CATRACA: uma Ferramenta para Classificação e Análise de Tráfego Escalável Baseada em Processamento por Fluxo*

Martin Andreoni Lopez[1,2], Igor J. Sanz[1], Diogo M. F. Mattos[1], Otto Carlos M. B. Duarte[1] e Guy Pujolle[2]

[1]GTA / PEE-COPPE / UFRJ – Brasil
[2]LIP6/CNRS (UPMC Sorbonne Universités) – França

Abstract. *The late detection of security threats causes a significant increase in the risk of irreparable damages, restricting any defense attempt. In this paper, we propose the CATRACA tool, an efficient online Intrusion Detection and Prevention System implemented as a Network Virtualized Function. The tool is based on a Big Data Streaming processing system, the Apache Spark, and it is deployed on the Open Platform for Network Functions Virtualization (OPNFV), providing an accurate real-time threat-detection service. The tool has two operating modes: on-line and off-line. The detection process includes feature selection and machine learning algorithms to differentiate normal traffic from denial of service (DoS) and probe attacks. Once the data arrive on the platform, the tool enriches these data with associated metadata to improve the detection process. Upon a threat detection, the system is able to promptly react and create blocking firewall rules. Our prototype demonstration shows the reaction and blocking of malicious flows in real time through the injection of traffic from a telecommunications operator. The system has a friendly graphical interface that provides a real-time visualization of the parameters and the attacks that occur in the network.*

Resumo. *A detecção tardia de ameaças à segurança causa um aumento significativo no risco de danos irreparáveis e restringe qualquer tentativa de defesa. Neste artigo, é proposta a ferramenta CATRACA, um sistema eficiente de detecção e prevenção de intrusão em tempo real como uma função virtualizada de rede. A ferramenta baseia-se em um sistema de processamento por fluxos para grandes massas de dados, o Apache Spark, e é implementada na Open Platform for Network Functions Virtualization (OPNFV), fornecendo um serviço eficiente de detecção de ameaças em tempo real e sobre a base de dados histórica. O processo de detecção inclui algoritmos de seleção de características e aprendizado de máquina para diferenciar o tráfego normal dos ataques de negação de serviço e varredura de porta. Uma vez que os dados chegam na plataforma, a ferramenta enriquece os dados com metadados associados para melhorar o processo de detecção. Assim que uma ameaça é detectada, o sistema é capaz de reagir rapidamente e enviar regras de bloqueio para fazer um encadeamento com firewalls. A demonstração da ferramenta mostra a reação e o bloqueio de fluxos maliciosos em tempo real através da injeção de tráfego de uma operadora de telecomunicações. O sistema possui uma interface gráfica amigável que fornece uma visualização em tempo real dos parâmetros e dos ataques que ocorrem na rede.*

1. Introdução

O monitoramento de tráfego é fundamental para manter a estabilidade, a confiabilidade e a segurança das redes de computadores [Hu et al. 2015]. O monitoramento de rede se estende

*Este trabalho foi realizado com recursos do CNPQ, da CAPES, da FAPERJ, e da FAPESP (2015/24514-9, 2015/24485-9, E 2014/50937-1).

desde uma simples coleta de estatísticas até uma complexa análise de tráfego de camadas superiores para ajuste de desempenho de rede e depuração de protocolos. As ferramentas atuais de monitoramento de rede, como NetFlow, SNMP, Bro, ou Snort, não atendem às necessidades atuais de velocidade e gerenciamento de grandes domínios de rede, precisando de *hardware* externo para satisfazer essas necessidades. Além disso, muitas dessas ferramentas geram uma enorme quantidade de arquivos que requerem processamento de outro tipo de ferramentas para extrair conhecimento dos dados coletados. Assim, os atuais sistemas de segurança, como o *Security Information and Event Management* (SIEM), não possuem um desempenho satisfatório, enquanto 82% das ameaças de segurança ocorrem em minutos, uma intrusão pode levar até 8 meses para ser detectada [Verizon Enterprise 2016]. É essencial que o tempo de detecção seja o mínimo possível para que a prevenção da intrusão seja eficaz [Wu et al. 2014].

