Sistema Distribuído para Detecção de Ameaças em Redes Utilizando *Deep Learning*

Fábio César Schuartz, Mauro Sérgio Pereira Fonseca e Anelise Munaretto Fonseca

***Resumo*— A detecção de ameaças na Internet é um fator essencial para manter a segurança de dados e informações. Um sistema de detecção de ameaças tenta prevenir que esses ataques ocorram através da análise de padrões e do comportamento do fluxo de dados na rede. Este artigo apresenta uma extensão para a plataforma distribuída de detecção e análise de dados em grande fluxo, apresentada no trabalho de Schuartz *et al.*, através do uso de *deep learning* para redução do espaço de características. A avaliação do sistema se baseia através da acurácia, do número de falsos positivos e de falsos negativos, onde cada classificador apresentou melhor acurácia ao utilizar 5 e 13 atributos. Ainda, o sistema apresentou menor número de falsos positivos e negativos, permitindo a detecção de ameaças em tempo real sobre um grande volume de dados, com maior precisão.**

***Palavras-Chave*— Aprendizado de máquinas, aprendizagem profunda, grande volume de dados, mineração de dados, sistema de detecção de ameaças, tempo real.**

***Abstract*— Detecting threats on the Internet is a key factor in maintaining data and information security. An intrusion detection system tries to prevent such attacks from occurring through the analysis of patterns and behavior of the data flow in the network. This paper presents an extension to the distributed large data flow detection and analysis platform presented in the work of Schuartz *et al.*, through the use of deep learning to reduce the feature space. The evaluation of the system is based on accuracy, number of false positives and false negatives, where each classifier presented better accuracy using 5 and 13 attributes, besides having fewer false positives and negatives, allowing the detection of real-time threats over a large volume of data with greater accuracy.**

***Keywords*— Big data, data mining, deep learning, intrusion detection system, machine learning, real-time.**

I. Introdução

O crescimento no uso da Internet criou uma necessidade maior em proteger os dados e informações guardadas em servidores centralizados e distribuídos, principalmente em sistemas acessados através de uma rede pública. Pessoas ganham benefícios e companhias geram lucro gerenciando seus recursos e transações através da rede, criando maiores oportunidades para usuários maliciosos roubarem informações pessoais e secretas. Segundo Leu [1], nos últimos anos diversas estatísticas mostram um número crescente de invasões reportadas no *Symantec Global Internet Security Threat Report*.

Um sistema de detecção de intrusão (IDS - Intrusion Detection System) é um sistema utilizado para monitorar as atividades de outro sistema ou de uma rede, procurando por atividades maliciosas e produzindo mensagens de alerta para a estação de controle, conforme Tan [2]. Um IDS consiste de dois componentes: detecção por assinaturas e detecção por anomalias. A detecção por assinaturas é utilizada para procurar por ataques baseado em padrões extraídos de invasões conhecidas, enquanto a detecção por anomalias tenta descobrir ataques baseado no comportamento do tráfego que difere do padrão normal de comportamento.

Uma maneira de estudar os ataques ocorridos na rede é utilizar técnicas de aprendizagem de máquina, procurando por padrões no tráfego da rede. Um classificador pode estudar diversos exemplos de entradas que produzem resultados conhecidos, através de um aprendizado supervisionado, conforme Harbi [3]. Porém, os ataques evoluem e podem não seguir os padrões de ataques anteriores. Assim, as técnicas de aprendizado não supervisionado podem apresentar melhores resultados, pois procuram por variações em padrões conhecidos, ao invés de classificar o tráfego em apenas uma categoria.

O aumento no volume, velocidade e variedade dos dados nas redes atuais demanda uma infraestrutura robusta de segurança. Monitorar e processar dados em altas taxas sem desperdiçar recursos é um enorme desafio atual, segundo Lopez et al. [4]. Uma área de pesquisa atual recebendo grande atenção é o de *deep learning*. Esta é uma sub-área avançada da aprendizagem de máquina, que permite sobrepujar as limitações do aprendizado superficial. Sua característica superior de camadas de aprendizagem pode resultar em desempenho superior ou equivalente, comparado às técnicas de aprendizado superficiais.

