Intrusion Detection System Based On Flows Using Machine Learning Algorithms

E. M. Kakihata, H. M. Sapia, R. T. Oikawa, D. R. Pereira, J. P. Papa,V. H. C. Alburquerque and F. A. Silva

***Abstract*— The use of technology by different types of devices generates a large flow of confidential and personal information. Referencing this situation, it is necessary to use computer security tools, such as Intrusion Detection Systems (IDS). This work presents an IDS that can perform the flow-based analysis (netflow). The first step of this research conducted an analysis on flows previously collected and properly detected three different types of attacks. In the second step, the flows were organized to be used in machine learning algorithms.**

***Keywords*— intrusion detection system, netflow, machine learning.**

I. INTRODUÇÃO

A TECNOLOGIA vem se desenvolvendo constantemente, que por consequência aumenta a probabilidade de um sistema informatizado conter novas falhas e vulnerabilidades de segurança [1]. Referenciando essa situação, se faz necessário utilizar mecanismos operacionais de segurança, conhecidos como sistemas de detecção de intrusão (do inglês*, Intrusion Detection System* – IDS) e Sistemas de Prevenção de Intrusão (do inglês, *Intrusion Prevention Systems* - IPS). Estes mecanismos atrelados à rede e trabalhando em conjunto, podem garantir a prevenção e detecção de ataques cibernéticos ou intrusões [7]. Ataques cibernéticos são ações que visam comprometer os pilares da segurança da informação, como integridade, confidencialidade e disponibilidade de um recurso computacional, independente de obter sucesso ou não [13].

Segundo Lima [8], os métodos de detecção podem ser divididos em duas abordagens, são elas: (*i*) mau uso (assinatura), ou (*ii*) anomalia. O *IDS* baseado em assinaturas analisa o comportamento da rede em busca de padrões de ataques pré-definidos, onde para cada ataque, se tem uma assinatura. Já *IDS* baseado em anomalias verifica ocorrências de ações fora do padrão na rede/sistema, sendo que para cada ação normal existe um perfil criado, e quando um comportamento não corresponde a um desses perfis, possivelmente algum ataque esta sendo realizado [13].

A maioria dos *IDS* do mercado realiza a análise baseada em pacotes de rede, este trabalho propõe um *IDS* que realiza análise baseada em fluxo de rede. Um fluxo de rede é uma sequência unidirecional entre dois *hosts* na rede, sendo representado por um tráfego com características comuns [3].

Redes de médio/grande porte, intrínseco ao aumento da capacidade da largura de banda nos dia de hoje, geram um número elevado de pacotes, o que requer um custo computacional considerável para que seja realizada a sua análise. Uma análise baseada em fluxos é uma solução interessante, pois o fluxo representa o resumo dos pacotes trafegados na rede, o que faz com que o número de itens a ser verificados seja menor. A análise dos fluxos exige a *priori* a coleta e armazenamento dos mesmos, para *posteriori* análise. Para a execução dessa tarefa foi utilizada uma ferramenta em Java capaz de coletar e armazenar os fluxos da rede em um banco de dados *MySQL*.

O *IDS* proposto realiza a verificação do comportamento da rede por meio de consultas ao banco de dados ao qual estão armazenados os fluxos. Posteriormente os fluxos são comparados com as assinaturas de ataques criadas, tendo como escopo informar o gerente da rede sobre algum ataque inerente a determinado *host*.

Após a detecção dos três comportamentos maliciosos, todos os fluxos da base de dados foram organizados de modo a serem aplicados em métodos de aprendizado de máquina.

Métodos de aprendizado de máquina são técnicas computacionais capazes de realizar reconhecimento de padrões e classificação de dados. Foram utilizados neste trabalho Floresta de Caminhos Ótimos (do inglês, *Optimum-Path Forest* - OPF), Máquinas de Vetores de Suporte (do inglês, *Support Vector Machine* - SVM), K-Vizinhos Mais Próximos (do inglês, *K-Nearest Neighbor* - KNN) e Classificador Bayesiano (Bayes).

As principais contribuições deste trabalho podem ser elencadas em 3 partes, são elas: *(i)* implementação de um *IDS* baseado em assinaturas utilizando fluxos de redes; (*ii*) utilização de diferentes métodos de aprendizado de máquina para classificação de fluxos legítimos e ataques; (*iii*) realização de testes minuciosos de modo a estabelecer um compromisso entre acurácia e tempo de execução despendido na classificação; haja visto que o *IDS* requer alta assertividade no menor tempo de resposta.

