UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Alexsander Haas

UM SISTEMA POR PROCESSAMENTO DE FLUXOS APLICADO À ANÁLISE E MONITORAMENTO DA REDE

Santa Maria, RS
2019

Alexsander Haas

UM SISTEMA POR PROCESSAMENTO DE FLUXOS APLICADO À ANÁLISE E MONITORAMENTO DA REDE

Dissertação apresentada ao Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Maria (UFSM), como requisito parcial para a obtenção do título de **Mestre em Ciência da Computação**.

Orientador: Prof. Dr. João Vicente Ferreira Lima

Santa Maria, RS
2019

Haas, Alexsander
UM SISTEMA POR PROCESSAMENTO DE FLUXOS APLICADO À ANÁLISE E MONITORAMENTO DA REDE / Alexsander Haas.- 2019.
92 p.; 30 cm

Orientador: João Vicente Ferreira Lima
Dissertação (mestrado) - Universidade Federal de Santa Maria, Centro de Tecnologia, Programa de Pós-Graduação em Ciência da Computação , RS, 2019

1. Arquitetura Lambda 2. Tráfego de rede 3. Detecção de ataques 4. Big Data I. Vicente Ferreira Lima, João II. Título.

Sistema de geração automática de ficha catalográfica da UFSM. Dados fornecidos pelo autor(a). Sob supervisão da Direção da Divisão de Processos Técnicos da Biblioteca Central. Bibliotecária responsável Paula Schoenfeldt Patta CRB 10/1728.

Alexsander Haas

UM SISTEMA POR PROCESSAMENTO DE FLUXOS APLICADO À ANÁLISE E MONITORAMENTO DA REDE

Dissertação apresentada ao Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Maria (UFSM), como requisito parcial para a obtenção do título de **Mestre em Ciência da Computação**.

Aprovado em 25 de março de 2019:

João Vicente Ferreira Lima, Dr. (UFSM)
(Presidente/Orientador)

Benhur de Oliveira Stein, Dr. (UFSM)

Guilherme Piêgas Koslovski, Dr. (UDESC) (Videoconferência)

Santa Maria, RS
2019

AGRADECIMENTOS

A concretização deste trabalho ocorreu, principalmente, pelo apoio e compreensão de algumas pessoas. Agradeço a todos que, de alguma forma, contribuíram para a conclusão deste estudo e, de maneira especial, agradeço:

A minha namorada Cyndi Jamile Canalles de Brito pelo apoio, suporte, paciência e companheirismo durante estes dois anos de estudo, com a qual foi possível concluir mais esta etapa. Aos meus pais Vanda Beatriz Szelbracikowski e Luiz Alberto Haas, e minha irmã Alessandra Haas, os quais sempre me apoiaram e incentivaram em direção aos estudos.

Ao meu orientador, professor Dr. João Vicente Ferreira Lima, por aceitar orientar-me no Mestrado, pelos ensinamentos, paciência e dedicação estando sempre disposto a ajudar-me durante essa jornada.

À empresa Meta e ao José Luiz Dias da Costa Lyra, pelo incentivo, apoio e a flexibilidade dos horários para ser possível conciliar a jornada de trabalho com os estudos, durante esses dois anos.

Por fim, aos meus colegas, amigos e professores que de alguma forma fizeram parte desta jornada, seja por compartilhar conhecimento, dicas ou apoio.

RESUMO

UM SISTEMA POR PROCESSAMENTO DE FLUXOS APLICADO À ANÁLISE E MONITORAMENTO DA REDE

AUTOR: Alexsander Haas
ORIENTADOR: João Vicente Ferreira Lima

Processamentos de dados são comumente realizados por sistemas Big Data, em arquiteturas tradicionais, efetuando a manipulação dos dados de forma *offline*. No entanto, com a necessidade de obter resultados com baixa latência, ocorre o uso de outras arquiteturas, como Lambda e Kappa para implementação de sistemas Big Data, direcionadas ao processamento de fluxos de dados. Diversos estudos na literatura começam a aplicar esse novo modelo de arquitetura para diferentes fins, assim como a utilização de diferentes tipos de ferramentas para que seja possível efetuar sua implementação. Neste cenário alguns sistemas são desenvolvidos com estes moldes para monitorar e processar o fluxo de dados gerados pelo tráfego de rede, empregando diferentes tipos de análises sobre os dados coletados, para obter desde informações sobre o consumo de banda da rede a identificar anomalias que ocorrem. Nesse contexto, este trabalho tem como objetivo o desenvolvimento de um sistema com base na arquitetura Lambda, aplicado ao monitoramento e processamento do fluxo de dados do tráfego de rede, realizando a integração de diferentes ferramentas de código aberto. Cada ferramenta é responsável por determinada funcionalidade implementada, desde o monitoramento e coleta do tráfego de rede, transporte de informações, normalização e armazenamento dos dados, para posteriormente efetuar análises dos mesmos e detectar anomalias originadas por ataques DDoS, força bruta e varredura de portas sobre determinados protocolos. Sobre as conexões classificadas como anômalas, obter-se-á informações pertinentes ao IP responsável por originar essa conexão. A análise experimental do sistema ocorre com o uso de um conjunto de dados controlado que possui diversas anomalias, bem como as que devem ser detectadas pelo sistema. Logo após essa etapa, o sistema é aplicado para processar dados que foram coletados de uma rede local durante onze dias, totalizando mais de quatorze milhões de conexões. Os resultados experimentais obtidos sobre o tráfego de rede real apresentam os três tipos de anomalias que foram consideradas nesse estudo, assim como traz informações sobre os IPs responsáveis pelas mesmas, identificando o país e sua respectiva organização.

Palavras-chave: Arquitetura Lambda. Big Data. Tráfego de rede. Detecção de ataques.

ABSTRACT

A FLOW PROCESSING SYSTEM APPLIED TO NETWORK ANALYSIS AND MONITORING

AUTHOR: Alexsander Haas
ADVISOR: João Vicente Ferreira Lima

Data processing is commonly performed by big data systems, in traditional architectures, performing data manipulation offline. However, with the need to get results with low latency, there is the use of other architectures, such as LAMBDA and Kappa for the implementation of big data systems, directed to the processing of data streams. Several studies in the literature begin to apply this new model of architecture for different purposes, as well as the use of different types of tools to make it possible to implement it. In This scenario some systems are developed with these molds to monitor and process the flow of data generated by network traffic, employing different types of analysis on the collected data, to get from information about network bandwidth consumption to identify anomalies that occur. In this context, this work aims to develop a system based on the Lambda architecture, applied to the monitoring and processing of the data flow of network traffic, performing the integration of different open source tools. Each tool is responsible for certain functionality implemented, from monitoring and collecting network traffic, information transport, normalization and data storage, to subsequently perform analyses thereof and detect anomalies originated by DDoS attacks, brute force, and port scanning on certain protocols. Regarding the connections classified as anomalous, information pertinent to the IP responsible for originating this connection will be obtained. The experimental analysis of the system occurs with the use of a controlled set of data that has several anomalies, as well as those that must be detected by the system. Shortly after this step, the system is applied to process data that was collected from a local network for eleven days, totaling more than 14 million connections. The experimental results obtained on the actual network traffic present the three types of anomalies that were considered in this study, as well as information about the IPs responsible for them, identifying the country and its respective organization.