Os ataques de segurança vêm se aperfeiçoando e a simples análise e filtragem de pacotes pelo cabeçalho IP e porta TCP deixaram de ser efetivas, porque o tráfego atacante procura se esconder das ferramentas de segurança falsificando o IP de origem e alterando dinamicamente a porta. Nesse contexto, uma alternativa promissora para classificar tráfegos e detectar ameaças é o uso de técnicas de aprendizado de máquina. Essas técnicas são adequadas para grandes massas de dados, pois, com mais amostras para treinar o classificador, os métodos tendem a terem maior acurácia [Mayhew et al. 2015]. No entanto, com grandes quantidades de dados, os métodos de aprendizado de máquina podem apresentar altas latência devido ao longo tempo de processamento. Essa alta latência é uma restrição para métodos de aprendizado de máquina que devem analisar os dados e detectar ameaças o mais rápido possível. Neste contexto, o processamento de fluxo em tempo real permite a análise imediata de diferentes tipos de dados e, consequentemente, beneficiam o monitoramento de tráfego para detecção de ameaças de segurança. Plataformas de processamento distribuído de fluxo de código aberto vêm recentemente sendo propostas para processar grandes massas de dados em fluxos com baixa latência. Para tanto, essas plataformas são a base para o desenvolvimento de aplicativos sob-medida para cada caso.

Este artigo propõe a ferramenta CATRACA[1] que utiliza a tecnologia de Virtualização de Funções de Rede (*Network Function Virtualization* - NFV) e que possui uma infraestrutura que combina virtualização, computação em nuvem e processamento de fluxo distribuído para monitorar o tráfego de rede e detectar ameaças. O objetivo é fornecer uma ferramenta de detecção de ameaças precisa, escalável e em tempo real, detecção de ameaças em tempo real de forma precisa e escalável, capaz de atender picos de uso, fornecendo uma alta qualidade de serviço. O monitoramento de tráfego e a detecção de ameaças como uma função de rede virtualizada apresentam duas vantagens principais: a capacidade de se autoadaptar a diferentes volumes de tráfego e a flexibilidade de instalação e migração de sensores na rede para reduzir a latência no monitoramento [Andreoni Lopez et al. 2014]. A ferramenta captura e processa grandes volumes de dados, as técnicas de aprendizado de máquinas classificam o tráfego em normal ou ameaça. Por fim, o conhecimento extraído dos fluxos é apresentado em uma interface web.

O restante do artigo está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 discute a arquitetura e o funcionamento da ferramenta. A Seção 4 descreve a demonstração da ferramenta a ser realizada. Por fim, a Seção 5 conclui o trabalho.

2. Trabalhos Relacionados

Existem algumas propostas que utilizam a ferramenta de processamento de fluxo *Storm* para realizar a detecção de anomalias em tempo real. Du *et al.* utilizam a ferramenta *Flume* e

[1] A ferramenta, a documentação e a licença estão disponíveis em https://gta.ufrj.br/catraca.

Storm para fazer o monitoramento de tráfego a fim de detectar anomalias. A proposta é fazer a detecção através do algoritmo de k-NN [Du et al. 2014]. O artigo apresenta alguns resultados de desempenho, mas carece de avaliação da acurácia da detecção e a ferramenta não recebe dados de múltiplas fontes. O trabalho de Zhao *et al.* usa os *softwares Kafka* e *Storm*, assim como o trabalho anterior, para a detecção de anomalias em redes [Zhao et al. 2015],caracterizando os fluxos no formato *NetFlow*. He *et al.* propõem uma combinação das plataformas de processamento distribuído *Hadoop* e *Storm*, em tempo real, para a detecção de anomalias. Nessa proposta, uma variante do algoritmo k-NN é usada como algoritmo de detecção de anomalias [He et al. 2015]. Os resultados mostram um bom desempenho em tempo real, porém sem utilizar nenhum processo de reação e prevenção de ameaças. Mylavarapu *et al.* propõem usar o *Storm* como plataforma de processamento de fluxos na detecção de intrusão [Mylavarapu et al. 2015].