O restante do trabalho está organizado da seguinte maneira.

E. M. Kakihata, Universidade do Oeste Paulista (Unoeste), Presidente Prudente, São Paulo, Brasil, eduardomassato@hotmail.com

H. M. Sapia, Universidade do Oeste Paulista (Unoeste), Presidente Prudente, São Paulo, Brasil, helton@unoeste.edu.br

R. T. Oikawa, Universidade do Oeste Paulista (Unoeste), Presidente Prudente, São Paulo, Brasil, oikawa@unoeste.br

D. R. Pereira, Universidade do Oeste Paulista (Unoeste), Presidente Prudente, São Paulo, Brasil, danilopereira@unoeste.br

J. P. Papa, Universidade Estadual Paulista (Unesp), Bauru, São Paulo, Brasil, papa@fc.unesp.br

V. H. C. de Albuquerque, Universidade de Fortaleza (Unifor), Fortaleza, Ceará, Brasil, victor.albuquerque@unifor.br

F. A. Silva, Universidade do Oeste Paulista (Unoeste), Presidente Prudente, São Paulo, Brasil, chico@unoeste.br

A Seção II apresenta alguns trabalhos relacionados. Na Seção III são descritos os conceitos relacionados com os tipos de comportamentos maliciosos. A Seção IV descreve brevemente os métodos de aprendizado de máquina utilizados nos testes. A metodologia empregada é apresentada a na Seção V e os experimentos e os resultados obtidos se encontram descritos na Seção VI. Por fim, a Seção VII apresenta as conclusões e propostas futuras.

II. TRABALHOS RELACIONADOS

Detecção de ataques e intrusões são temáticas amplamente pesquisadas e que seguem a tendência de aplicar métodos de aprendizado de máquina na detecção de intrusões. O trabalho de Lima [8] combina métodos de análise semântica e redes neurais artificiais para a detecção de comportamentos da rede.

De modo geral a complexidade das técnicas de inteligência artificial para o aprendizado de novos ataques é alta, que por consequência inviabiliza o seu treinamento em tempo real, Pereira [12] mostrou que o classificador de padrões OPF [11] é muito eficiente e rápido em termos de reconhecimento quando comparado com outras técnicas de aprendizado de máquina.

Rede de computadores de médio e grande porte geram um número significativo de pacotes trafegados e que devem ser analisados pelo *IDS*, sendo assim Corrêa [3] propôs um *IDS* que realiza a detecção de intrusão em redes de computadores utilizando o *NetFlow* versão 5. Esse método de detecção de eventos é realizado por meio de assinaturas que se constituem de um número de passos, essas assinaturas são usadas para detectar padrões que se repetem. O trabalho de Batista [1] propõe o desenvolvimento de um *IDS* que realiza a detecção de eventos em redes de computadores por meio de detecção de novidade junto com métodos de aprendizado não-supervisionados, onde foram utilizados k-médias, x-médias e redes neurais.

Este trabalho verifica a viabilidade dos métodos de aprendizado de máquina (OPF, SVM, KNN e Bayes) no contexto de detecção de comportamentos maliciosos em redes de computadores utilizando fluxos de rede (*netflow*).

Um fluxo de rede é uma sequência unidirecional de pacotes entre dois *hosts* sendo representado por um tráfego de características comuns. O fluxo pode ser entendido como uma tupla, que contém algumas informações como por exemplo: IP origem, IP destino, porta origem, porta destino, protocolo e etc. Todos os pacotes que possuem o mesmo valor nos campos descritos acima são contabilizados em um registro de fluxo [3].

Esses fluxos depois de serem contabilizados, exportados e armazenados tornam-se informações de grande valor, sendo utilizados para análises futuras e servindo como modelo de comportamento para outras redes em análise, auxiliando assim na segurança da rede.

