “positivo” (ou seja, houve uma detecção de ataque) e a identificação da classe *normal* como “negativo”. As descrições das métricas são então as seguintes:

- Precisão - porcentagem de verdadeiros positivos (TP) sobre todas as instâncias classificadas como positivas, sejam de fato verdadeiras (TP) ou falsos positivos (FP):

$$\text{Precisão} = \frac{TP}{TP+FP} \qquad (5)$$

- Acurácia - porcentagem de tudo o que foi classificado corretamente, ou seja, verdadeiros positivos e verdadeiros negativos (TP e TN):

$$Acurácia = \frac{TP+TN}{TP+TN+FP+FN} \qquad (6)$$

- *Recall* - o número de verdadeiros positivos (TP) sobre o que é realmente positivo, ou seja, verdadeiros positivos e falsos negativos (TP e FN), o que equivale à taxa de verdadeiros positivos (TPR):

$$Recall = \frac{TP}{TP+FN} \qquad (7)$$

- *F-measure* - média harmônica entre Precisão e *Recall*:

$$F - Measure = \frac{2}{\frac{1}{Prec}+\frac{1}{Recall}} \qquad (8)$$

- Curva ROC (*Receiver Operating Characteristic*) - curva que relaciona Recall (TPR) e taxa de falsos positivos (FPR), sendo que, para reduzir essa curva a um valor escalar,é calculada a área abaixo dela.

4 I RESULTADOS

Os experimentos foram realizados em uma máquina virtual Ubuntu 16.04 LTS com 4 CPUs e 24 GB de memória. A Figura 3 mostra os resultados das métricas de avaliação para cada cenário e para cada algoritmo usado.

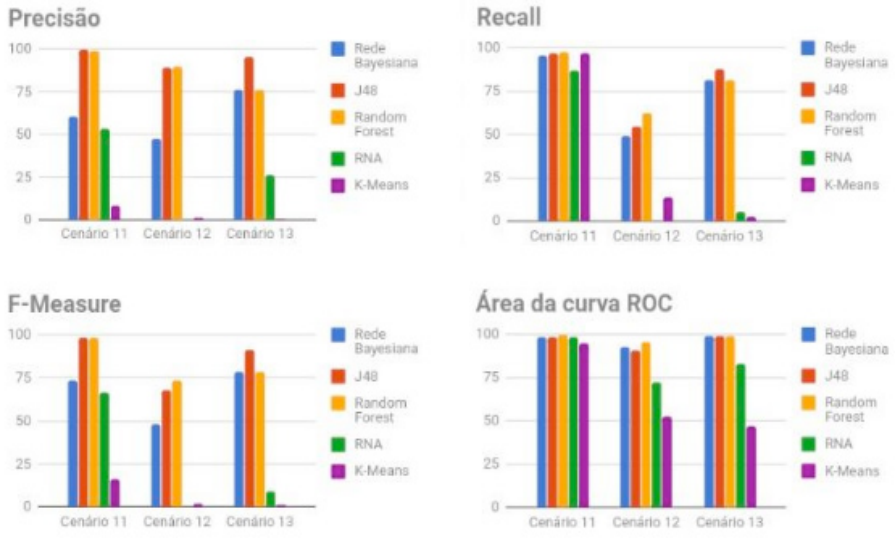

Figura 3: Resultados dos algoritmos de classificação

Ao analisar esses resultados, verificou-se que o paradigma “Simbólico” foi o que mais se destacou, sendo representado pelos algoritmos J48 e *Random Forest*. Todavia, ressalta-se a importância de se analisar as métricas em conjunto. Esses algoritmos obtiveram uma alta *Precisão*, porém, ao se analisar o cenário 12, verifica-se um baixo *Recall*. Isso se dá devido a estes algoritmos atingirem um número de

