Sistema Distribuído para Detecção de Ameaças em Tempo Real Utilizando Big Data

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 plataforma distribuída para detecção e análise de dados em grande fluxo, através de algoritmos de classificação presentes em unidades de processamento de fluxo. A arquitetura do sistema permite cada classificador trabalhar em paralelo e os resultados são agrupados em uma interface de visualização, permitindo o monitoramento de cada unidade. A avaliação do sistema se baseia na acurácia, no número de falsos positivos e de falsos negativos, onde cada classificador apresentou acurácia acima de 90% e, com excessão do algoritmo *Naive Bayes*, reduzido número de falsos positivos e negativos, permitindo a detecção de ameaças em tempo real sobre um grande volume de dados.**

***Palavras-Chave*— Detecção de ameaças, IDS, mineração de dados, Big Data, aprendizado de máquinas, 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 a distributed platform for detection and analysis of data in a large flux, through classification algorithms present in flux processing units. The system architecture allows each classifier to work in parallel and the results are grouped into a visualization interface, allowing the monitoring of each unit. The evaluation of the system is based on the accuracy, number of false positives and false negatives, where each classifier presented accuracy above 90% and, with the exception of the Naive Bayes algorithm, reduced number of false positives and negatives, allowing The detection of real-time threats over a large volume of data.**

***Keywords*— Intrusion detection, IDS, data mining, Big Data, machine learning, real-time.**

I. Introdução

O desenvolvimento acelerado na tecnologia dos computadores em rede e da Internet trouxe grandes benefícios às pessoas. Entretanto, esse 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 *hackers* roubarem informações pessoais e secretas.

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

Segundo Leu [1], nos últimos anos diversas estatísticas mostram um número crescente de invasões reportadas no *Symantec Global Internet Security Threat Report*. Para combater tais ataques, diversos pesquisadores e instituições começaram a desenvolver tecnologias de detecção de invasão. Um sistema de detecção de intrusão (IDS) 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, tal como classificação e regressão, para procurar por padrões no tráfego da rede. Um classificador pode estudar diversos exemplos de entradas que produzem resultados conhecidos, através de um aprendizado supervisionado. Porém, os ataques evoluem e novos ataques, ou variações de ataques já conhecidos, 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 uma categoria.

Até o presente, nenhuma combinação de tecnologias pode proteger completamente um sistema. Assim, pesquisadores estão procurando técnicas melhores para minimizar e sobrepujar algumas limitações das técnicas de prevenção existentes. Este artigo propõe um sistema distribuído para detecção de ameaças em tempo real através da análise de grandes volumes de dados, pelo processamento por fluxo. A plataforma utiliza ferramentas de código aberto dentro de uma topologia onde vários classificadores analisarão os dados coletados de diversas fontes, procurando por possíveis ataques à rede. Estes pontos poderão se comunicar e trocar informações para aumentar o grau de acerto na classificação de ataques e detecção de anomalias. Um protótipo do sistema foi desenvolvido e avaliado e os resultados mostram uma acurácia acima de 90% na detecção de ameaças, com baixo número de falsos-positivos. 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 KDD99, transformado em fluxos.

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

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

II. Trabalhos Relacionados

Em 2003, um sistema de detecção de anomalias na rede baseado em tempo real foi proposto por Balupari [3]. O sistema analisa o fluxo de dados gerado pelo *TcpTrace* em tempo real, que reporta periodicamente estatísticas de todas as conexões TCP/IP abertas na rede. Com isso, são construídos perfis estatísticos que mostram o comportamento normal dos serviços na rede, permitindo qualquer comportamento anormal ser caracterizado como uma invasão. Embora esta abordagem seja capaz de monitorar quaisquer serviços na rede sem o conhecimento prévio do seu comportamento, ele falha em detectar fluxos proveniente de diferentes redes.