III. COMPORTAMENTOS MALICIOSOS E SUAS ASSINATURAS

Algumas informações foram selecionadas para a criação de assinaturas dos comportamentos maliciosos. Posteriormente essas assinaturas foram usadas na identificação de padrões. Os campos utilizados foram escolhidos segundo sua importância no processo de identificação da ação (normal/maliciosa); são eles: *srcaddr* (endereço IP origem), *dstaddr* (endereço IP destino), *srcport* (porta origem), *dstport* (porta destino), *dPkts* (quantidade de pacotes no fluxo), *dOctets* (quantidade de *bytes* no fluxo), *first* (tempo em milissegundos da entrada do primeiro pacote no fluxo), *last* (tempo em milissegundos da entrada do ultimo pacote no fluxo) e *prot* (tipo de protocolo). A partir dos campos selecionados, foi possível aferir outras informações importantes como duração do fluxo (*last* – *first*) e média de *bytes* por pacote (*dOctets* / *dPkts*).

Este trabalho utiliza três assinaturas, duas foram baseadas em assinaturas apresentadas no trabalho de Corrêa [3], portanto foram adaptadas para o ambiente de pesquisa e a outra foi criada no decorrer do desenvolvimento deste trabalho.

A. Port Scanning

O *port scanning* é uma varredura de portas utilizada por usuários maliciosos para obtenção de informações de um ambiente de rede. Essa técnica é uma fase de reconhecimento da rede. Visa extrair informações para explorar as vulnerabilidades do alvo, que consequentemente facilitará o ataque posterior [10].

A assinatura do *port scanning* consiste em duas etapas: (*i*) seleção de endereços IP que acessaram diversas portas distintas em um mesmo IP destino; (*ii*) análise da quantidade de fluxos de acordo com a assinatura para cada *host* origem e destino apresentado etapa anterior. Caso os fluxos apresentem o campo *prot* igual a 6 (referente ao protocolo TCP), quantidade de *bytes* no fluxo entre 150 a 154, *bytes* por pacote entre 48 a 54, duração menor que 15 segundos, e ultrapasse a quantiade de 100 fluxos nesse padrão, é disparado um alarme de aviso comunicando que o *host* destino sofreu um ataque de *port scanning*.

B. Força Bruta

O ataque de força bruta é executado em *hosts* que possuem serviço SSH (*Secure Shell Protocol*) e que geralmente utiliza a porta 22 por *default* ou qualquer outro serviço de autenticação por *login* e senha. Seu objetivo é testar várias combinações de caracteres e numéricos e alfanuméricos até decifrar um *login* e senha válida [3].

A assinatura da força bruta também consiste de duas etapas, são elas: (*i*) verificação de *hosts* que realizaram um número considerável de tentativas de acesso a um mesmo *host* destino na porta 22; (*ii*) analisar se todo *host* origem e destino (apresentado da etapa anterior) expõe fluxos com o campo *prot* igual a 6, e *bytes* por pacote entre 85 a 105, e duração menor que 15 segundos e ultrapasse a quantidade de 15 fluxos, então o *host* destino é alertado que está ocorrendo um ataque de força bruta.

C. Negação de Serviço

O ataque de negação de serviço (do inglês, *Denial of Service – DoS*) é uma grande ameaça para qualquer sistema ou rede, causando danos irreversíveis e prejuízos de milhões de dólares por ano. Este ataque consiste em deixar algum serviço ou

recurso indisponível [9]. Existem diferentes meios de realizar este ataque, seja por meio de inundação de pacotes ICMP (do inglês, *Internet Control Message Protocol*), inundação de pacotes com FLAG SYN, entre outros. Neste trabalho, foi utilizado a técnica *DoS* por inundação de pacotes ICMP.

Esta técnica consiste em somente uma etapa, onde se verifica qual *host* enviou uma quantidade de pacotes ICMP (que é representado pelo campo *prot* com valor igual a 1) maior que 40.000 em um único fluxo e média de *bytes* por pacote igual a 28.

IV. ALGORITMOS DE APRENDIZADO DE MÁQUINA

A. Floresta de Caminhos Ótimos

O OPF é um classificador de padrões, que se baseia na teoria dos grafos. O OPF modela o problema de reconhecimento de padrões como um problema de floresta de caminhos ótimos em um grafo definido no espaço de atributos [11].

Cada classe é representada por uma ou mais árvores de caminhos ótimos em que suas raízes são amostras denominadas protótipos. Os nós são as amostras, que são representados pelos seus vetores de atributos (características), e os arcos são definidos de acordo com uma relação de adjacência pré-estabelecida. Os arcos são ponderados pela distância entre os vetores de características de seus nós, e diversas funções de custo podem ser empregadas para particionar o grafo em árvores de caminhos ótimos, e os mesmos são enraizadas pelos respectivos protótipos na fase de treinamento [11].