A proposta Stream4Flow utiliza o Apache Spark com a pilha ElasticStack para fazer monitoramento de redes [Jirsik et al. 2017]. O protótipo serve como visualização dos parâmetros da rede. Stream4Flow, no entanto, não possui nenhuma inteligência para realizar detecção de anomalias. Dos Santos *et al.* utilizam uma combinação do Snort IDS e OpenFlow para criar o Of-IDPS. O Snort IDS é usado como uma ferramenta de detecção, enquanto as ações do OpenFlow realizam a mitigação ou a prevenção de ataques detectados [Santos et al. 2014]. Uma evolução do Of-IDPS foi proposta para desenvolver um sistema de computação autônoma para criar automaticamente regras de segurança em comutadores de Rede Definida por Software (SDN) [dos Santos et al. 2016]. As regras são criadas aplicando um algoritmo de aprendizado de máquina para alertas do Snort IDS e *logs* do OpenFlow.

O projeto OpenSOC: The Open Security Operations Center [Santos 2015] é um projeto de desenvolvimento colaborativo que integra diversos *softwares* de código aberto que objetiva uma ferramenta de análise de segurança extensível e escalável. Assim, o OpenSOC é um arcabouço de segurança analítica para monitorar grandes massas de dados usando processamento distribuído de fluxos. O OpenSOC foi descontinuado e deu origem ao projeto Apache Metron [Apache Software Foundation 2017] que é uma evolução do OpenSOC e propõe uma nova arquitetura que visa facilitar a adição de novas fontes de dados, e explorar melhor o paralelismo da ferramenta Storm.

A ferramenta CATRACA proposta, assim como o Metron, também foi inspirada no OpenSOC e objetiva a monitoração de grandes volumes de dados usando processamento de fluxo. A ferramenta CATRACA é implementada como uma função virtualizada de rede (VNF) no ambiente Open Plataform for Network Function Virtualization (OPNFV). CATRACA foca em técnicas de captura dos pacotes em tempo real, de seleção de características, de aprendizado de máquina e possui um mecanismo de ação para bloqueio imediato de fluxos considerados maliciosos. Assim, a ferramenta CATRACA atua como uma função virtualizada de rede de detecção e prevenção de intrusão que reportam resumos de fluxos e que pode ser encadeada com outras funções virtualizadas de rede conforme definido nos padrões de encadeamento de funções de rede (*Service Function Chainning* - SFC) e cabeçalhos de serviço de rede (*Network Service Header* - NSH). Além disso, o enriquecimento dos dados com metadados de geolocalização dos endereços de rede, a marcação de tempo e a correlação de eventos permitem a melhor compreensão da origem e o destino dos ataques através de uma interface simples para a visualização do conhecimento inferido a partir dos dados.

3. A Arquitetura da Ferramenta CATRACA

A arquitetura da ferramenta CATRACA, mostrada na Figura 1, é composta por três camadas: captura, processamento e visualização. A **camada de captura**, é responsável pela cap-

tura dos pacotes, através do espelhamento de tráfego, pela biblioteca *libpcap*. Uma aplicação, escrita em Python e baseada no *flowtbag*[2], abstrai os pacotes em fluxos, que são definidos como uma sequência de pacotes que possuem a mesma quíntupla IP de origem, IP de destino, porta de origem, porta de destino e protocolo, durante uma janela de tempo. Ao todo, 46 características de fluxo são extraídas e publicadas em uma fila de um serviço produtor/consumidor de mensagens *Kafka*, com baixa latência, para serem consumidos pela camada de processamento.