Este artigo propõe uma expansão do sistema distribuído para detecção de ameaças em tempo real através da análise de grandes volumes de dados, pelo processamento por fluxo, apresentado no trabalho de Schuartz *et al.* [5]. O objetivo é utilizar o método de *deep learning* na camada de pré-processamento de dados para obter uma redução no espaço de características dos dados, resultando assim, em maior acurácia na detecção de ameaças, com menor índice de falsos-positivos e falsos-negativos. Foram realizadas simulações para três casos: utilização dos 41 atributos existentes na base de dados (sem uso de *deep learning*), utilização da redução do espaço das características para 5 atributos e para 13 atributos. Resultados avaliados mostram que utilizando o método de *deep learning* é possível obter resultados com maior acurácia e menor incidência de falsos-positivos e falsos-negativos. Para avaliação do sistema, utilizou-se um conjunto de dados com as classes marcadas, contendo dados normais e ataques, proveniente de uma base de dados de teste KDD́99, transformado em fluxos.

Este artigo é dividido em cinco seções, onde a seção 2 apresenta os trabalhos relacionados. A seção 3 apresenta o

Fábio César Schuartz, Mauro Sérgio Pereira Fonseca e Anelise Munaretto Fonseca, CPGEI, Universidade Tecnólogica Federal do Paraná, Curitiba, Paraná (41) 30332–0250, E-mails: phabyo@gmail.com, maurofonseca@utfpr.edu.br, anelise@utfpr.edu.br.

sistema proposto. Os resultados obtidos são apresentados na seção 4. Por fim, a seção 5 finaliza o artigo.

II. Trabalhos Relacionados

Em 2017, um sistema de detecção de intrusão em redes (NIDS - *Network Intrusion Detection System*) baseado em anomalias foi construído por Van [6] utilizando técnicas de aprendizagem profunda (*deep learning*). Essa técnicas mostraram o grande poder dos modelos generativos com boa classificação, capazes de deduzir parte do seu conhecimento de dados incompletos e adaptar-se. O trabalho foi capaz de detectar intrusões baseadas em anomalias e classificá-las em cinco grupos, com acurácia baseada nas fontes de dados de rede.

Ainda em 2017, no trabalho de Kim [7] são mostradas algumas limitações em IDSs anteriores que utilizam apredizagem de máquina clássicas e introduzem o aprendizado de características, incluindo a construção, extração e seleção de características para superar os desafios. Também discutem algumas técnicas de *deep learning* e suas aplicações para utilização em IDS.

Em 2017, Alom [8] apresentou um trabalho sobre IDS utilizando técnicas de *deep learning* não-supervisionadas. As amostras de entrada são codificadas numericamente para a aplicação de técnicas não-supervisionadas, como *Auto-Encoder* (AE) e *Restricted Boltzmann Machine* (RBM), para extração de características e redução de dimensionalidade. Então é aplicado agrupamento iterativo *k-means* para aglomeração em um espaço dimensional menor com apenas 3 atributos.

Em 2017, Schuartz [5] apresentou uma proposta de uma plataforma distribuída para detecção de ameaças em tempo real, utilizando *big data*. A plataforma proposta é capaz de detectar diversos tipos de ataques com precisão acima de 90% e baixo número de falsos-positivos e falsos-negativos. O sistema utiliza 41 atributos da base de dados para o treinamento e detecção de ameaças, resultando em falha na detecção de alguns ataques em específicos, devido ao treinamento imparcial dos diversos tipos de ataques e pela baixa representatividade de tais ataques dentro da base de dados.

Em 2018, Shone [9] propôs uma técnica de *deep learning* para detecção de intrusões utilizando um auto-codificador não-simétrico de profundidade (NDAE - *Nonsymmetric Deep Autoencoder*) para aprendizado de características não-supervisionadas. O modelo de classificação proposto foi desenvolvido utilizando o TensorFlow e uma unidade de processamento gráfica (GPU - *Graphics Processing Unit*) e foi avaliada através das bases de dados KDD Cup '99 e NSL-KDD.

Embora existam diversas técnicas e propostas na literatura para a detecção de intrusão, a maioria delas ainda não são eficientes o suficiente para trabalhar com um grande fluxo de dados (*Big Data*) em tempo real, enquanto outras propostas não apresentam acurácia suficiente. Este artigo propõe a utilização do método *deep learning* para a redução do espaço de características dos fluxos de dados, permitindo assim o treinamento e a detecção de ameaças de maneira mais eficiente e precisos.