A partir dos protótipos é realizado o processo de conquista onde as amostras tentam encontrar o protótipo que lhe oferecer o melhor caminho (caminho ótimo). O rótulo da amostra a ser classificada é o mesmo do protótipo mais fortemente conexo a ela [11].

B. Máquinas de Vetores de Suporte

As SVM resolvem problemas de classificações de padrões pressupondo que é possível separar classes diferentes no espaço, admitindo que os dados não são linearmente separáveis, ou seja, as amostras podem ser separadas em grupos onde a superfície de decisão irá usar linhas e curvas para delimitar as regiões limites [14].

O escopo do SVM é pré-processar os dados e identificar uma função discriminante não linear, representando o problema no espaço de hiperplano. É feito o mapeamento dos dados que estão em uma dimensão qualquer para outra maior, o que faz os dados se tornarem linearmente separáveis. O mapeamento transforma o vetor de entrada em outro vetor maior usualmente, o que faz com que o novo conjunto de dados seja linearmente separável [12].

O hiperplano separa as amostras de modo que o maior número de amostras de uma mesma classe fique de um lado e maximiza a distância de cada classe pelo hiperplano. Essa distância é chamada de margem de separação. O hiperplano é determinado por um subconjunto das classes a serem classificadas, denominados de vetores de suporte.

C. K Vizinhos Mais Próximos

O KNN é um algoritmo não-paramétrico que usa todas as amostras de treinos como protótipos, enquanto que no OPF, são selecionadas as melhores amostras para serem os protótipos [11] e [5].

A amostra é representada por um vetor de característica e seu rótulo (denominado classe). Para realizar a classificação, o KNN utiliza um parâmetro de entrada *k*, que representa o número de vizinhos que contribuem para a classificação de uma amostra [5].

A classificação de uma amostra teste *t* é dada pelas amostras de treino (quantidade determinada por *k*) mais próximas e centrado em t, onde a classe que possui a maioria determina o rótulo (por exemplo: azul ou laranja) de *t*. Para situações onde o $k = 1$, a amostra teste *t* será classificado pela amostra de treino mais próxima.

D. Classificador Bayesiano

O Bayes é um classificador de padrões que se baseia em probabilidades e custos. Assume-se que o problema de decisão é mapeado em termos probabilísticos e que os valores são conhecidos [2]. Esse classificador vem sendo amplamente utilizado em diversas aplicações devido a sua simplicidade e eficiência.

Seja $p(w_i \vee x)$ a probabilidade de que uma determinada amostra $x \in R^n$ pertence a $w_i$classe, $i = 1,2, \dots, c$. Essa probabilidade é derivada a partir do Teorema de Bayes [6]:

$$p(w_i \vee x) = \frac{p(x \vee w_i) P(w_i)}{p(x)}, \quad (1)$$

onde $p(x \vee w_i)$ é a probabilidade de observar o vetor de característica x dado pela classe $w_i, P(w_i)$ é a probabilidade a priori da classe w_i e $p(x)$ representa a probabilidade de x. O Bayes atribui um rótulo (classe) w_i para uma amostra x quando

Os valores de probabilidade de $P(w_i)$ podem ser obtidos por meio do calculo e representado por histograma das classes. No entanto, estimar $p(x \vee w_i)$ é uma tarefa complexa, visto que a única informação disponível é o conjunto de vetores de característica e seus respectivos rótulos. Para isso, assume que a função de verossimilhança é Gaussiana, e, portanto podem-se estimar os parâmetros do conjunto de dados [4].

V. METODOLOGIA

Na primeira etapa, foi realizado um estudo detalhado sobre fluxos de redes, principalmente os campos que o integram. Por meio de análise, foi possível combinar características de *IDS* e ferramental teórico de modo a obter a detecção eficiente por meio de assinaturas utilizando álgebra relacional.