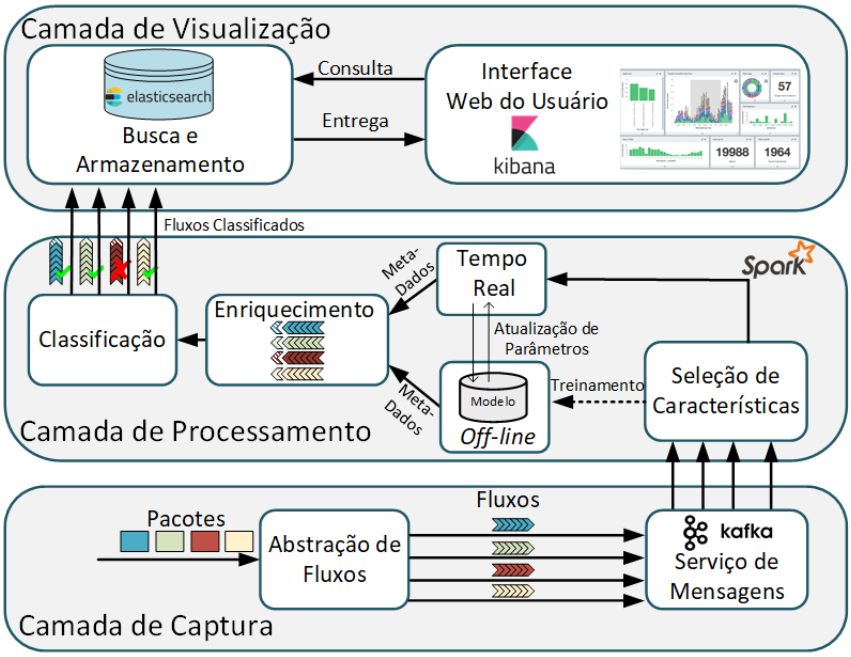

Figura 1: A arquitetura em camadas da ferramenta CATRACA: a camada de captura, a camada de processamento e a camada de visualização.

A **camada de processamento** é instanciada em uma nuvem dedicada para a classificação que possui o *Spark* como núcleo principal de processamento. A plataforma `Spark` foi escolhida entre as diferentes plataformas de processamento de fluxos por possuir o melhor comportamento em relação a tolerância a falhas [Andreoni Lopez et al. 2016], tornando a ferramenta CATRACA mais robusta em caso de falha de nós de processamento. O *Spark* é implementado em um aglomerado de máquinas no modelo mestre/escravo, em que os escravos possuem a capacidade de expansão e redução de recursos, tornando o sistema escalável. Uma vez que os fluxos chegam na camada de processamento, um algoritmo seleciona as características mais importantes para a classificação dos ataques [Andreoni Lopez et al. 2017a]. Na etapa de processamento, os dados são enriquecidos através de distintas informações como a localização geográfica dos IPs analisados. Em seguida, os fluxos são classificados em maliciosos ou bem-comportados através do algoritmo de aprendizado baseado em árvores de decisão.

Finalmente, a **camada de visualização** é implementada utilizando a pilha de *software Elastic*[3]. A pilha *Elastic* permite a visualização personalizada de eventos em tempo real. Assim, a saída da camada de processamento é enviada ao componente *elasticsearch* que fornece um rápido serviço de busca e armazenamento. A interface do usuário, que é executada no componente *Kibana* da mesma pilha de *software*, comunica-se com *elasticsearch* através de consultas.. O *Kibana* processa os resultados das consultas e gera a visualização final.