III. Esquema Proposto

O sistema proposto visa expandir a plataforma aberta para coleta, distribuição, análise e processamento de dados proveniente de um fluxo de dados, apresentada em [5]. O processo consiste em, durante a fase de normalização das informações coletadas, utilizar o método de *deep learning* para reduzir o espaço de características dos dados, fornecendo maior acurácia na detecção de ameaças, maior rapidez no treinamento e detecção de anomalias, com redução no número de falsos-positivos e falsos-negativos.

O objetivo do método *deep learning* é aprender atributos hierárquicos de menor nível em atributos de maior nível. O método pode aprender características independentemente em múltiplos níveis de abstração e então descobrir funções de mapeamento complexas entre a entrada e a saída diretamente dos dados puros sem depender de atributos customizados por especialistas. Em abstrações de alto nível, os humanos normalmente não identificam as relações e conexões de uma entrada sensorial pura. Assim, a habilidade em aprender características complexas, também chamadas de extração de características, se torna necessária em vista do aumento na quantidade de dados [10].

A extração de características através do codificador automático empilhado (SAE - *Stacked Auto Encoder*) é capaz de reduzir a complexidade das características originais do conjunto de dados. Entretanto, além de ser um extrator de características, o SAE também pode ser utilizado para tarefas de classificação e aglomeração. É possível melhorar o processo de aprendizagem de características através da combinação da extração de características empilhadas com seleção de características com pesos. A extração de características do SAE é capaz de transformar as características originais em uma representação mais significativa ao reconstruir seus dados de entrada e fornecendo um meio de verificar a informação relevante nos dados capturados. O SAE pode ser eficientemente utilizado no aprendizado não-supervisionado em um conjunto de dados complexos.

A estrutura do SAE é apresentada na Figura 1. O primeiro *auto-encoder* recebe uma entrada e realiza uma redução nas características, procurando obter como saída, através de um *decoder*, a entrada original, pelo método de retropropagação (*back-propagation*). H_1 representa a camada escondida (*hidden layer*) do *auto-encoder* 1. Na sequência, H_1 será a entrada do *auto-encoder* 2, que irá gerar H_2 através de processo semelhante e que será a entrada do *auto-encoder* 3, resultando em H_3. Observando de uma percepção multi-camadas (*multilayer*), será o equivalente da entrada passar por 3 camadas escondidas, gerando a saída reduzida.

A. Auto-Encoder (AE)

Um *auto-encoder* é uma estratégia de rede neural profunda utilizada no aprendizado de características não-supervisionadas com codificação eficiente. O objetivo principal do AE é o aprendizado e representação (codificação) do dado, tipicamente com a finalidade da redução da dimensionalidade do dado. Esta técnica de AE consiste em duas partes: o codificador e o decodificador. Na fase de codificação, os mapas

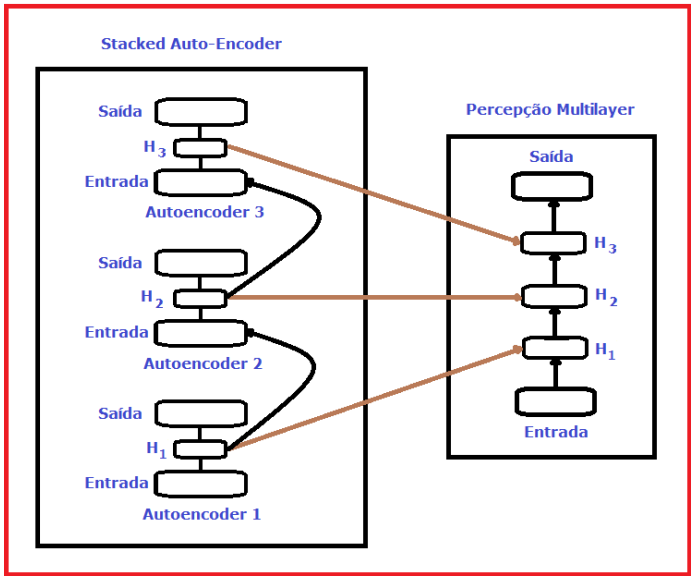

Fig. 1. Exemplo de uma estrutura *Stacked Auto-Encoder*, para três camadas.