Em 2011, Holtz [4] propôs uma arquitetura para IDS em uma rede distribuída, onde a análise dos dados é realizada em um ambiente de computação em nuvem. Utilizando equipamentos de rede, como servidores e estações de trabalho, foram coletados dados do tráfego na rede, arquivos *logs* do sistema operacional e dados de aplicações gerais em diversos pontos da rede, sendo estes agregados, processados e comparados através da estrutura *Map-Reduce*. A detecção de intrusões e atividades maliciosas podem ser identificadas analisando as correlações dos eventos. Embora o sistema proposto possa manusear grandes fluxos de dados, o mesmo utiliza a ferramenta Apache Hadoop, a qual é uma ferramenta de processamento em lotes, não permitindo assim a detecção de intrusão em tempo real.

Em 2013, Devikrishna K.S et al [5] propôs um sistema de detecção de intrusão capaz de capturar dados diretamente da rede, permitindo o sistema atuar como um IDS em tempo real. Para testar o sistema, foi utilizada a base de dados KDD'99, amplamente conhecida e testada na comunidade, gerando um fluxo de pacotes em sequência. A técnica utilizada neste modelo para a detecção é baseada em assinaturas. Porém, o modelo falha em manusear grandes quantidades de dados por não possuir processamento em paralelo, o que não é adequado para obter resultados instantâneos ou em tempo real.

Em 2015, Rathore [6] propôs um IDS utilizando *Map-Reduce* usando o Hadoop para processar um arquivo sequencial e calcular os valores dos parâmetros a serem utilizados em diversos classificadores. A utilização de menos parâmetros para a classificação e o uso de um ambiente paralelo do Hadoop permitiu o processamento de grandes bases de dados em um período de tempo menor que outros IDS propostos, com similar eficiência e precisão. Porém, o sistema apresentado utiliza uma base de dados previamente coletada e processamento em lotes, não sendo adequado para utilização em um ambiente de tempo real.

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 de ferramentas de código aberto dentro de uma arquitetura específica que permite a análise e o processamento de fluxos de dados em tempo real e com acurácia.

III. Esquema Proposto

O sistema proposto visa construir uma plataforma aberta para coleta, distribuição, análise e processamento de dados proveniente de um fluxo de dados, com o objetivo de detectar ameças na rede em tempo real. O processo é dividido em segmentos que englobam a coleta de dados de diversas fontes distintas, a normalização das informações coletadas, o transporte dos dados normalizados para o sistema de processamento de informações, a análise do fluxo de dados em tempo real e em lotes, incluindo o armazenamento dos resultados para posterior análise e retroalimentação de parâmetros ao sistema, e a visualização dos resultados obtidos. A topologia do sistema proposto é apresentada na Figura 1.

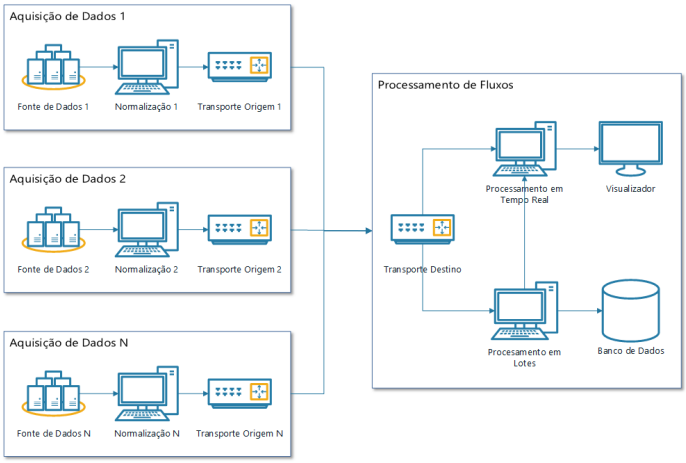

Fig. 1. Topologia do sistema proposto, considerando N fontes de dados coletados.

A. Coleta de Dados

Inicialmente, os dados são coletados de fontes independentes, em diversos locais na rede. Estas fontes são, em geral, estruturas por onde passa um grande tráfego de informações, como Pontos de Troca de Tráfego (PTT). Esses dados são coletados diretamente do equipamento roteador, através da captura dos pacotes Ethernet que entram pelas interfaces do sistema e de *logs* do sistema operacional e outras aplicações.