Keywords: Architecture Lambda. Big data. Network Traffic. Detection of attacks.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ASCII	American Standard Code for Information Interchange
BR	Brasil
BRO	The Bro Network Security Monitor
CA	Canadá
CN	China
DCE/RPC	Distributed Computing Environment/Remote Procedure Calls
DDoS	Distributed Denial of Service
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DoS	Denial of Service
DStream	Discretized Stream
DTLS	Datagram Transport Layer Security
EG	Egito
ELM/GRBFK	Extreme Learning Machine neural network with Gaussian Radial Basis Function kernel
FFSc	Feature Feature Score
FR	França
FTP	File Transfer Protocol
GB	Gigabyte
GB	Reino Unido
GHz	Gigahertz
HD	Disco Rígido
HDFS	Hadoop Distributed File System
HK	Hong Kong
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IES	Instituição de Ensino Superior
IOT	Internet of Things
IP	Internet Protocol
IRC	Internet Relay Chat
JAR	Java ARchive
JBDC	Java Database Connectivity
JSON	JavaScript Object Notation
KRB	Kerberos
MB	Megabyte
MLlib	Machine Learning Library
NBKE	Naive Bayes Kernel Estimator
NL	Holanda
NTLM	NT LAN Manager
NTP	Network Time Protocol
OBDC	Open Database Connectivity
OPNFV	Open Source Platform for Network Functions Virtualization
PCAP	Packet Capture
RAM	Random Access Memory
RDD	Resilient Distributed Datasets
RO	Romênia

SA	Arábia Saudita
SAM	Self Adjusting Memory
SARIMA	Seasonal Autoregressive Integrated Moving Average
SC	Seicheles
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SQL	Structured Query Language
SSH	Secure Shell
SSL	Secure Sockets Layer
SVM	Suport Vector Machine
TB	Terabyte
TCP	Transmission Control Protocol
TEDA	Typicality and Eccentricity Data Analysis
TLS	Transport Layer Security
UDP	User Datagram Protocol
UID	User Identifier
UFSM	Universidade Federal de Santa Maria
UNB	University of New Brunswick
URL	Uniform Resource Locator
US	Estados Unidos
VM	Virtual Machine
WAL	Writed Ahed Log
XMPP	Extensible Messaging and Presence Protocol
YARN	Yet Another Resource Negotiator
ZEEK	The Zeek Network Security Monitor

SUMÁRIO

1 INTRODUÇÃO

Com a rápida evolução da internet ocorre o aumento do volume, velocidade e variedade de dados gerados e transmitidos pela rede, demandando uma infraestrutura robusta para realizar o monitoramento e processamento destas informações, com altas taxas de respostas e de forma eficiente, esse é um grande desafio atualmente. O avanço da internet das coisas (IoT) é um dos principais responsáveis por esse acréscimo de dados gerados e esse crescimento ocorre de forma acelerada, pois de 2 bilhões de objetos em 2006 para uma projeção de 200 bilhões em 2020[1]. Ocorrem muitos acessos à internet, seja por meio de Websites, mídias socias ou dispositivo (IoT), isso acaba gerando uma enorme quantidade de dados continuamente, assim uma grande quantidade de informações é concebida em forma de fluxo.

Surge, deste modo, o conceito *Big Data*, quando o volume de dados excede a capacidade de processamento dos sistemas de banco de dados convencionais, sendo necessário adotar maneiras alternativas de processá-los (DUMBILL, 2012). Portanto, os sistemas implementados sobre esse conceito de *Big Data*, são utilizados para realizar análise de grandes conjuntos de dados, com objetivo de obter informações relevantes sobre os mesmos, aplicando diferentes métodos de processamento com diversas finalidades. Um âmbito de aplicação destes sistemas é para efetuar o monitoramento do tráfego de rede, com objetivos diversificados, tais como o processamento dos dados para detecção de ataques e anomalias na rede, previsão de consumo da largura de banda de rede, identificar aplicações e seu consumo, descobrir acessos indevidos, entre outras finalidades.

Alguns dos meios frequentemente implementados com o intuito de fazer essa análise, realizam o monitoramento de rede por um determinado período, através de ferramentas capazes de gerar arquivos de *logs* com informações sobre o tráfego de rede deste período monitorado. Esses arquivos são coletados e armazenados em banco de dados ou sistemas de arquivos, com o propósito de disponibilizá-los para serem acessados e processados posteriormente. Um modelo de sistema que realiza esses passos, utiliza algumas ferramentas para captura de dados, tais como NetMap, Wireshark e The Zeek Network Security Monitor (ZEEK).

Para as etapas de armazenamento e processamento, o framework Hadoop, em conjunto com o MapReduce, é considerado um dos mais eficientes (OUNACER et al. 2017), por possuir um desempenho proporcional a complexidade de uma grande quantidade de dados. Uma de

[1] IDC, Intel, United Nations: https://www.intel.com/content/www/us/en/internet-of-things/infographics/guide-to-iot.html

suas características é realizar o processamento dos dados em *batch*, sendo esse mesmo ponto sua limitação, pois não consegue trabalhar com o que acontece em tempo real, ou seja, realizar o processamento dos dados em forma de fluxo, trabalhando apenas com os novos dados.

O problema deste modelo de sistema é que depende do processamento em *batch,* que fornece bons resultados sobre o que ocorreu no passado (TALHAOUI et al. 2017), contudo não possui a mesma eficiência para processar e obter informações sobre os dados recentes. Conforme mencionado anteriormente, esse modelo é baseado por exemplo, no *framework* Hadoop em conjunto MapReduce, que atua de forma distribuída para processar grandes quantidades de dados, ou seja, um conjunto finito de dados. Essas ferramentas não são adequadas para trabalhar com um fluxo de dados, sendo impróprio para disponibilizar resultados com baixa latência. Com objetivo de apresentar soluções a esse problema, novas arquiteturas foram propostas, conforme o conceito de arquitetura Lambda abordada por Marz (2011) que é composta por três camadas, *batch*, *speed* e *serving*.