de amostras de entradas são reduzidas para um espaço de características de dimensão menor, contendo as características construtivas mais importantes. Este processo pode ser repetido até alcançar o espaço dimensional de característica desejada. Na fase de decodificação, são reconstruídas as características ogirinais a partir de uma dimensão menor de características, através do processo de reversão. O diagrama conceitual do AE é mostrado na Figura 2. As transições de codificação e decodificação podem ser representadas por ϕ e φ:

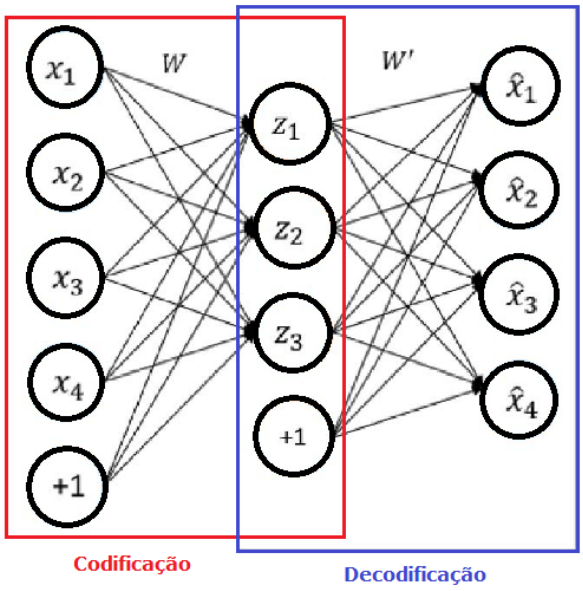

Fig. 2. Diagrama para um *auto-encoder* com as fases de codificação e decodificação.

$$\phi : \chi \rightarrow F \tag{1}$$

$$\varphi : F \rightarrow \chi \tag{2}$$

$$\phi, \varphi = argmin_{\phi,\varphi} \|X - (\phi, \varphi)\|^2 \tag{3}$$

Se considerarmos o AE mais simples com apenas uma camada escondida, onde a entrada é $xR^d = X$, o qual é mapeada em $XR^p = F$, então a expressão pode ser escrita na seguinte operação:

$$z = \sigma_1(W * x + b) \tag{4}$$

onde W é a matriz de peso e b é o *bias*. σ_1 representa a função de ativação, tal como uma sigmóide ou uma unidade linear retificada (RLU). Considerando z novamente mapeado ou reconstruído em x', com a mesma dimensão de x, a reconstrução pode ser expressa como:

$$x' = \sigma_2(W' * z + b') \tag{5}$$

Estas técnicas podem ser treinadas com o mínimo de erros de reconstrução:

$$L = \|x - x'\|^2 == \|x - \sigma_2(W' * \sigma_1(W * x + b) + b')\|^2 \tag{6}$$

Normalmente o espaço de características de F possui a menor dimensão do espaço de características de entrada X, que pode ser associada como a representação comprimida da amostra de entrada. No caso de um AE de multicamadas, a mesma operação será incorporada a medida que for necessária nas fases de codificação e decodificação.

B. Conjunto de Dados

A base de dados KDD'99 [11] é um conjunto de dados de referência que foi simulado em ambiente de rede militar em 1998 e derivado em conjunto de atributos em 1999. O pacote da base de dados foi reunido e pré-processado em 41 atributos. A base de dados contém quatro categorias de ataques diferentes (DoS, R2L, U2R e probing), sendo um total de 22 tipos de ataques na base de treinamento e 14 tipos de ataques extras na base de teste que não existem na base de treinamento.

C. Protótipo do Sistema Proposto

Inicialmente, a coleta de dados é feita em um único ponto da rede. Esses dados foram extraídos de uma base de dados KDD'99, amplamente utilizada e testada na comunidade. Os dados são pré-processados e caracterizados em fluxos compostos de 41 atributos, utilizados para detectar 22 tipos de ataques. Estes fluxos de dados são encaminhados para o SAE (*Stacked Auto-Encoder*), composto por três camadas escondidas que irão realizar a redução no espaço de características de entrada de 41 atributos para 5 e 13 atributos principais. Esses mapas reduzidos serão então publicados na rede.