falsos positivos baixo, mas um número de falsos negativos alto nesse cenário. Esse baixo *Recall* no cenário 12 aconteceu para todos os algoritmos, e o desempenho baixo do *F-Measure* no cenário 12 dá-se pelo mesmo motivo, já que essa métrica é uma média harmônica entre o *Recall* e a *Precisão*. Vale mencionar que o cenário 12 é justamente aquele que representa uma abordagem preventiva de detecção. Ou seja, o que está sendo analisado no cenário 12 é o desempenho na detecção do tráfego de comando e controle, fase anterior à deflagração do ataque em si. Com este fato, conclui-se que as *botnets* com arquitetura descentralizada (P2P) realizando apenas sincronização são mais difíceis de serem detectadas pelos algoritmos deste trabalho. Por fim, verifica-se que os algoritmo Rede Neural (RNA) e *K-Means* apresentaram desempenho muito aquém dos demais. Apenas na métrica *Recall*, no cenário 11, conseguiram um bom desempenho, devido ao elevado número de verdadeiros positivos que esses algoritmos alcançaram nesse cenário. Porém, ainda nesse cenário, a métrica *F-Measure* dos dois algoritmos foi baixa devido ao elevado número de falsos positivos. Ou seja, por mais que tenham acertado muitos fluxos de *botnets*, também classificaram erroneamente muitos fluxos legítimos como *botnets*, indicando que a classificação realizada somente com aprendizado não supervisionado, ou com o paradigma conexionista, não é recomendada para esses cenários de *botnets*.

5 I TRABALHOS RELACIONADOS

O trabalho em [Fedynyshyn et al., 2011] propõe uma solução baseada em HIDS, identificando diferentes canais de C&C sem verificar o *payload* do pacote, algo que abordagens anteriores não conseguiam. Nele são avaliados os algoritmos de aprendizado de máquina J48 e *Random Forest* aplicados a um *dataset* próprio, contendo tráfego legítimo e diversas famílias de *bots*. Foram validadas as hipóteses de que o tráfego de C&C da *botnet* pode ser diferenciado de outros, incluindo do tráfego legítimo, e de que as características de diferentes estilos de C&C são semelhantes em diferentes famílias *botnet*.

Em [Zand et al., 2014] é apresentada uma abordagem para extrair assinaturas C&C usando um *dataset* produzido com o *bot Anubis*, para tal examinando o *payload* dos pacotes de dados transmitidos na rede. Primeiro são extraídas as “sequências” mais frequentes no tráfego na rede, depois é utilizada uma função de ranqueamento que dará uma pontuação maior às “sequências” que aparecem mais frequentemente em uma classe de conexões e raramente em outras. Essa abordagem é motivada pelo pressuposto de que conexões C&C de uma família de *malware* compartilham similaridades, enquanto conexões que não sejam de C&C apresentam maior diversidade de características entre si. Com isso, a proposta dos autores obteve um desempenho superior a de outros trabalhos relacionados.

Em [Beigi et al., 2014] são comparados e avaliados atributos associados ao conceito de “fluxo de pacotes na rede”, empregados nos estudos de detecção de

botnets existentes. A abordagem utilizada contou com o método *Wrapper* para a seleção de atributos e com o algoritmo de classificação de árvore de decisão C4.5. Para os experimentos, foi criado um *dataset* contendo um conjunto diversificado de *botnets* misturado com amostras de tráfego normal da rede, muitas vezes chamado de tráfego de *background*.

Em [Alejandre et al., 2017] é proposta uma metodologia para a seleção de atributos objetivando detectar *botnets* na fase de comando e controle. Foi utilizado um algoritmo genético para selecionar o conjunto de atributos que fornece a maior taxa de detecção ao se usar o algoritmo C4.5 para a classificação dos dados. Experimentos foram realizados no sentido de extrair os melhores atributos para cada *botnet* analisada e para cada tipo de *botnet* em geral.