Os sistemas atuais estão utilizando arquitetura Lambda, pois ela tem como objetivo unificar o processamento em lote (*batch layer*), que armazena e processa todos os dados, com o processo de velocidade (*speed layer*), que processa apenas os novos dados conforme eles chegam, com isso os resultados de ambos processos são exibidos (*serving layer*). Outro conceito de arquitetura que está sendo aplicado é a arquitetura Kappa, abordada por Kreps (2014) inspirada na arquitetura Lambda, sendo uma alternativa de implementação simplificada, pois possui apenas duas camadas *speed* e *serving*, não efetuando o armazenamento de dados para realizar algum processamento em *batch*, dedicando-se a processar dados em tempo real.

Implementações que utilizam alguma destas arquiteturas, possuem como principal característica o processamento de fluxo de informações com baixa latência, conforme abordado por Persico et al. (2018) que realizou comparação de performance entre estas duas arquiteturas, aplicadas ao processamento de dados de mídias socias. Para o desenvolvimento destes sistemas podem ser utilizadas diversas ferramentas, cada uma executando um processo específico e realizando a integração entre as demais. Para aplicação em *speed layer* podem ser destacadas algumas ferramentas de código aberto, que processam dados em um fluxo continuo segundo Gurusamy et al. (2017), Apache Storm, Apache Spark, Apache Samza e Apache Flink.

1.1 OBJETIVO

O objetivo principal deste trabalho é desenvolver um sistema baseado na arquitetura Lambda, capaz de processar e analisar o fluxo de dados do tráfego de rede, com o propósito de

identificar anomalias de rede causadas por ataques de *Distributed Denial of Service* (DDoS), força bruta e varredura de portas, sobre os protocolos *Hypertext Transfer Protocol* (HTTP), *Secure Shell* (SSH) e *Transmission Control Protocol* (TCP) respectivamente. A partir dos resultados, obter informações de organização e país sobre o *Internet Protocol* (IP), responsáveis por estas práticas.

Esse trabalho realiza uma análise experimental do sistema sobre um conjunto de dados controlados e, posteriormente, efetua o processamento de um fluxo de dados real, coletado de uma rede local da UFSM, alcançando assim resultados sobre um conjunto de dados não controlados.

1.2 ORGANIZAÇÃO DO TEXTO

Este trabalho é composto por mais quatro capítulos, que estão organizados da seguinte forma. No Capítulo 2 apresentam-se anomalias presentes no tráfego de rede e as possíveis análises a serem aplicadas, conceitos e definições sobre *Big Data*, assim como algumas possíveis arquiteturas de processamento, e por fim, aborda alguns trabalhos relacionados que desenvolveram sistemas e aplicaram diferentes métodos para o processamento dos dados da rede.

No Capítulo 3 é descrito o sistema proposto, a arquitetura utilizada e sua estrutura lógica, o detalhamento da integração realizada entre as ferramentas selecionadas e os métodos de processamentos implementados, bem como as análises que serão aplicadas sobre os dados. O Capítulo 4 traz o ambiente no qual o sistema desenvolvido foi implementado e os recursos de *hardware* utilizados, também detalha os conjuntos de dados utilizados e realiza uma análise sobre os resultados obtidos de cada conjunto. No Capítulo 5 apresentam-se as conclusões e os trabalhos a serem desenvolvidos no futuro.

2 PROCESSAMENTO DO FLUXO DE DADOS DA REDE

Este capítulo apresenta conceitos de análise e processamento do fluxo de dados da rede, bem como a arquitetura a ser utilizada para esse objetivo. Para isso, é realizada inicialmente uma abordagem sobre a análise e anomalias da rede na seção 2.1. Na sequência, a seção 2.2 destaca as características e conceitos sobre *Big Data* e seus principais desafios, seguindo a linha de *Big Data,* a seção 2.3 aborda as arquiteturas de processamento que são utilizadas na implementação de sistemas de *Big Data* para análise de dados, bem como as características apresentadas por cada uma. A última seção 2.4 enumera os trabalhos relacionados na área de análise do fluxo de dados da rede, com foco em arquiteturas para processamento em tempo real, descrevendo algumas de suas ferramentas, arquiteturas, algoritmos e resultados obtidos em seus estudos.

2.1 ANÁLISE E ANOMALIAS DA REDE

Análises e processamentos sobre os dados da rede têm como objetivo o monitoramento e detecção de anomalias. As anomalias podem ser definidas como agrupamento de dados que não se adequam ao comportamento padrão do conjunto de dados, ou seja, referem-se à existência de padrões detectáveis do tráfego de rede no qual ocorrem desvios em seu comportamento (CHANDOLA, et al., 2009). Essas anomalias podem surgir por uma variedade de razões como atividades de intrusões cibernéticas, queda do sistema, erros de configurações, atividades de *softwares* maliciosos, falhas de *hardware*, e etc. (SILVA, 2016; SAHNI e SHARMA, 2013). Chandola et al. (2009) classificam as anomalias em três tipos:

- Pontuais: Quando dados individuais se desviam dos demais;
- Contextuais: Considera determinado dado anômalo em um contexto, mas não em outro;
- Coletivo: Um grupo de dados é anômalo em comparação a todo o conjunto de dados;

Ataques a Rede

Dentro destas anomalias podemos destacar algumas que são causadas por ataques, como os *Denial of Service* (DoS) que são comportamentos maliciosos iniciados por uma pessoa ou organização com o objetivo de deixar indisponível serviços online (ZHANG, et al., 2017), essa indisponibilidade ocorre ao esgotar os recursos que existem na máquina da vítima (WANG et al., 2015). Os ataques DDoS consistem no envio de diferentes pacotes de dados originados por diversas máquinas, denominadas *Bots,* para um único destino, geralmente essas máquinas são

controladas de modo remoto por outro computador o *Master*, formando assim um grupo interligado de máquinas nomeado *Botnet*, que executam de forma coordenada ações maliciosas definidas pela *Master* (WELZEL, et al., 2014), com objetivo de sobrecarregar e travar o sistema da vítima tornando-o inacessível.

Com as características descritas acima, podemos considerar o cenário proposto neste estudo para identificação de ataques DDoS sobre o protocolo HTTP, para apresentar o fator que será considerado anomalia. Em determinadas variações destes ataques, a vítima sofre um aumento repentino do número de conexões recebidas pelo protocolo HTTP, causadas por determinados IPs que efetuam um número significativo de conexões. Desta forma, a característica a ser observada para identificar esses ataques e classificá-los como anomalias, é o número de conexões originadas por um mesmo IP com o uso de diversas portas de origem, em direção a um mesmo IP de destino e porta.