Em outro ponto da rede, três unidades distintas de processamento de fluxos irão receber as informações publicadas na rede, agindo, cada um, como um assinante para o mesmo fluxo de dados. Cada unidade irá, então, alimentar esse fluxo

para a topologia criada no *Apache Storm*. Dentro do *Storm*, é criado um classificador utilizando a ferramenta *Weka*. Os classificadores foram treinados utilizando um conjunto de dados de treinamento do KDD'99. Cada unidade possuirá um algoritmo de aprendizagem de máquina diferente e irá processar o fluxo de dados recebido, caracterizando o mesmo em um tipo específico de ataque ou em um fluxo normal de dados. Por último, os resultados obtidos por cada unidade de processamento são enviados para uma única interface de visualização, onde serão exibidos. Cada unidade realizará três modelos de treinamento e classificação. Inicialmente será alimentado um fluxo contendo os 41 atributos, sem o uso de *deep learning*. Em seguida, o processo será repetido, porém com um fluxo contendo 5 atributos. E por último, o processo será feito com um fluxo de dados contendo 13 atributos, sendo esses dois últimos processos utilizando o método de *deep learning* para redução do espaço de características na base de dados.

Neste protótipo, foram escolhidos três classificadores presentes na ferramenta *Weka*:

1) Árvores de Decisão: Uma árvore de decisão, ou árvore de classificação, é um sistema de suporte à decisão que utiliza um gráfico na forma de árvore para a tomada de decisões e seus possíveis efeitos posteriores. O algoritmo é usado para aprender uma função de classificação que decide o valor de um atributo dependente (uma variável), considerando os valores dos atributos independentes de entrada, conforme Bhargava [12].

A árvore de decisão J48 é a implementação do algoritmo ID3 (*Iterative Dichotomiser 3*) pelo *WEKA*. Maiores detalhes do algoritmo pode ser encontrado no trabalho de Quinlan [13].

2) Naive Bayes: Naive Bayes é uma técnica probabilística para construção de classificadores baseado no teorema de Bayes, onde assume-se uma forte independência entre os atributos.

Classificadores *Naive Bayes* são escalonáveis e podem processar um grande número de variáveis lineares (parâmetros) em uma tarefa de aprendizagem. Em uma única iteração dos dados de treinamento, o algoritmo calcula a probabilidade de distribuição condicional de cada atributo de um determinado rótulo, seguido pela aplicação do teorema de Bayes para determinar a distribuição da probabilidade condicional do rótulo, usado para a previsão do resultado, de acordo com Aggarwal [14].

Maiores informações da implementação do *Naive Bayes* pelo *WEKA* pode ser encontrado no trabalho de Langley [15].

3) Tabelas de Decisão: Uma tabela de decisões é uma tabela que associa condições com ações a serem tomadas, apresentando um resultado após seguir uma série de decisões relacionadas. Ela permite modelar um conjunto complexo de regras com suas ações correspondentes.

No trabalho de Kohavi [16], podem ser encontradas maiores informações da Tabela de Decisão implementada pelo *WEKA*.

Neste protótipo, o objetivo é a prova de conceito da caracterização dos fluxos de dados em tempo real. Assim, não será utilizado o processamento em lotes e o armazenamento das informações em um banco de dados, que permitem a realimentação de parâmetros para os algoritmos se adaptarem em tempo real. Os parâmetros calculados no processamento *off-line* com os dados históricos servem para ajustar o modelo de processamento em tempo real, dando ao sistema uma característica adaptativa, pois os parâmetros podem ser atualizados, se ajustando a novos padrões de uso.

IV. Resultados Numéricos

O desempenho do sistema será avaliado através de duas métricas: a acurácia (percentual de acerto da classificação sobre a base de teste), e o número de falsos positivos e falsos negativos de cada classe.

A acurácia é a relação entre o número de amostras classificadas corretamente pelo número total de amostras. O número de falsos positivos indica quantas amostras normais foram classificadas como ataque e o número de falsos negativos indica quantas amostras de ataque foram classificadas como uma amostra normal.

Cada classificador será treinado e testado utilizando inicialmente os 41 atributos da base de dados KDD'99, sem o uso do *deep learning* para redução do espaço das características, servindo assim como a base de comparação. Depois, será repetido o treino e classificação utilizando 5 atributos e depois com 13 atributos. Os resultados serão, então, comparados pelas métricas estabelecidas.