Dentre os trabalhos aqui relacionados, [Zand et al., 2014] realiza uma análise do *payload* do pacote, o que hoje é inviável pois a comunicação via C&C geralmente é criptografada, esse trabalho é validado em apenas um tipo de *bot*, trazendo dúvidas sobre a validade dessa metodologia. Ainda sobre a avaliação de modelos para se detectar *botnets*, a comparação entre os modelos estudados nos trabalhos não é possível de ser realizada, pois não há uma padronização de *dataset* e de métricas de avaliação.

Em cima desse problema, foi realizada a pesquisa [Silva et al., 2017], que também utilizou o dataset CTU-13 para levantar atributos relevantes à detecção de *botnets* através das técnicas de seleção de atributos, além de analisar a eficiência dos algoritmos de aprendizado de máquina J48, SVM, *Naive Bayes* e k-NN para a detecção do tráfego de *botnets*. Porém, ainda faltam paradigmas do Aprendizado Supervisionado para serem comparados, e a própria pesquisa aponta como trabalhos futuros a avaliação de outros paradigmas como o conexionista. Além disso, o trabalho não abordou o aprendizado não supervisionado.

Buscando preencher essas lacunas, foi desenvolvido o presente trabalho em que diversos paradigmas do aprendizado de máquina supervisionado e não supervisionado são comparados, além de ser realizada uma seleção de características do tráfego de rede. Assim, foi possível analisar se existe um algoritmo que satisfaz os diversos cenários de *botnets* estudados ou se uma combinação dessas técnicas deve ser usada para uma classificação mais eficaz, determinando a combinação de características e técnicas de aprendizado de máquina ideais para o problema da detecção de *botnets*.

6 | CONCLUSÃO

O número de incidentes de segurança na Internet vem aumentando a cada dia, assim como a dependência nos serviços por meio dela oferecidos. Nesse contexto, as *botnets* surgem como fontes geradoras de diversas atividades maliciosas. Detectar essas redes de máquinas "zumbis" é tarefa desafiadora, devido às características de suas arquiteturas.

Diante disso, este trabalho propõe uma metodologia utilizando modelos de aprendizado de máquina capazes de detectar *botnets* de forma preventiva, através do tráfego de comando e controle, e reativa, a partir de tráfegos de atividades maliciosas. Tal se deu por meio de características extraídas dos fluxos de tráfego, sem a inspeção do conteúdo de dados (*payload*) dos pacotes. Para isso, foi realizado um processo que apontou uma nova característica relevante: o número de *bytes* transmitidos pela origem. Foi então analisado um conjunto de algoritmos de aprendizado de máquina supervisionado representando os paradigmas conexionista, estatístico e simbólico, além do aprendizado não supervisionado representado por um algoritmo de *clustering*, todos eles aplicados a cenários reais, contribuindo na busca por algoritmos e características do tráfego de rede que consigam detectar os diversos cenários de *botnets*. Os modelos que seguem o paradigma simbólico se destacaram, com os algoritmos J48 e *Random Forest* praticamente igualados em termos de desempenho, porém destaca-se um maior número de falsos negativos ao se realizar a detecção na forma preventiva.

Dessa forma, a partir dos experimentos realizados, as características do tráfego de rede que se destacaram (SrcBytes, *MedPktSize*, *TotBytes* e *MedBitsSecond*) em conjunto com os algoritmos de aprendizado de máquina do paradigma simbólico (J48 e *Random Forest*) são as mais indicadas para serem usadas em uma arquitetura de um IDS real para a detecção de *botnets*. Como trabalhos futuros, pretende-se expandir este estudo para outros cenários e analisar outros paradigmas de aprendizado de máquina, considerando-se também a criação de uma janela de tempo para agrupar as sequências de fluxo e extrair novas características que melhorem a detecção. Além disso, deseja-se implementar o melhor modelo encontrado em uma arquitetura real.