Varredura de porta (*Scan Port)* é um ataque que gera anomalias na rede, pois é realizado para detectar qual porta da rede está aberta em determinada máquina, para isso é encaminhada uma mensagem para cada porta de forma individual para posterior análise das repostas, a fim de obter informações sobre quais portas estão sendo utilizadas (PATEL e SONKER, 2016). Com estas premissas, em uma rede que sofre de varreduras de porta, ocorre um aumento significativo de conexões originadas por um mesmo IP e porta, com destino a um mesmo IP e diferentes portas, deste modo é possível classificar como anomalia conexões que apresentam estas características.

O ataque de força bruta SSH ocorre com o objetivo de obter acesso a máquina alvo por meio deste serviço, logo para que isso seja possível, são realizadas inúmeras tentativas de acesso utilizando diversas senhas (YAO, et al., 2017). Ao ser vítima deste tipo de ataque, o tráfego de rede apresenta características que são utilizadas no cenário deste trabalho, para classificar se determinadas conexões são anômalas ou não. Ao habilitar conexões SSH em uma máquina, podem ser definidos limites de tentativas de autenticações permitidas por conexão, ou seja, se quiser realizar outra tentativa, uma nova conexão tem de ser iniciada. Desta forma, para este tipo de ataque, o tráfego de rede apresenta um número significativo de conexões originadas por um mesmo IP e diferentes portas, com destino um único IP e porta, sendo essa característica o fator determinante para classificar as conexões como anômalas ou não.

Outros tipos de ataques que não são considerados para detecção no sistema desenvolvido, são abordados a seguir. IP *Spoofing* tem como finalidade mascarar o IP de origem, enviando pacotes para a vítima com o endereço IP de origem de um *host* conhecido e

confiável em vez de seu próprio IP, o *host* de destino pode vir a aceitar o pacote e agir de acordo com ele, assim o invasor consegue permanecer anônimo, efetuar ataques e burlar algumas restrições de segurança (MUKADDAM, et al., 2014).

O ataque denominado SQL *Injection* ocorre por meio de vulnerabilidades presentes em aplicativos e páginas da internet que interagem com banco de dados por comandos SQL, no qual o invasor efetua uma ação que gera uma solicitação SQL, sobre a qual podem ser inseridos comandos SQL maliciosos para obter informações do banco de dados, acessos privilegiados ou alterar o conteúdo que está armazenado (KIEYZUN, et al., 2009; SONEWAR E THOSAR, 2016) .

Técnicas e Algoritmos

Algumas técnicas aplicadas à detecção de anomalias, podem ser classificadas em supervisionada, semi-supervisionada e não supervisionada. Segundo Silva (2016) na literatura são encontrados, comumente, diversos métodos para detecção de anomalias: estatístico, baseado em classificação, agrupamento e desvios, conhecimento e combinação de classificadores. Dentro destas técnicas podemos citar alguns algoritmos definidos por Wu et al. (2008) como influentes: *K-Means, Naive Bayes e Suport Vector Machine (SVM)*.

K-means é um algoritmo de classificação que divide o conjunto de dados em diversos grupos (*clusters)* conforme definido pelo usuário, para gerar esses *clusters* o algoritmo faz uma comparação entre cada valor do conjunto de dados para calcular a distância entre os mesmos, sabendo assim o quão longe uma ocorrência esta da outra, com base nessa informação são calculados os centroides para cada um dos *clusters*. A cada iteração os valores dos centroides são refinados conforme a média dos valores de cada dado do conjunto analisado, essa operação se repete até quando não ocorrer mais diferença entre os valores dos centroides da iteração atual e da anterior. Com os centroides definidos, o algoritmo define a posição dos dados de acordo com a sua distância em relação ao centroide.

O *Naive Bayes* é um algoritmo de classificação probabilístico que parte do princípio de que as características são independentes, significando que a ocorrência de uma característica não afeta a probabilidade de ocorrência de outra. O modelo possui uma tabela de probabilidades para cada característica, definidas a partir da base de treinamento. É possível determinar a probabilidade ao se analisar o conjunto de dados de treino utilizando o teorema de *Bayes.*

O *Suport Vector Machine* tem como objetivo encontrar a melhor função de classificação entre membros de duas classes utilizando os dados de treino. O SVM calcula o hiperplano que

define uma fronteira entre as classes, ele maximiza as fronteiras que separam os dados das duas classes e a classificação ocorre ao analisar se os dados estão acima ou abaixo deste hiperplano.

Outras abordagens também são apresentadas na literatura, Liu et al. (2008) aplica *naive Bayes kernel estimator* (NBKE) para detecção de varredura de portas e utiliza o expoente de *Hurst* no mecanismo responsável pela análise e caracterização dos ataques, já Kaushik et al., (2010) para esse mesmo caso implementa um sistema forense utilizando sistema de detecção de intrusão (IDS) para capturar, marcar os dados relevantes e armazená-los para posteriormente analisá-los. O método proposto por Treurniet (2011) é capaz de detectar varreduras de portas que são executadas lentamente. Hasanifard et al. (2014) propõe um sistema que opera em paralelo utilizando IDS para detecção de anomalias, Maruhashi et al. (2011) emprega a mineração de padrões com análise tensorial, apresentando resultados empíricos que sugerem atividades suspeitas de varredura de porta e DDoS.

A taxonomia de anomalia de rede para classificação do tráfego de rede, em conjunto com rótulos de anomalias e assinaturas correspondentes, foi implementado por Mazel et al. (2014). Hoque et al. (2016) definiu e aplicou a medida estatística *Feature Feature Score* (FFSc) para análise de dados multivariado com objetivo de diferenciar o tráfego de rede que possui ataques DDoS do normal. Enquanto Martins et al. (2018) propôs a detecção de ataques aplicando o conceito de análise de dados baseada em tipicidade e excentricidade (TEDA), conceito proposto por Angelov (2014).

2.2 BIG DATA: CONCEITO E DEFINIÇÃO

Com a crescente disponibilidade e capacidade do armazenamento de dados surge o conceito *Big Data*, não havendo, contudo, uma definição única sobre esse conceito, diversas abordagens surgiram ao longo do tempo. As definições mais comumente referenciadas, convergem na ideia de que *Big Data* se apoia na característica dos dados e na limitação das tecnologias para transformar esses dados em valor (DANESH, 2014).