B. Normalização dos Dados

Os dados coletados contém diversas informações, como mensagens enviadas pelos protocolos ICMP e ARP, entre outras. Para trabalhar os dados coletados nos algoritmos de aprendizagem de máquina, é necessário obter informações de fluxo através da extração de dados dos cabeçalhos dos pacotes, utilizando, por exemplo, janelas de 2 segundos (tempo suficiente para agregar informações para a composição dos fluxos). Os fluxos, por exemplo, podem ser definidos por uma sequência de pacotes que possuem um mesmo IP de origem e um mesmo IP de destino. Assim, a partir dos cabeçalhos dos pacotes TCP/IP de cada fluxo, obtêm-se N características, como por exemplo a quantidade de portas de origem e destino, e número de pacotes TCP, UDP e ICMP.

A normalização dos dados na fonte permite transportar informações de maneira resumida, evitando consumir banda extra enviando informações desnecessárias à unidade de processamento de fluxos.

C. Transporte dos Dados

Para transportar os dados de um ponto ao outro na Internet, é utilizado um agente de mensagens, isto é, uma ferramenta para a publicação e assinatura (*publish and subscribe*) de uma fila de mensagens. Cada fonte produz dados normalizados e os publica em uma fila. Cada unidade de processamento de fluxos escolhe quais filas assinar, podendo ele escolher uma ou mais filas.

D. Processamento de Fluxos e Visualização dos Resultados

Para o processamento de fluxos, o sistema adota a arquitetura *Lambda*, a qual mistura recursos de processamento de dados em lote (*batch*) e em tempo real (*streaming*), e é dividida em três camadas de processamento: camada de processamento em lote (*Batch Layer*), camada de processamento de fluxo de dados (*Speed Layer*) e a camada de serviço (*Serving Layer*). A Figura 2 representa a arquitetura *Lambda*.

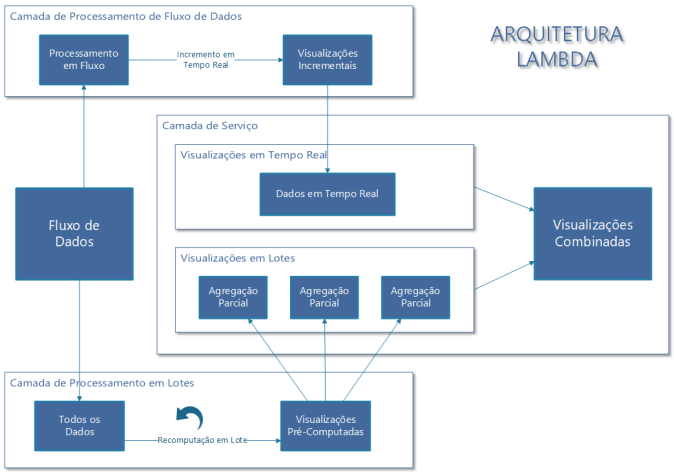

Fig. 2. Arquitetura Lambda, composta de três camadas: processamento em lote, processamento de fluxo de dados e camada de serviço.

O fluxo de dados de entrada é enviada para duas camadas: a de processamento de fluxo de dados e a de processamento em lotes. Na camada de processamento de fluxo de dados, os dados são tratados em tempo real, utilizando algoritmos de classificação por aprendizado de máquina, por exemplo. Na camada de processamento em lotes, os dados são armazenados e analisados em grandes quantidades através de técnicas de computação distribuída, como, por exemplo, o *map-reduce*. Na sequência, a camada de serviço recebe as informações obtidas das duas camadas anteriores (visualização em tempo real e visualização histórica) e cria as saídas combinadas dos dados analisados. Com isso, a arquitetura Lambda analisa, em tempo real e de forma precisa, os fluxos de dados provenientes de diferentes fontes na velocidade em que são gerados.