Os resultados apresentam uma maior acurácia para a classificação de ataques em cada uma das unidades de processamento de fluxos criada quando se utilizam 13 atributos. O uso de apenas 5 atributos teve sua acurácia reduzida devido aos ataques do tipo R2L e U2L, pois o modelo SAE requer quantidades maiores de dados para realizar o aprendizado corretamente. Infelizmente, devido ao pequeno número de dados de treinamento disponíveis para esses ataques, os resultados obtidos não foram satisfatórios. Entretanto, no geral, mesmo utilizando 5 atributos, obteve-se melhores resultados comparados a não utilização de *deep learning*. A Tabela I apresenta a comparação entre os distintos algoritmos de classificação em termos de acurácia para os casos de 41, 5 e 13 atributos. Pode-se observar que com a redução dos atributos obtêm-se uma maior acurácia, em geral, na classificação de ataques.

TABELA I
COMPARAÇÃO DE ACURÁCIA (%) ENTRE OS ALGORITMOS DE CLASSIFICAÇÃO, UTILIZADO 41, 5 E 13 ATRIBUTOS.

Algoritmo Classificador	**41 Atrib.**	**5 Atrib.**	**13 Atrib.**
Árvores de Decisão	96,4980	96,7387	98,2162
Naive Bayes	90,2766	90,9497	96,6401
Tabelas de Decisão	95,8408	96,1994	98,0171

Os falsos positivos e falsos negativos são mostrados na Tabela II, a qual compara os diversos algoritmos utilizados pelas unidades de processamento de fluxo considerando 41, 5 e 13 atributos. As colunas da tabela mostram o número de ataques classificados corretamente (amostras classificadas como ataque, porém de tipo diferente não são consideradas), o número de falsos positivos (conexão normal classificada como um ataque) e o número de falsos negativos (ataque classificado como conexão normal). Observa-se que, para um

mesmo fluxo de dados, o algoritmo *Naive Bayes* apresenta um número muito maior de falsos-positivos, embora sua acurácia seja próxima dos outros algoritmos de classificação. Ainda, a classificação utilizando-se 5 atributos sofre novamente com a falta de dados para o treinamento correto dos ataques R2L e U2L, aumentando assim o número de falsos-positivos e falso-negativos comparado ao uso de 13 atributos.

TABELA II
RESULTADOS OBTIDOS PELOS ALGORITMOS DE CLASSIFICAÇÃO, EM UM TOTAL DE 4.898.431 ENTRADAS, PARA 41, 5 E 13 ATRIBUTOS SELECIONADOS.

Classificador	**Ataques Certos**	**FP**	**FN**
Árvores: 41 atrib.	3.768.413	684	1.489
Árvores: 5 atrib.	3.901.872	598	1.249
Árvores: 13 atrib.	4.429.091	323	881
Bayes: 41 atrib.	3.710.219	460.861	1.676
Bayes: 5 atrib.	3.881.231	442.893	1.393
Bayes: 13 atrib.	4.399.182	329.122	901
Tabelas: 41 atrib.	3.763.595	782	2.439
Tabelas: 5 atrib.	3.891.125	633	2.219
Tabelas: 13 atrib.	4.511.297	391	1.128

Para testar o desempenho do sistema proposto em tempo real, criou-se um ambiente virtual em uma máquina com processador Intel Core i7-4770S e 8 GB de memória RAM, executando o sistema operacional Linux Ubuntu 16. A medida de desempenho foi realizada pela média de fluxos processados por minuto pelas três unidades de processamento. O sistema foi capaz de processar aproximadamente 630 mil fluxos por minuto, com incerteza de 25 mil, dentro de um intervalo de confiança de 95%. Assim, cada ataque teve um tempo de detecção aproximado de 95 microssegundos.