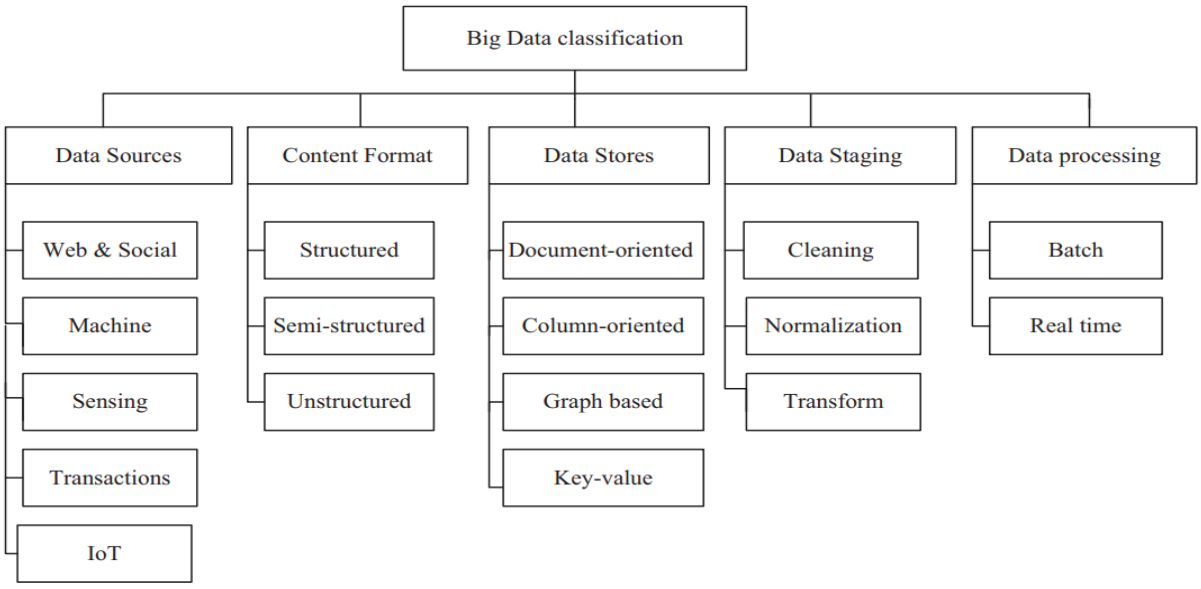
Para Schonberger et al. (2014) *Big Data* refere-se sobre o que pode ser feito em grande escala para extrair novas formas de valor que possam impactar em diversos setores, segundo Danesh (2014) é um grande volume de dados que se torna difícil de coletar, processar e armazenar por técnicas e sistemas tradicionais exigindo novas técnicas de processamento ou infraestrutura. J.J. Berman (2013) caracteriza *Big Data* por “3 V” volume, variedade e velocidade, já J. Gantz (2011) define que pode ser caracterizado pelos seguintes “Vs” volume, variedade, velocidade e valor, sendo essa última a definição amplamente reconhecida pois

destaca a necessidade e significado de *Big Data*. Segundo Storey e Song (2017) *Big Data* é reconhecido tradicionalmente por três "3 V" volume, variedade e velocidade, mas além destes veracidade e valor também são importantes, sendo definido "5 V" os desafios dominantes na prática do *Big Data* e nas pesquisas.

Nesse contexto de *Big Data* v*olume* refere-se a quantidade de dados de todos os tipos que são gerados de diferentes fontes que continua a expandir. Os diferentes tipos de dados áudio, vídeo e *logs* em formatos estruturados ou não estruturados, coletados por meio de sensores, redes sociais, IoT e entre outros é o que define a v*ariedade* (HASHEM, et al., 2015). *Velocidade* refere-se ao tempo que os dados levam para serem gerados, sendo o intervalo de tempo entre a chegada do dado e o seu processamento para a tomada de decisão sobre o seu resultado (KHAN et al., 2014). A v*eracidade* refere-se a imprecisão dos dados, destacado sobre os pontos de qualidade, confiabilidade e incerteza (STOREY E SONG, 2017). O aspecto mais importante é o *valor,* pois refere-se a descoberta de valores sobre enormes conjuntos de dados (CHEN, M., et al., 2014), *valor* não é uma característica direta do *Big Data*, mas sim o objetivo máximo do seu uso (KHAN et al., 2014).

Para Hashem et al. (2015) *Big Data* pode ser classificado em diferentes categorias conforme pode ser observado na Figura 1, que são baseadas em cindo aspectos: (a) fonte de dados, (b) formato do conteúdo, (c) armazenamento dos dados, (d) preparação dos dados, e (e) processamento dos dados.

Figura 1 - *Big Data Classification.*



Fonte: Hashem, et al., (2015).

2.3 ARQUITETURA LAMBDA

A arquitetura Lambda unifica o processamento em tempo real e em lote em uma única ferramenta, disponibilizando baixa latência e resultados melhores (OUNACER et al., 2017), segundo Bertran (2014) ela consiste nos componentes essenciais que asseguram um sistema tolerante a falhas e robusto, mantendo o desempenho, utilizando recursos de *stream* e *batch*.

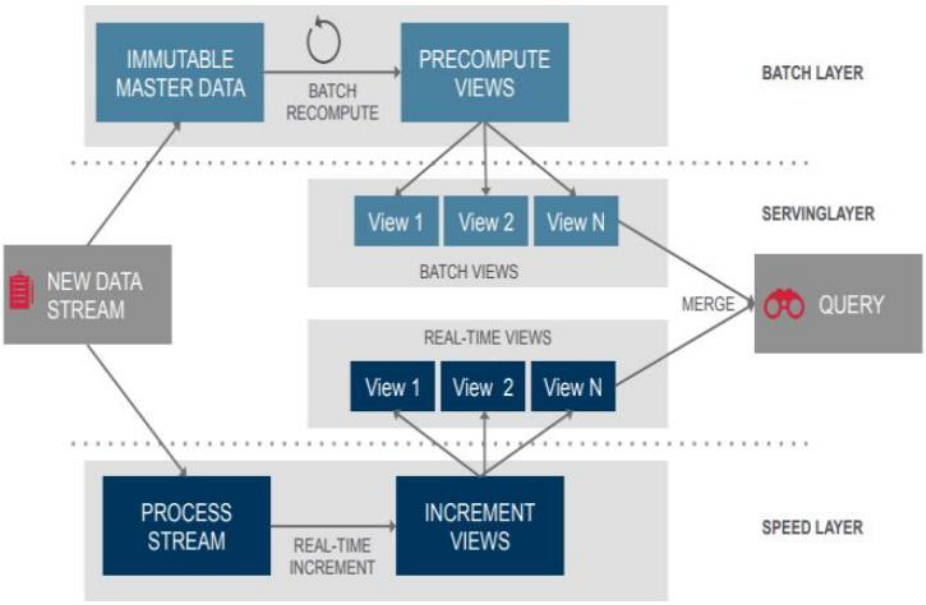
O objetivo de implementar a arquitetura Lambda é construir um sistema *Big Data* com um conjunto de camadas, onde cada umas destas camadas satisfaz um subconjunto de necessidades, assim conforme Kiran et al. (2015) adequada para aplicações em que não podem ocorrer atrasos entre a coleta dos dados e sua exibição, permitindo que os usuários definam seus custos de processamentos, decidindo quais informações precisam ser processadas em baixa latência ou em lote. Segundo Marz et al. (2015) a Lambda é formada por três camadas, conforme exibido na Figura 2 e detalhadas a seguir.