Assim, o sistema possui informações coletadas de diversos pontos da rede e existem diversos locais de processamento de fluxo de dados, trabalhando em paralelo. Cada unidade de processamento pode optar por receber um fluxo de dados de um ou mais locais distintos, além de poder se comunicar com outras unidades de processamento, trocando informações, caso necessário. Ao se detectar uma ameaça na rede, uma unidade de processamento pode alertar outras unidades para tomarem providências, tais como ignorar o fluxo de dados vindo de determinado endereço, ou configurar *on-the-fly* regras de proteção contra o ataque em seus equipamentos, antes mesmo do ataque ser detectado por aquela unidade de processamento em específico.

E. Protótipo do Sistema Proposto

O protótipo do sistema proposto para a detecção em tempo real de ameaças é composto de ferramentas de código aberto, distribuídas livremente na *Internet*, e código Java. A composição do sistema consiste na integração das ferramentas descritas a seguir:

F. Conjunto de Dados

A base de dados KDD'99 [7] 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.

1) Apache Kafka: Como ferramenta para publicação e assinatura, optou-se pelo Apache Kafka [8], uma plataforma de *streaming* distribuída de software livre. O Kafka é projetado para sistemas distribuídos de alto desempenho e, comparado com outros sistemas de mensagens, ele possui melhor rendimento, *built-in* de particionamento, replicação e tolerância a falhas inerentes.

2) Apache Storm: O Storm [9] é um Data Streaming Processor (DSP), isto é, um processador de eventos em tempo real. Ele é composto por topologias que formam Grafos Acíclicos Dirigidos (GAD) compostos por *spouts* (elementos de entrada) e *bolts* (elementos de processamento). Os *spouts* e *bolts* trabalham em paralelo, permitindo o processamento simultâneo de diferentes amostras de fluxos em tempo real.

O Storm é altamente escalonável, possui uma solução tolerante a falhas e garante que os dados serão processados pela topologia, sem a perda dos mesmos.

3) Apache Zookeeper: Para coordenar o grupo de nodos (*cluster*) e manter os dados compartilhados, utilizando técnicas de sincronização robustas, será utilizada a ferramenta Apache ZooKeeper [10]. O Storm não possui estados, por isso depende do Zookeeper para monitorar os estados dos nodos que estão trabalhando. Com isso, garante-se o processamento dos dados mesmo se algum dos nodos conectados no *cluster* morrer ou alguma mensagem for perdida.

4) Weka: O Weka [11] é um *software* de código aberto composto por uma coleção de algoritmos de aprendizagem de máquina utilizado para tarefas de mineração de dados. Ela contém ferramentas que podem ser aplicadas diretamente

sobre uma base de dados ou ser incorporada em código Java para o pré-processamento de dados, classificação, regressão, clusterização, associação de regras e visualização dos resultados obtidos. Os dados podem ser classificados em lotes, através de arquivos contendo os dados no formato ARFF (*Attribute-Relation File Format*), ou em fluxo, com cada dado sendo classificado individualmente.

A Figura 3 mostra como o sistema proposto interliga as ferramentas de código aberto.

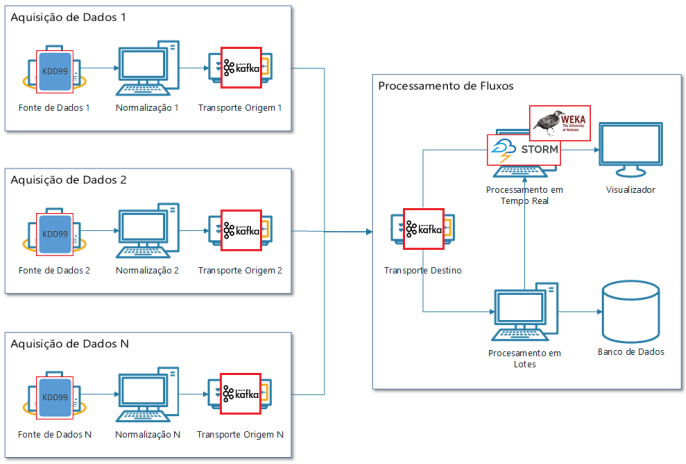

Fig. 3. Representação das ferramentas de código aberto interligadas no sistema proposto.