[2]Estatisticas de fluxos *flowtbag*: https://dan.arndt.ca/projects/flowtbag/
[3]*Elastic Stack*: https://www.elastic.co/

3.1. A Classificação de Tráfego Proposta

A classificação inicia com o pré-processamento de seleção das características mais importantes dos fluxos [Andreoni Lopez et al. 2017a]. Em seguida, a ferramenta pode operar em modo de tempo diferenciado ou de tempo real. A classificação de tráfego em tempo diferenciado (*off-line*) consiste no processamento de mini-lotes da plataforma *Spark*. Nesse modo, conjuntos de dados de grandes volumes são carregados em um sistema de arquivos distribuído, como por exemplo o *Hadoop Distributed File System* (HDFS). O conjunto de dados é separado em um conjunto de treinamento e um conjunto de teste em uma proporção de 70% para o treinamento e 30% para o teste. Logo, o *Spark* realiza o processamento através da técnica *map-reduce*. Um algoritmo de aprendizado de máquinas é treinado para obter o modelo de classificação.

O algoritmo de classificação por árvore de decisão é implementado na ferramenta, devido à sua velocidade de treinamento aliada à sua alta acurácia e precisão [Lobato et al. 2016]. A árvore de decisão é um algoritmo guloso que executa um particionamento binário recursivo do espaço de recursos. Cada folha é escolhida selecionando a melhor separação de um conjunto de divisões possíveis, para maximizar o ganho de informações em um nó da árvore. A divisão em cada nó da árvore é escolhida a partir do $argmax_d GI(CD, d)$, onde $GI(CD, d)$ é o ganho de informação quando uma divisão d é aplicada a um conjunto de dados CD. O ganho de informação GI da ferramenta CATRACA é a impuridade de Gini, $\sum_{i=1}^{C} f_i(1 - f_i)$, que indica o quão separadas estão as classes, em que f_i é a frequência da classe i em um nó e C é o número exclusivo de classes. Uma vez obtido, o modelo é armazenado no sistema de arquivos e carregado para ser usado no modo de classificação de tráfego em tempo real (*on-line*). No entanto, é também possível validar o modelo com o conjunto de treinamento de 30% obtido anteriormente.

A Tabela 1 mostra a matriz de confusão da avaliação do conjunto de dados de segurança [Lobato et al. 2016]. A matriz de confusão especifica claramente a taxa de falsos positivos e outras métricas de cada classe no conjunto de dados de teste. As linhas representam os elementos que de fato pertencem à classe real e as colunas os elementos que foram classificados como pertencentes à classe. Portanto, os elementos na diagonal em destaque dessa matriz representam o número de elementos que são corretamente classificados. A Tabela 2 mostra as

Tabela 1: Matriz de Confusão na Árvore de Decisão.

	Normal	**DoS**	**Varredura**
Normal	64134	0	1550
DoS	0	2557067	0
Varredura	13	0	31254

Tabela 2: Métricas de Avaliação *Off-line*

Métrica	**Acurácia**	**Erro**	**Precisão**	***Recall***	**F1 *Score***
Valor	99.94%	0.0005 %	99.94%	99.94%	99.94%

métricas complementares à matriz de confusão. Observando os valores de Acurácia e Precisão é possível ver o bom desempenho do algoritmo de árvores de decisão na classificação *off-line*. Comparando-se as Tabelas 1 e 2, verifica-se que o algoritmo apresentou uma alta acurácia praticamente em todas as classes, com uma baixa taxa de falsos positivos. Outra forma de ver a taxa de falsos positivos é observar os valores que estão fora da diagonal principal na Tabela 1.

O treinamento do sistema é realizado a partir de um conjunto de dados criado no laboratório GTA/UFRJ, contendo sete tipos de negação de serviço (DoS) e nove tipos de varredura de portas [Lobato et al. 2016]. O conjunto de dados contém cerca de 95 GB de dados de captura de pacotes, resultando em 154.187 fluxos.