- **Batch:** ingere e armazena uma grande quantidade de dados, realizando o processamento sobre eles.
- **Speed:** processa os dados incrementando informações sobre eles, esta operação sendo realizada em uma forma de transmissão de baixa latência.
- **Serving:** tem como objetivo agregar as informações dos resultados obtidos da primeira e segunda camada, criando assim uma visão dos dados que foram processados pelas duas camadas anteriores.

O fluxo de dados enviados para arquitetura Lambda consiste em receber o mínimo de processamento possível até ser distribuído nas duas camadas de lote e velocidade. A camada de lote possui como principais responsabilidades o armazenamento dos dados de forma imutável e o processamento dos dados para gerar os resultados a serem exibidos na camada de serviço. É responsável também pelo reprocessamento dos dados, quando há o recebimento de novas informações, para serem agregadas e exibidas na camada de serviço, de modo que esse ciclo se repete conforme a quantidade de dados a serem processados.

A camada de velocidade efetua o processamento dos dados com objetivo de gerar resultados que possam ser visualizados em tempo real na camada de serviço, incrementando os resultados que são exibidos, à medida que novos dados são recebidos, sendo que esses resultados são temporários, pois, posteriormente, serão reprocessados pela camada em lote. A grande diferença da camada de velocidade, é que ela trabalha apenas com os dados recentes, isto é, conforme os mesmos são recebidos já são processados, desta forma ela não efetua o processamento de todos os novos dados de uma única vez (MARZ et al. 2015).

Figura 2 - Arquitetura Lambda.



Fonte: Ounacer et al., (2017).

Com estas características as camadas são implementadas para trabalharem em conjunto, utilizando uma coleção de ferramentas. A arquitetura torna o sistema híbrido combinando o processamento lento efetuado na camada em lote com o processo rápido e curto realizado na camada de velocidade, obtendo, desta forma, o melhor resultado de cada um.

A arquitetura Kappa abordada por Kreps (2014), inspirada na arquitetura Lambda, surge como uma versão simplificada, pois dispensa a camada em lote focando apenas nas camadas de serviço e velocidade, sua aplicação se dá em casos que necessitem processar apenas eventos ou dados de sensores, que não necessitem análises pesadas sobre eles (SOBREIRO 2018), não sendo necessário o reprocessamento periódico de todos os dados que estão na camada em lote.

A arquitetura Kappa tem como característica processar todos os dados em forma de fluxo e realizar o reprocessamento apenas quando ocorrer alteração na lógica do sistema, de modo que, para atingir esse objetivo é empregado um processador de fluxo com capacidade de trabalhar com uma taxa de dados maior do que está chegando, possuindo um sistema de fluxo escalável para retenção de dados (WINGERATH et al., 2016).

2.4 TRABALHOS RELACIONADOS

Existem diversos trabalhos na comunidade de pesquisa, que se concentram em desenvolver soluções para realizar a análise dos dados, detecção de ataques e identificar anomalias de rede. O número de trabalhos presentes na literatura com soluções desenvolvidas baseando-se no fluxo de dados de rede com processamento em tempo real é limitado em comparação com processamento em lote. Esta seção apresenta uma breve discussão sobre os trabalhos relacionados, considerando que as abordagens com maior relevância para esse estudo serão descritas de forma detalhada.

Para realizar a coleta e análise do fluxo de dados de rede, as pesquisas elaboradas estão aplicando diversos métodos com diferentes objetivos, tais como identificar a fonte de um ataque DDoS por meio dos arquivos capturados durante a observação tráfego de rede (OJENIYI et al., 2016), identificação de dispositivos, sistemas operacionais e aplicações de clientes que utilizam o protocolo *Hyper Text Transfer Protocol Secure* (HTTPS), por meio de um dicionário de dados com informações de uma lista de cifras obtidas do SSL/ TLS (*fingerprinting*) correlacionados aos *user-agent* ambos capturados da rede (HUSÁK et al., 2016). Processar o fluxo de dados de rede para reunir e correlacionar padrões anômalos com o objetivo de identificar ameaças à rede independentemente do seu tipo (DIVAKARAN et al., 2017).

A proposta apresentada por Karimi et al. (2016), é um sistema para detecção de intrusos (IDS), para isto é realizada a extração de informações do fluxo de dados de rede com base no protocolo TCP. Os dados são coletados por meio do Netmap, enviados ao Spark para extrair as informações do cabeçalho do protocolo e armazenar estes dados no *Hadoop File System*, então ocorre a extração das particularidades dos dados que foram armazenados. A partir das características processadas o sistema pode ser integrado com a biblioteca Spark MLib com o objetivo de aplicar um método para de detecção de anomalias.

Um sistema com a captura do fluxo de dados de rede, realizando o processamento em lote, com a finalidade de detectar anomalias é abordado por Kozik (2017), que aplica aprendizado de máquina (MA) para análise das informações obtidas. Seguindo nesta mesma linha, um sistema de detecção de ataques DDoS é aplicado por Han et al. (2017), com uma abordagem diferente, com base na entorpia (medição da desordem) das informações coletadas do tráfego de rede, essa implementação utiliza o Spark aplicando três variações do algoritmo *K-Means* (*Ordinary K-Means, Ordinary K-Means on the Spark Cluster e Improved K-Means on the Spark Cluster*). Em sua análise comparativa dos algoritmos, o terceiro obteve resultados

melhores por ter sua taxa de precisão maior que os demais e o tempo de consumo para sua execução ser o menor.

Neste mesmo segmento de detecção de ataques, há pesquisas que propuseram sistemas para realizar o processamento em tempo real, conforme Zhou et al. (2017) que utiliza o módulo de *Streaming* do Spark, para processar os dados capturados da rede em tempo real. Bem como no trabalho apresentado por Chen et al. (2016), é proposto um sistema para monitoramento e detecção de ameaças da rede em tempo real, aplicando os algoritmos *Fuzzy, C-Means* e *K-Means*, este último utilizando o Spark *Streaming*. A solução realiza as etapas de captura dos dados do tráfego de rede em tempo real por meio da ferramenta Wireshark, com isso realiza um pré-processamento para padronizar os dados que são recebidos de diferentes tipos de dispositivos, encaminha as informações ao servidor Flume e este repassa o fluxo de dados para o Spark realizar o processamento dos dados aplicando o *MapReduce* e armazenado os resultados, se forem identificadas atividades maliciosas ocorrem notificações. Em seus resultados é apresentada uma análise comparativa entre os dois algoritmos utilizados, tendo o *K-Means* se destacado em termos de precisão para detecção de ataques de varredura de portas.