Com os fluxos já armazenados e com as três assinaturas criadas, o *IDS* desenvolvido realiza a verificação dos fluxos armazenados no banco de dados, e caso o comportamento da rede consultado esteja de acordo com os padrões estabelecidos pelas assinaturas (descritos na SeçãoIII), o *IDS* gera o alerta informando o *host* (vítima) sobre atos maliciosos previstos. Se o fluxo analisado não corresponder a nenhuma das assinaturas, o *IDS* marca o fluxo como comportamento normal, caso

contrário o alerta é gerado informando que a rede sofreu ataque e qual ataque foi realizado.

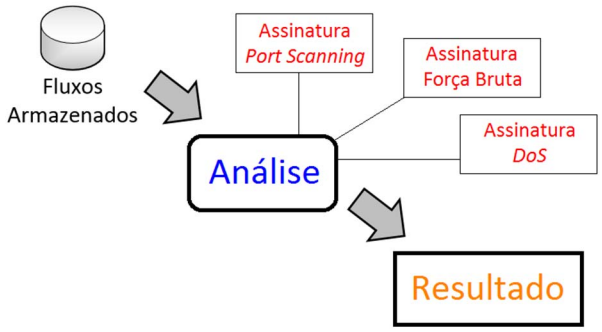

Figura 1. Funcionamento do *IDS*.

A Fig. 1 mostra como é realizado a análise do *IDS*, onde são selecionados os fluxos que estão armazenados para serem analisados, posteriormente é verificado se esses fluxos possuem características padrões aos das assinaturas dos ataques e então mostra os resultados da análise.

Após o *IDS* ter detectado de forma eficiente todos os ataques contidos na base de dados, a segunda etapa compreende a implementação de uma aplicação Java que atribui para cada fluxo a sua respectiva classe (1: Comportamento normal, 2: *Port Scanning*, 3: Força Bruta, 4: *DoS*). Com todos os fluxos já rotulados, a aplicação Java exporta para um arquivo texto os fluxos com suas informações. Este arquivo texto é gerado num formato pré-organizado exigido pelos métodos de aprendizado de máquina.

```
21413 4 7
0 1 1 78 38373 53 17 0 78.0
1 1 1 56 53 38373 17 0 56.0
2 1 1 67 44228 53 17 0 67.0
3 1 1 56 53 44228 17 0 56.0
4 1 1 67 56159 53 17 0 67.0
5 1 1 56 53 56159 17 0 56.0
...
21412 4 46453 1300684 15 8 1 35000 28.0
```

Figura 2. Formato de arquivo de entrada para algoritmos de aprendizado de máquina, no caso do OPF.

Conforme mostrado a Fig. 2, na primeira linha temos o número *N* de amostras, o número *C* de classes e o número *F* de características. Nas linhas seguintes, temos as informações das N amostras, são elas, respectivamente: *id* é valor associado a amostra de modo a identifica-la (começa em 0 e termina em N-1), a classe c_{id} que denota a classe da amostra *id*, e seu vetor característica f_{id}. As características são representadas pelo número de pacotes no fluxo, o número de *bytes* no fluxo, porta origem, porta destino, protocolo, duração do fluxo (em milissegundos) e quantidade de *bytes* por pacote.

Para cada um dos métodos de aprendizado de máquina foram analisados os resultados, levando em consideração o tempo e o número de erros e acertos.

A execução dos algoritmos de aprendizado de máquina foi realizada em um computador com processador Intel(R) Xeon(R) CPU E3-1240 v3 3.40GHz e 16 GB de memória RAM.

VI. EXPERIMENTOS

O estudo foi realizado no laboratório de pesquisa *NetBuilder* da Faculdade de Informática de Presidente Prudente (FIPP). A base de dados contém 307.641 fluxos que contabilizam 4.414.541 pacotes.

A base de dados de fluxos contém 27 ataques, conforme descriminado na Tabela I. A ferramenta *IDS* criada durante a primeira etapa da pesquisa conseguiu detectar todos os ataques descritos.

TABELA I
COMPORTAMENTOS MALICIOSOS QUE CONTÉM NA BASE DE DADOS

Comportamento Malicioso	**Quantidade**
Port Scanning	4
Força Bruta	20
DoS	3
Total	**27**

Após a detecção utilizando a abordagem de análise baseada em fluxos, foram rotulados todos os fluxos da base de dados com a sua respectiva classe, a Tabela II mostra a quantidade de fluxos por classe.