Após a obtenção do modelo de classificação a partir da base histórica, pode-se avaliar a acurácia da ferramenta com dados chegando em tempo real. A operação da ferramenta CATRACA em tempo real utiliza o módulo *streaming* da plataforma *Spark*. Assim, os pacotes abstraídos em fluxos, capturados em diferentes máquinas virtuais na nuvem, são processados conforme chegam à plataforma *Spark*. Quando um fluxo chega na ferramenta de detecção, é resumido em características utilizando o algoritmo de seleção [Andreoni Lopez et al. 2017a], a fim de reduzir o tempo de processamento. Assim, o vetor de características selecionadas é avaliado no modelo obtido no processamento *off-line*. Após a extração dos dados analíticos dos fluxos, os resultados são armazenados em um banco de dados para uma análise posterior. Os dados armazenados possuem as informações coletadas durante a detecção de ameaças e podem ser reprocessados *offline* para calcular os parâmetros a serem usados no modelo em tempo real. Para tornar o sistema mais preciso, há uma retroalimentação, uma vez que os parâmetros calculados *offline* com dados históricos ajustam o modelo de processamento para a detecção de ameaças em tempo real.

3.2. A Visualização em Tempo Real dos Dados Enriquecidos

A visualização dos dados enriquecidos ocorre através de uma interface *web* simples e amigável para permitir ao usuário monitorar as diferentes configurações de parâmetros da rede em tempo real. O visualizador de código aberto *Kibana*, um componente da pilha *Elastic*, foi utilizado para o desenvolvimento da interface *web*, pois permite a visualização dos dados de forma simples e rápida aliada ao desempenho de processamento de consultas com grandes volumes de dados com baixa latência. A Figura 2 mostra alguns dos diferentes cenários que podem ser visualizados no painel de controle, como, por exemplo, as portas de destino/origem mais acessadas, os endereços IP destino mais utilizados, o tamanho médio dos fluxos nas direções de ida e volta, a quantidade de fluxos analisados, entre outros. Vale ressaltar a visualização dos ataques em andamento através de um mapas que retratam a origem, o destino e número de ocorrências. Isso é possível devido ao enriquecimento dos dados através da correlação com metadados de geolocalização no módulo de processamento. Assim, tanto os dados como as ameaças são visualizados em tempo real. Além disso, todos os dados são armazenados com estampa de tempo, permitindo o processamento dos dados através de séries temporais.

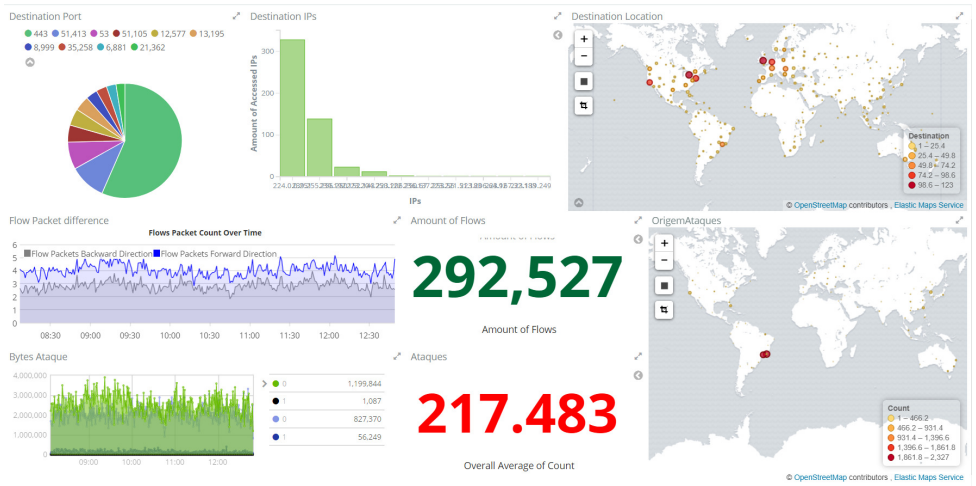

Figura 2: Visualização do painel de controle da ferramenta.