V. CONCLUSÕES

Este artigo propõe um sistema de detecção de intrusão distribuído em tempo real, através do processamento de fluxos, utilizando *deep learning* para redução do espaço de características. O sistema apresentado utiliza ferramentas de código aberto que permite o processamento paralelo de diversos algoritmos de aprendizado de máquina, permitindo analisar um grande volume de dados em tempo real. A avaliação do sistema proposto é sobre a base de dados KDD'99, vastamente utilizada na comunidade, que foi transformada em um fluxo rotulado. Para a classificação dos dados, foram utilizados três unidades de processamento de fluxos, cada um com um algoritmo distinto de aprendizado de máquina - árvore de decisão, *Naive Bayes* e tabelas de decisão. Os resultados obtidos por cada algoritmo são, então, encaminhados para um visualizador. Cada unidade de processamento de fluxos classificou 3 fluxos de dados diferentes. Inicialmente com 41 atributos (sem a utilização de *deep learning*, seguido por um fluxo de dados com 5 atributos e, por último, um fluxo de dados com 13 atributos. Utilizando-se *deep learning* para reduzir o espaço das características apresentou um melhor resultado em termos de acurácia e redução no número de falsos-positivos e falsos-negativos, embora ao utilizar-se apenas 5 atributos, notou-se uma falha na detecção de ameaças do tipo R2L e U2L devido a baixa representatividade desses ataques na base de dados, prejudicando o aprendizado. O sistema incorporando o método *deep learning* apresenta melhor acurácia em relação ao sistema sem redução de características, fornecendo uma solução para a detecção de ameaças em tempo real com maior desempenho.

Em trabalhos futuros, será proposta a comunicação entre as unidades de processamento de fluxos para troca de informações e a inclusão de diferentes fontes de fluxo de dados, sendo assim possível avisar os diferentes pontos da rede sobre um possível ataque à rede caso uma ameaça seja detectada por uma unidade. Será utilizado o *deep learning* não apenas para redução do espaço de características, mas também para a detecção de anomalias através da distribuição Gaussiana dos dados trafegados pela rede, permitindo a detecção de ataques desconhecidos.

REFERÊNCIAS

[1] F. Y. Leu, K. L. Tsai, Y. T. Hsiao, and C. T. Yang, "An internal intrusion detection and protection system by using data mining and forensic techniques," *IEEE Systems Journal*, vol. PP, no. 99, pp. 1–12, 2015.

[2] Z. Tan, U. T. Nagar, X. He, P. Nanda, R. P. Liu, S. Wang, and J. Hu, "Enhancing big data security with collaborative intrusion detection," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 27–33, Sept 2014.

[3] N. Harbi and E. Bahri, "Real detection intrusion using supervised and unsupervised learning," in *2013 International Conference on Soft Computing and Pattern Recognition (SoCPaR)*, Dec 2013, pp. 321–326.

[4] M. A. Lopez, A. G. P. Lobato, O. C. M. B. Duarte, and G. Pujolle, "An evaluation of a virtual network function for real-time threat detection using stream processing," in *2018 Fourth International Conference on Mobile and Secure Services (MobiSecServ)*, Feb 2018, pp. 1–5.

[5] F. C. Schuartz, M. S. P. Fonseca, and A. M. Fonseca, "Sistema distribuído para detecção de ameaças em tempo real utilizando big data," in *XXXV Simpósio Brasileirp de Telecomunicações e Processamento de Sinais - SBrT 2017*, Sept 2017, pp. 472–476.

[6] N. T. Van, T. N. Thinh, and L. T. Sach, "An anomaly-based network intrusion detection system using deep learning," in *2017 International Conference on System Science and Engineering (ICSSE)*, July 2017, pp. 210–214.

[7] K. Kim and M. E. Aminanto, "Deep learning in intrusion detection perspective: Overview and further challenges," in *2017 International Workshop on Big Data and Information Security (IWBIS)*, Sept 2017, pp. 5–10.

[8] M. Z. Alom and T. M. Taha, "Network intrusion detection for cyber security using unsupervised deep learning approaches," in *2017 IEEE National Aerospace and Electronics Conference (NAECON)*, June 2017, pp. 63–69.

[9] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, Feb 2018.

[10] Y. B. et al, "Learning deep architectures for ai," *Foundations and trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[11] W. Lee, S. J. Stolfo, and K. W. Mok, "Mining in a data-flow environment: Experience in network intrusion detection," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1999, pp. 114–124.

[12] N. Bhargava, G. Sharma, R. Bhargava, and M. Mathuria, "Decision tree analysis on j48 algorithm for data mining," *Proceedings of International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 6, 2013.

[13] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[14] C. C. Aggarwal, *Data Classification: Algorithms and Applications*, 1st ed. Chapman & Hall/CRC, 2014.

[15] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1995, pp. 338–345.

[16] R. Kohavi, "The power of decision tables," in *Proceedings of the 8th European Conference on Machine Learning*, ser. ECML '95. London, UK, UK: Springer-Verlag, 1995, pp. 174–189. [Online]. Available: http://dl.acm.org/citation.cfm?id=645324.649649