TABELA II
QUANTIDADE DE FLUXOS POR CLASSE

Classe	**Quantidade**
Normal (1)	296.228
Port Scanning (2)	6.948
Força Bruta (3)	4.419
DoS (4)	46

Com os fluxos rotulados, a aplicação Java desenvolvida realizou a exportação das informações dos fluxos (características e respectiva classe) no formato de entrada para ser usado pelos algoritmos classificadores em um arquivo texto.

Esse arquivo texto é submetido a experimentos, que são usados todos os fluxos das classes de comportamentos maliciosos, porém para a classe de comportamentos legítimos, foi separada uma quantidade específica, onde se tem 10.000 fluxos normais (1), 6.948 fluxos de *port scanning* (2), 4.419 fluxos de força bruta (3) e 46 fluxos de *DoS* (4).

Na sequência foram realizados experimentos com os quatro métodos de classificação de padrões usando 10 % das amostras para o treinamento. A Tabela III demonstra que para cada algoritmo de aprendizado de máquina, a taxa de acerto por classe usando dados não normalizados.

TABELA III
TAXA DE ACERTO POR CLASSE (NÃO NORMALIZADO)

	Normal	***Port Scanning***	**Força Bruta**	***DoS***
Bayes	97,48% ± 0,48	97,87% ± 0,32	97,80% ± 0,46	93,33% ± 4,01
KNN	97,47% ± 0,48	97,90% ± 0,33	97,80% ± 0,46	93,33% ± 4,01
OPF	99,17% ± 0,20	99,48% ± 0,24	99,83% ± 0,05	100,00% ± 0,00

SVM	98,03% ± 5,24	61,82% ± 13,44	55,22% ± 1,54	15,23% ± 2,00

Outro item relevante a ser observado na Tabela III são os classificadores que tiveram as melhores taxas de acerto como OPF e o Bayes. O trabalho de Pereira[12], que realiza a classificação de comportamentos maliciosos em redes de computadores usando algoritmos de aprendizado de máquina, os dois classificadores que mais se destacaram em termos de acurácia (taxa acerto) também foram o OPF e o Bayes.

Outro experimento utilizando também 10% das amostras para treinamento, porém, com dados normalizados. A Tabela IV demonstra os resultados para cada classificador e a taxa de acerto por classe.

TABELA IV
TAXA DE ACERTO POR CLASSE (NORMALIZADO)

	Normal	***Port Scanning***	**Força Bruta**	***DoS***
Bayes	99,02% ± 0,12	99,51% ± 0,15	99,93% ± 0,10	100,00% ± 0,00
KNN	98,73% ± 0,45	99,72% ± 0,22	99,94% ± 0,09	100,00% ± 0,00
OPF	93,57% ± 7,10	78,44% ± 20,06	50,95% ± 12,84	100,00% ± 0,00
SVM	99,10% ± 0,10	99,96% ± 0,00	99,92% ± 0,14	90,00% ± 10,39

Neste experimento, o classificador SVM melhorou consideravelmente a sua taxa de acerto, ao contrário do classificador OPF que obteve uma taxa de acurácia menor em relação aos outros algoritmos. Com os dados normalizados, os melhores resultados obtidos foram do classificador Bayesiano e o KNN.

Dado c classes a matriz de confusão M_c de dimensão cxc representa as taxas de acertos e erros de cada classe. Cada elemento $M_{c\{i,j\}}$ representa a porcentagem de amostras da classe i que foram classificadas com o rótulo j. A seguir são apresentadas as Tabelas V e VI das matrizes de confusão usando dados não normalizadas e normalizados, respectivamente.

TABELA V
MATRIZ DE CONFUSÃO DO CLASSIFICADOR OPF NO EXPERIMENTO COM DADOS NÃO NORMALIZADOS

	Normal	***Port Scanning***	**Força Bruta**	***DoS***
Normal	0.99	0.01	0.00	0.00
Port Scanning	0.01	0.99	0.00	0.00
Força Bruta	0.00	0.00	1.00	0.00
DoS	0.00	0.00	0.00	1.00

TABELA VI
MATRIZ DE CONFUSÃO DO CLASSIFICADOR BAYES NO EXPERIMENTO COM DADOS NORMALIZADOS

	Normal	***Port Scanning***	**Força Bruta**	***DoS***
Normal	0.99	0.01	0.00	0.00
Port Scanning	0.01	0.99	0.00	0.00
Força Bruta	0.00	0.00	1.00	0.00
DoS	0.00	0.00	0.00	1.00

As Tabelas VII e VIII apresentam o tempo médio de treinamento e teste em segundos dos experimentos realizados para cada um dos métodos utilizados, usando os dados não normalizados e normalizados, respectivamente.