4. A Demonstração do Ferramenta CATRACA

A demonstração da ferramenta CATRACA é efetuada pela análise e a classificação em tempo real do tráfego que atravessa uma máquina virtual, que será executada em um computador no SBSeg[4]. A captura dos pacotes será enviada ao nó central, nó *master*, localizado na nuvem OPNFV no Grupo de Teleinformática e Automação (GTA/UFRJ), no Rio de Janeiro. O nó presente no GTA é responsável por receber os fluxos provenientes do *Kafka*, e gerenciar distribuí-los para os nós escravos do aglomerado *Spark*.

A demonstração evidencia a acurácia da ferramenta quando é capturado tráfego normal e quando é injetado tráfego malicioso através de uma interface de rede na máquina virtual monitorada. O classificador é previamente treinado com os parâmetros do modelo da árvore de decisão. Um conjunto de dados de uma operadora de telecomunicações [Andreoni Lopez et al. 2017b] da cidade do Rio de Janeiro é usado para gerar tráfego na máquina virtual. A topologia da demonstração é mostrada na Figura 3.

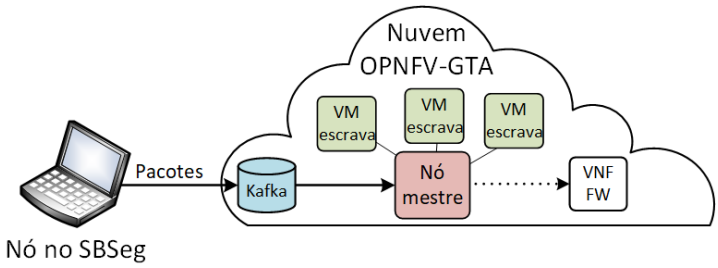

Figura 3: Topologia do experimento a ser realizado no salão de ferramentas do SBSeg.

5. Conclusão

A ferramenta CATRACA propôs uma função virtualizada de rede para detecção de intrusão em tempo real e em tempo diferenciado. A detecção de ameaças é realizada usando algoritmos de aprendizagem em máquina executados em uma plataforma de processamento de fluxo. A ferramenta proposta executa a análise de ameaças sobre o tráfego entrante em tempo real ou sobre uma base histórica em tempo diferenciado. Para aumentar a velocidade de análise e melhorar a eficiência em grandes volumes de dados, a ferramenta utiliza um algoritmo rápido, não supervisionado, para seleção das características mais importantes e, consequentemente, redução de dimensionalidade dos dados de entrada. O algoritmo de seleção de características reduz até 100 vezes do tempo de processamento, quando comparado com algoritmos tradicionais de seleção e redução de características [Andreoni Lopez et al. 2017a]. A ferramenta executa sobre plataforma de código aberto OPNFV como uma função virtual de rede e exibe o conhecimento extraído dos dados enriquecidos através de uma interface gráfica amigável para a visualização de diferentes análises e da localização geográfica da origem e destino das ameaças em tempo real.

A ferramenta pode ser obtida em `http://www.gta.ufrj.br/catraca`, onde consta o manual do usuário, que detalha os procedimentos de instalação e uso da ferramenta, a documentação, que permite compreender o projeto de *software* e obter mais detalhes sobre o código da ferramenta, além de outras informações úteis. Como trabalhos futuros, pretende-se oferecer diferentes algoritmos de aprendizado de máquina e monitorar tráfego em redes reais.

[4]A demonstração requer um computador com acesso à Internet para ser configurado com os *softwares* de teste.

Referências

Andreoni Lopez, M., Figueiredo, U. d. R., Lobato, A. G. P. e Duarte, O. C. M. B. (2014). BroFlow: Um Sistema Eficiente de Detecção e Prevenção de Intrusão em Redes Definidas por Software. *XII WPerformance (XXXIV CSBC)*, páginas 1919–1932.