Em pesquisas recentes, os sistemas propostos para realizar processamentos em tempo real, utilizam como modelo a arquitetura Lambda. Esta arquitetura foi utilizada por Hun et al (2018), em um sistema de monitoramento do fluxo de informações geradas por uma mídia social, um sistema semelhante foi implementado por Das et al. (2018) utilizando a mesma fonte de dados.

A implementação realizada por Nupairoj et al. (2016) é baseada na arquitetura Lambda, o objetivo do sistema é a detecção de anomalias por meio da previsão da largura de banda de rede. O fluxo da implementação consiste nas seguintes etapas, coleta dos *logs*, processamento dos *logs* nas camadas de lote e velocidade, respectivamente para se obter o histórico do consumo da rede e analisar o consumo atual. Nos dois módulos de processamento foi aplicado o modelo *Seasonal Autoregressive Integrated Moving Average* (SARIMA), executando esse método em ambas camadas é possível prever a largura de banda de rede com o objetivo de detectar anomalias.

Na pesquisa elaborada por Lopez et al. (2018) foi desenvolvido uma função de rede virtualizada no *Open Source Platform for Network Functions Virtualization* (OPNFV), que fornece um serviço preciso para a detecção de ameaças em tempo real. A arquitetura desta função é baseada na arquitetura Lambda, o sistema é composto por três módulos principais, captura dos dados, processamento do fluxo e alarme. O fluxo de dados de rede é capturado por

meio de sensores do BRO, para que seja possível realizar o processamento destes dados, os mesmos são armazenados no Apache Kafka que abstrai o fluxo de mensagens em tópicos para serem consumidos pelo Spark *Streaming,* que realiza o processamento destas informações com o objetivo de detectar ameaças, se forem encontradas são enviados alertas para o módulo de alarme. A função não implementou a arquitetura Lambda de forma completa pois a camada de lote não foi utilizada e para detecção de ameaças, foi implementada uma rede neural no módulo de processamento do fluxo.

Demertziz et al. (2019) elaborou um *framework* de fluxo de dados forense aplicado sobre a arquitetura Lambda (λ-NF3) para análise do tráfego de rede, empregando o algoritmo proposto em seu estudo, sendo ele uma combinação de *Extreme Learnign Machine* com *Gaussian Radial Basis Function kernel* (ELM/GRBFK) para o processamento em lote, enquanto que para tratar do fluxo de dados na camada de velocidade é utilizado o classificador *k-NN* em conjunto com *Self Adjusting Memory* (SAM/k-NN). Para fins de validação do desempenho dos algoritmos aplicados em cada camada, o mesmo realizou a comparação com métodos equivalentes, em lote foi considerado os algoritmos SVM, *Multi-Layer Artificial Neural Network*, *k-Nearest Neighbor* e *Random Forest*, na camada de velocidade utilizou *Hoeffding Adaptive Tree* e *SPegasos.* Em ambas camadas, o algoritmo proposto pelo autor teve resultados melhores do que os seus equivalentes.

Todos os trabalhos que foram citados durante essa seção, num total de treze, são apresentados na Tabela 1, na mesma ordem que foram mencionados no texto. Pode ser observado que os primeiros quatro trabalhos realizam análises e processamentos sobre os dados com diferentes métodos, dos quais foram obtidas informações sobre processos específicos que são aplicados em cada um. Os quatro trabalhos seguintes, aplicam em seu estudo a análise do fluxo de dados da rede, para isso utilizam diferentes algoritmos para classificação dos dados. Nesta sequência os dois trabalhos seguintes implementaram sistemas com base na arquitetura Lambda, com objetivo de processar fluxos de dados de uma rede social para obtenção de informações sobre a mesma, nos quais foi possível observar diferentes aplicações para esta arquitetura.

Os três últimos trabalhos da Tabela 1, aplicaram o conceito da arquitetura Lambda no desenvolvimento de seus sistemas, com o objetivo de efetuar o processamento do fluxo de dados da rede para detecção de anomalias. Os procedimentos aplicados foram distintos, bem como as informações consideradas, pois um considerou o consumo de banda de rede para detecção de anomalias, outro atuou sobre as informações do cabeçalho dos pacotes e o último

considerou as informações individuais de cada conexão para classificá-las entre anomalias ou não. Com estes três trabalhos foi possível identificar a aplicação de diferentes ferramentas, métodos e algoritmos que podem ser implementados sobre a arquitetura Lambda para efetuar a análise sobre o tráfego de rede.

Tabela 1 - Trabalhos relacionados.

Autor/Ano	Metodologia	Atuação
OJENIYI et al., 2016	Analisa o cabeçalho dos pacotes do protocolo ICMP	Identificar ataques DDoS
HUSÁK et al., 2016	Monitora o protocolo HTTPS e analisa informações de SSL/TLS	Identificar as ferramentas, navegadores e sistemas operacionais
DIVAKARAN et al., 2017	Processa fluxo de dados da rede referente as conexões HTTP, correlacionando padrões anômalos	Identificar *Malwares*
KARIMI et al., 2016	IDS aplicado sobre conexões TCP, com o uso de NETMAP e Spark	Identificar ataques DDoS
KOZIK, 2017	Analisa o fluxo de dados aplicando o algoritmo *Randon Forest*	Detectar atividades de *Botnet* no fluxo da rede
HAN et al., 2017	Analisa o fluxo de dados com base na entorpia das informações, aplicando variações do algoritmo *K-means* no processamento	Detecção de ataques DDoS
ZHOU et al., 2017	Realiza o processamento do fluxo de dados da rede pelo Spark *Streaming*, classifica as conexões conforme as características que apresentam	Detecção de ataques DDoS
CHEN et al., 2016	Um sistema para processamento do tráfego de rede em tempo real com Spark *Streaming*, que aplica os algoritmos *K-means* e *C-means*	Detectar ameaças DDoS
HUN et al., 2018	Implementou um sistema baseado na arquitetura Lambda, aplicando o algoritmo *MapReduce*	Extrair informações do fluxo de dados do Twitter
DAS et al., 2018	Desenvolveu sistema baseado na arquitetura Lambda aplicando redes neurais recorrentes	Realizar previsão de ações predominantes, com base no fluxo de dados do Twitter
NUPAIROJ et al., 2016	Sistema com base na arquitetura Lambda para processamento do fluxo de dados da rede, aplicando o modelo SARIMA em ambas camadas	Detecção de anomalias na rede com base na largura de banda de rede
LOPEZ et al., 2018	Função de rede virtualizada desenvolvida com base na arquitetura Lambda, implementando as camadas de velocidade e serviço. Para processamento do fluxo de dados da rede, analisando as informações de cabeçalho do pacote	Detecção de ameaças DoS
DEMERTZIZ et al., 2019	Desenvolveu um *framework* com base na arquitetura *Lambda,* para análise forense do fluxo de dados da rede. Aplicando algoritmos diferentes para a camada de lote e velocidade	Detecção de *Malwares*

Fonte: O autor.