TABELA VII
TEMPO MÉDIO DE TREINO E TESTE EM SEGUNDOS (DADOS NÃO NORMALIZADOS)

	Treinamento (s)	**Teste (s)**
Bayes	0,0319	1,7814
KNN	99,6357	0,3015
OPF	0,1182	0,5037
SVM	265,7153	1,8933

TABELA VIII
TEMPO MÉDIO DE TREINO E TESTE EM SEGUNDOS (DADOS NORMALIZADOS)

	Treinamento (s)	**Teste (s)**
Bayes	0,0309	1,8671
KNN	98,8633	0,3225
OPF	0,0950	0,3476
SVM	65,9257	0,1531

Bayes foi o método que apresentou menor tempo de treinamento em ambos cenários (dados normalizados e não normalizados), contudo apresentou alto tempo para classificação das amostras do conjunto de testes, inviabilizando seu uso para aplicações em tempo real. O SVM por sua vez apresentou excelente acurácia e se mostrou muito rápido, contudo teve tempo de treinamento muitíssimo alto (cerca de 650 vezes o tempo gasto pelo OPF). O OPF obteve o melhor resultado para dados não normalizados e tempo de treinamento e teste compatíveis com os melhores, assim o OPF se mostrou uma opção viável quando se leva em consideração um compromisso entre acurácia e tempos de treinamento e teste.

VII. CONCLUSÃO

A primeira parte deste trabalho aborda a implementação de um *IDS* que realiza a análise em fluxos de rede e realizou a detecção de três comportamentos maliciosos selecionados. Essa ferramenta *IDS* detectou todos três tipos de atos maliciosos gerados de forma controlada, detectando ataques de força bruta originados por terceiros a partir da rede FIPP. Os outros dois comportamentos maliciosos não foram detectados a partir da rede FIPP devido às configurações do *firewall* que bloqueia alguns pacotes.

Na segunda etapa, foram utilizados algoritmos de aprendizado de máquina (Bayes, KNN, OPF e SVM). Nesse contexto foi observado que é viável em termos de acurácia e

tempo, a utilização desses algoritmos no contexto de detecção de comportamento de rede na identificação de fluxos legítimos e maliciosos.

O classificador OPF obteve excelentes resultados considerando-se o compromisso entre taxa de acerto e tempo de classificação quando comparado com outros classificadores. O classificador SVM ao utilizar dados não normalizados foi o que obteve a pior taxa de acerto. Entretanto ao normalizar os dados, obteve ótima taxa de acertos e teve uma redução considerável no tempo gasto na etapa de treinamento e classificação.

Como trabalhos futuros sugerem-se realizar a detecção de outros tipos de ameaças, como por exemplo, ataques de execução de código arbitrário em páginas na *Internet* (*cross-site script*) e envenenamento de memória do *DNS* (*ofuscacion* DNS).

REFERÊNCIAS

[1] BATISTA, Maira Lambort. Análise de Eventos de Segurança em Redes de Computadores Utilizando Detecção de Novidade. Dissertação de Mestrado em Ciência da Computação, Universidade Estadual Paulista "Júlio de Mesquita Filho", São José do Rio Preto, São José do Rio Preto – SP, 2012.

[2] CERQUEIRA, Pedro H. Ramos. Um Estudo Sobre Reconhecimento de Padrões: Um Aprendizado Supervisionado Com Classificador Bayesiano. Dissertação de Mestrado em Ciências, Universidade de São Paulo Escola Superior de Agricultura "Luiz de Queiroz", Piracicaba – SP, 2010.

[3] CORRÊA, Jorge Luiz. Um Modelo de Detecção de Eventos em Redes Baseado no Rastreamento de Fluxos. Dissertação de Mestrado em Ciência da Computação, Universidade Estadual Paulista "Júlio de Mesquita Filho", São José do Rio Preto – SP, 2009.

[4] DUDA, R. O. et. al. Patters Classification – 2nd edition, Wiley-Interscience Publication, 2000.

[5] HALL, Peter et al. Choice of neighbor order in nearest-neighbor classification. The Annals of Statistics, 2008.