Andreoni Lopez, M., Lobato, A. G. P. e Duarte, O. C. M. B. (2016). A Performance Comparison of Open-Source Stream Processing Platforms. Em *IEEE GLOBECOM*, páginas 1–6, Washington, USA. IEEE.

Andreoni Lopez, M., Lobato, A. G. P., Mattos, D. M. F., Alvarenga, I. D., Duarte, O. C. M. B. e Pujolle, G. (2017a). Um Algoritmo Não Supervisionado e Rápido para Seleção de Características em Classificação de Tráfego. Em *XXXV SBRC'2017*, Belém- Pará, PA,.

Andreoni Lopez, M., Silva, R., Alvarenga, I., Mattos, D. e Duarte, O. C. M. B. (2017b). Coleta e Caracterização de um Conjunto de Dados de Tráfego Real de Redes de Acesso em Banda Larga. Em *XXII WGRS'17*.

Apache Software Foundation (2017). Apache Metron. `https://cwiki.apache.org/confluence/display/METRON/About+Metron`. Acessado em 29/08/2017.

dos Santos, L. A. F., Campiolo, R., Monteverde, W. A. e Batista, D. M. (2016). Abordagem autonômica para mitigar ciberataques em LANs. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2016*.

Du, Y., Liu, J., Liu, F. e Chen, L. (2014). A real-time anomalies detection system based on streaming technology. Em *Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, volume 2, páginas 275–279. IEEE.

He, G., Tan, C., Yu, D. e Wu, X. (2015). A real-time network traffic anomaly detection system based on storm. Em *Proceedings - 2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics, IHMSC 2015*, volume 1, páginas 153–156.

Hu, P., Li, H., Fu, H., Cansever, D. e Mohapatra, P. (2015). Dynamic defense strategy against advanced persistent threat with insiders. Em *IEEE Conference on Computer Communications (INFOCOM)*, páginas 747–755.

Jirsik, T., Cermak, M., Tovarnak, D. e Celeda, P. (2017). Toward Stream-Based IP Flow Analysis. *IEEE Communications Magazine*, 55(7):70–76.

Lobato, A. G. P., Andreoni Lopez, M. e Duarte, O. C. M. B. (2016). Um Sistema Acurado de Detecção de Ameaças em Tempo Real por Processamento de Fluxos. Em *SBRC'2016*, páginas 572–585, Salvador, Bahia.

Mayhew, M., Atighetchi, M., Adler, A. e Greenstadt, R. (2015). Use of machine learning in big data analytics for insider threat detection. Em *IEEE MILCOM*, páginas 915–922.

Mylavarapu, G., Thomas, J. e TK, A. K. (2015). Real-Time Hybrid Intrusion Detection System Using Apache Storm. Em *17th International Conference on High Performance Computing and Communications*, páginas 1436–1441. IEEE.

Santos, L. A. F., Campiolo, R. e Batista, D. M. (2014). Uma Arquitetura Autonômica para Detecçao e Reação a Ameaças de Segurança em Redes de Computadores. Em *III WoSiDA'14*, páginas 1–4.

Santos, O. (2015). Big data analytics and netflow. Em *Network Security with NetFlow and IPFIX: Big Data Analytics for Information Security*. Acessado em 29/08/2017.

Verizon Enterprise (2016). 2016 Data Breach Investigations Report. http://www.verizonenterprise.com/resources/reports/rp_DBIR_2016_Report_en_xg.pdf.

Wu, K., Zhang, K., Fan, W., Edwards, A. e Yu, P. S. (2014). RS-Forest: A Rapid Density Estimator for Streaming Anomaly Detection. Em *IEEE International Conference on Data Mining (ICDM)*, páginas 600–609.

Zhao, S., Chandrashekar, M., Lee, Y. e Medhi, D. (2015). Real-time network anomaly detection system using machine learning. Em *11th International Conference on the Design of Reliable Communication Networks (DRCN)*, páginas 267–270. IEEE.