Inicialmente, a coleta de dados é feita em um único ponto da rede. Esses dados foram extraídos de uma base de dados KDD99, 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 então publicados na rede pela ferramenta Apache Kafka.

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 KDD99. 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.

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 caracterizção 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.

Os resultados apresentam uma alta acurácia para a classificação de ataques em cada uma das unidades de processamento de fluxos criada. A Tabela I apresenta a comparação entre os distintos algoritmos de classificação em termos de acurácia. Pode-se observar que cada um dos três algoritmos apresentam uma alta acurácia na classificação de ataques, todos acima de 90%.

Os falsos positivos e falsos negativos são mostrados na Tabela II, a qual compara os diversos algoritmos utilizados

TABELA I
COMPARAÇÃO DE ACURÁCIA ENTRE OS ALGORITMOS DE CLASSIFICAÇÃO UTILIZADOS NAS UNIDADES DE PROCESSAMENTO DE FLUXOS.

Algoritmo Classificador	Acurácia (%)
Árvores de Decisão	96,4980
Naive Bayes	90,2766
Tabelas de Decisão	95,8408

pelas unidades de processamento de fluxo. As colunas da tabela mostram o número de ataques classificados corretamente (amostras classificadas como ataque, porem 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.

TABELA II
RESULTADOS OBTIDOS PELOS ALGORITMOS DE CLASSIFICAÇÃO, EM UM TOTAL DE 4.898.431 ENTRADAS.

Classificador	Ataques Certos	FP	FN
Árvores de Decisão	3.768.413	684	1.489
Naive Bayes	3.710.219	460.861	1.676
Tabelas de Decisão	3.763.595	782	2.439

Para testar o desempenho do sistema proposto em tempo real, criou-se um ambiente virtual em uma máquina com processador Intel Core i5-5600 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. COMENTÁRIOS FINAIS

Este artigo propõe um sistema de detecção de intrusão distribuído em tempo real, através do processamento de fluxos. 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 um mesmo fluxo de dados, apresentando resultados acima de 90% de acurácia, embora o algoritmo *Naive Bayes* tenha apresentado um elevado número de falsos-positivos. O sistema apresenta grande desempenho e precisão, permitindo o processamento de *Big Data* e fornecendo uma solução para a detecção de ameaças em tempo real.

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. Serão ainda acrescentados métodos de redução de atributos para verificar se os algoritmos de aprendizado de máquina podem ser melhores optimizados, e a utilização de unidades de processamento 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] R. Balupari, B. Tjaden, S. Ostermann, M. Bykova, and A. Mitchell, "Real-time system security," B. Tjaden and L. R. Welch, Eds. Commack, NY, USA: Nova Science Publishers, Inc., 2003, ch. Real-time Network-based Anomaly Intrusion Detection, pp. 1–19. [Online]. Available: http://dl.acm.org/citation.cfm?id=903866.903869

[4] M. D. Holtz, B. M. David, and R. T. de Sousa Júnior, "Building scalable distributed intrusion detection systems based on the mapreduce framework," *Revista Telecommun*, vol. 13, no. 2, p. 22, 2011.

[5] D. K. S and R. B. B, "An artificial neural network based intrusion detection system and classification of attacks," *International Journal of Engineering Research and Applications*, 2013.

[6] M. M. Rathore, A. Paul, A. Ahmad, S. Rho, M. Imran, and M. Guizani, "Hadoop based real-time intrusion detection for high-speed networks," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–6.

[7] 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.

[8] A. S. Foundation. (2016) Apache kafka. Acessado em: 27-03-2017. [Online]. Available: http://kafka.apache.org

[9] ——. (2015) Apache storm. Acessado em: 27-03-2017. [Online]. Available: http://storm.apache.org

[10] ——. (2010) Apache zookeeper. Acessado em: 27-03-2017. [Online]. Available: http://zookeeper.apache.org

[11] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[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