REFERENCIAS

Alejandre, F. V., Cortés, N. C., e Anaya, E. A. (2017). Feature selection to detect botnets using machine learning algorithms. In *Electronics, Communications and Computers (CONIELECOMP), 2017 International Conference on*, p. 1–7. IEEE.

Arbor, N. (2017). 12º relatorio de seguranca de infraestrutura mundial anual. http://br. arbornetworks. com/visibilidade-de-redes/. Acessado em 28/10/2017.

Azuaje, F. (2006). Witten ih, frank e: Data mining: Practical machine learning tools and techniques 2nd edition. *BioMedical Engineering OnLine*, 5(1):1.

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

Breiman, L. (2001). Random forests. Machine learning, 45(1):5–32.

CERT, B. (2018). Incidentes reportados ao cert.br, janeiro a dezembro de 2017. https://www.cert. br/stats/incidentes/2017-jan-dec/analise.html. Acessado em 20/03/2018.

Claise, B. (2004). Cisco systems netflow services export version 9.

Clay, P. (2015). A modern threat response framework. Network Security, 2015(4):5–10.

Fedynyshyn, G., Chuah, M. C., e Tan, G. (2011). Detection and classification of different botnet c&c channels. In International Conference on Autonomic and Trusted Computing, p. 228–242. Springer.

Ficco, M. e Rak, M. (2015). Stealthy denial of service strategy in cloud computing. IEEE transac- tions on cloud computing, 3(1):80–94.

Freiling, F. C., Holz, T., e Wicherski, G. (2005). Botnet tracking: Exploring a root-cause metho- dology to prevent distributed denial-of-service attacks. In *European Symposium on Research in Computer Security*, p. 319–335. Springer.

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

Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning, 1989. *Reading: Addison-Wesley*.

Hall, M., Witten, I., e Frank, E. (2011). Data mining: Practical machine learning tools and techni- ques. *Kaufmann, Burlington*.

Hall, M. A. (1999). *Correlation-based feature selection for machine learning*. PhD thesis, Univer- sity of Waikato Hamilton.

Haykin, S. (2007). *Redes neurais: princípios e prática*. Bookman Editora.

Kohavi, R. e John, G. H. (1997). Wrappers for feature subset selection. *Artificial intelligence*, 97 (1-2):273–324.

Monard, M. C. e Baranauskas, J. A. (2003). Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações*, 1(1):32.

Pearl, J. (2011). Bayesian networks. *Department of Statistics, UCLA*.

Ponemon, I. (2015). Ibm (2015). 2015 cost of data breach study: Global analysis.

Silva, D. C., Silva, S. S., e Salles, R. M. (2017). Metodologia de detecção de botnets utilizando aprendizado de máquina.

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

Thornton, G. (2017). The global impact of cyber crime. *Grant Thornton International Business Report*.

Zand, A., Vigna, G., Yan, X., e Kruegel, C. (2014). Extracting probable command and control signatures for detecting botnets. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, p. 1657–1662. ACM.

SOBRE O ORGANIZADOR

Ernane Rosa Martins - Doutorado em andamento em Ciência da Informação com ênfase em Sistemas, Tecnologias e Gestão da Informação, na Universidade Fernando Pessoa, em Porto/Portugal. Mestre em Engenharia de Produção e Sistemas, possui Pós-Graduação em Tecnologia em Gestão da Informação, Graduação em Ciência da Computação e Graduação em Sistemas de Informação. Professor de Informática no Instituto Federal de Educação, Ciência e Tecnologia de Goiás - IFG (Câmpus Luziânia), ministrando disciplinas nas áreas de Engenharia de Software, Desenvolvimento de Sistemas, Linguagens de Programação, Banco de Dados e Gestão em Tecnologia da Informação. Pesquisador do Núcleo de Inovação, Tecnologia e Educação (NITE), certificado pelo IFG no CNPq.

Agência Brasileira do ISBN
ISBN 978-85-7247-478-8