[6] JAYNES, E. T. Probability Theory: The Logic of Science. Cambridge University Press, 2003.

[7] KUROSE, James F., Redes de Computadores e a Internet: Uma Abordagem Top-Down – 5ª Edição, São Paulo, Editora Pearson, 2010.

[8] LIMA, Igor. Uma Abordagem Simplificada de Detecção de Intrusão Baseada em Redes Neurais Artificiais. Dissertação de mestrado em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis – SC, 2005.

[9] MCCLURE, Stuart et al. Hackers Expostos – 4ª Edição, Rio de Janeiro, Editora Campus Ltda, 2003.

[10] MORENO, Daniel, Introdução ao Pentest – São Paulo, Editora Novatec Ltda, 2015.

[11] PAPA, João Paulo. Classificação Supervisionada de Padrões Utilizando Floresta de Caminhos Ótimos. Tese de Doutorado em Ciência da Computação, Universidade Estadual de Campinas, Campinas – SP, 2009.

[12] PEREIRA, Clayton Reginaldo. Detecção de Intrusão em Redes de Computadores Utilizando Floresta de Caminhos Ótimos. Dissertação de Mestrado em Ciência da Computação, Universidade Estadual Paulista, São José do Rio Preto – SP, 2012.

[13] STALLINGS, Willian, Criptografia e Segurança em Redes – 4ª Edição, São Paulo, Editora Pearson, 2008.

[14] VAPNIK, Vladimir. An Overview of Statistical Learning Theory. IEEE Transactions on Neural Networks, 1999.

Eduardo Massato Kakihata received his B.Sc. in Computer Science from from the University of Western São Paulo (Unoeste) in 2015.

Helton Molina Sapia received his B.Sc. in Computer Science from the University of Western São Paulo (Unoeste), SP, Brazil in 1995. Actually he is master student at São Paulo State University (UNESP). He has been Professor at University of Western São Paulo (Unoeste), Brazil.

Ronaldo Toshiaki Oiakawa received his B.Sc. in Information System from from the University of Western São Paulo (Unoeste) in 2003. In 2015, he received his M.Sc. in Enviroment Science from University of Western São Paulo (Unoeste). He has been Professor at University of Western São Paulo (Unoeste), Brazil.

Danillo Roberto Pereira received his B.Sc. in Computer Science from the São Paulo State University (UNESP), SP, Brazil in 2006. In 2009, he received his M.Sc. in Computer Science from University of Campinas (UNICAMP), SP, Brazil. In 2013, he received his Ph.D. in Computer Science from the University of Campinas, SP, Brazil. He concluded the post-doctoral at Sao Paulo State University in 2016. Actually he is post-doctoral student at Federal University of São Carlos and he has been Professor at University of Western São Paulo (Unoeste), Brazil. His research interests include machine learning, nonlinear optimization, finite element methods and image processing.

João Paulo Papa received his B.Sc. in Information Systems from the Univ Estadual Paulista (UNESP). In 2005 and 2008, respectively, he received his M.Sc. and Ph.D. in Computer Science from the Federal University of São Carlos and University of Campinas, Brazil. He concluded the post-doctoral at University of Campinas in 2009 and at Harvard University in 2015. He has been Assistant Professor at the Computer Science Department, Univ Estadual Paulista (UNESP), since 2009. His interests include computer vision and machine learning.

Victor Hugo C. de Albuquerque Phd in Mechanical Engineering with emphasis on materials by Federal University of Paraíba (UFPB, 2010), Master in Computer Engineering from Federal University of Ceará (UFC, 2007), degree in Technology in Mechatronics the Federal Center of Technological Education of Ceará (CEFETCE, 2006), Mechanical Technician by the Federal Center for Education Technology of Ceará (CEFETCE, 2004). It is currently Adjunct Professor in Applied Informatics Graduate Program at the University of Fortaleza (UNIFOR). Has experience in the area of intelligent systems, visualization and interaction and machine learning.

Francisco Assis da Silva graduated in Computer Science at University of Western Sao Paulo (Unoeste), Brazil (1998), Master in Computer Science at Federal University of Rio Grande do Sul (UFRGS), Brazil (2002) and Ph.D. in Science, Electrical Engineering program at University of São Paulo (USP), Brazil (2012). He is currently professor at University of Western São Paulo (Unoeste), Brazil.