2.5 SUMÁRIO

O presente capítulo detalhou alguns tópicos relacionados ao processamento do fluxo de dados da rede. Após apresentar determinados tipos de ataques que causam anomalias no tráfego de rede e técnicas que podem ser aplicadas na detecção destes eventos, abordou-se o conceito sobre *Big Data* e arquitetura Lambda. Por fim, descreveu-se alguns trabalhos relacionados que

aplicaram diferentes métodos para a análise do tráfego de rede, inclusive sistemas implementados com base na arquitetura Lambda com o uso de ferramentas diferentes. Todavia, nenhuma delas aplicou análises sobre os resultados classificados como anômalos com objetivo de obter informações pertinentes a sua origem.

3 SISTEMA DESENVOLVIDO

A análise de grandes conjuntos de dados é comumente realizada por processamentos em lotes que apresentam grandes latências para exibição dos resultados. O sistema proposto, é baseado na arquitetura Lambda, tem como objetivo o processamento do fluxo de dados da rede. Com a utilização da arquitetura Lambda, o sistema possui capacidade de processamento com baixa latência de resposta disponibilizada pela camada de velocidade, bem como capacidade de analisar os dados de forma mais profunda e reprocessar o seu histórico, se necessário, obtendo essa característica da camada de lote, e, por fim, a possibilidade de visualizar os resultados obtidos das camadas anteriores por meio da camada de serviço.

A computação realizada sobre o tráfego de rede tem como objetivo identificar anomalias geradas por ataques de DDoS por meio do protocolo HTTP, força bruta contra o serviço SSH e varredura de portas efetuados sobre o protocolo TCP, as análises serão restritas a esses três tipos de ataques, sendo que cada um considera um protocolo específico. Com os resultados classificados como anômalos, efetuar-se-á uma análise sobre os endereços IP responsáveis por estas ações para obter informações sobre país, cidade, latitude, longitude e organização dos mesmos.

Além disso, gerar-se-á resultados exibindo a linha de tempo das conexões efetuadas por estes IP's, identificando assim o período no qual ocorreu a anomalia e visualizar a diferença com o período considerado normal. Outro propósito é conseguir informações sobre o tráfego de rede de modo geral, como os totais de conexões, duração, *Megabytes* (MB) enviados e recebidos referentes aos protocolos de comunicação e serviço, assim como definir os IPs externos que foram mais acessados a partir da rede local e os IPs que efetuaram o maior número de conexões da rede externa para rede local.

Na sequência deste capítulo, é abordado o sistema proposto, apresentando a arquitetura sobre a qual foi implementado, as ferramentas utilizadas e todas as configurações e codificações realizadas para a integração entre as mesmas, a estrutura lógica de funcionamento desde a coleta dos dados até o armazenamento, o processamento aplicado sobre os dados, assim como as análises realizadas. Portanto realizou-se, inicialmente, o detalhamento da arquitetura do sistema e as ferramentas utilizadas, assim como o funcionamento lógico para efetuar a coleta dos dados, transporte, normalização e armazenamento que são abordados na seção 3.1.

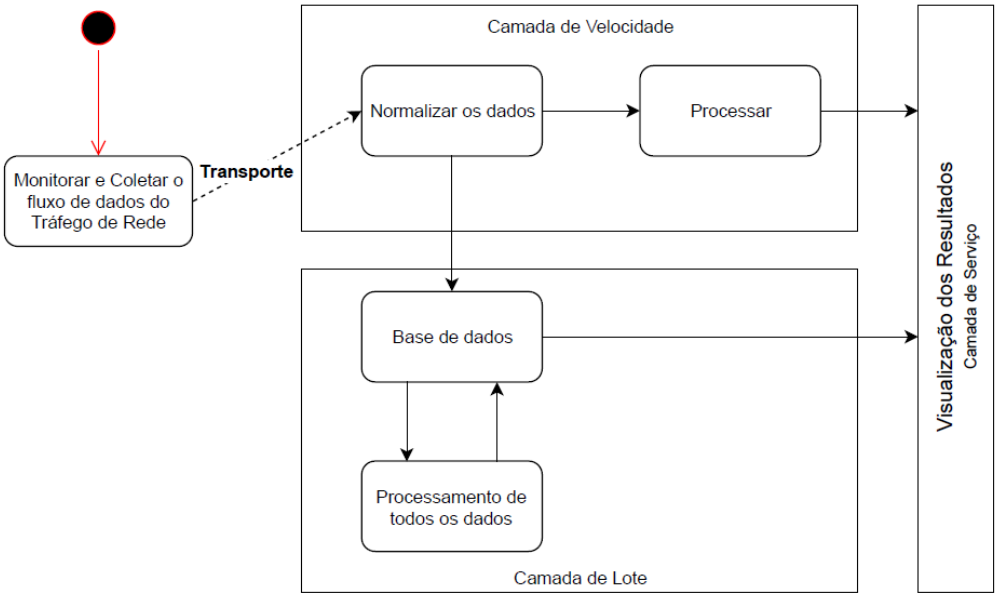
A seção 3.2 especifica os procedimentos aplicados sobre o fluxo de dados do tráfego de rede desde a sua coleta até o seu armazenamento. Por fim, a seção 3.3 detalha os processamentos

efetuados sobre os dados com o objetivo de obter informações sobre os mesmos, assim como os procedimentos executados para identificar anomalias aplicando o algoritmo *K-means*.

3.1 ARQUITETURA DO SISTEMA

A implementação foi baseada na arquitetura Lambda que permite a manipulação e análise de grandes quantidades de dados por meio das camadas de velocidade, lote e visualização dos resultados na camada de serviço. A arquitetura pode ser dividida em algumas etapas que são agrupadas em suas respectivas camadas, velocidade, lote e serviço, conforme pode ser observado na Figura 3. A primeira etapa é responsável por monitorar e coletar os dados do tráfego de rede que devem ser analisados. O próximo passo é o transporte e disponibilização destas informações para serem processadas, a partir disso, os mesmos são normalizados, ou seja, são acrescentadas informações e os dados são filtrados de acordo com a característica de sua conexão, disponibilizando-os em um formato que estejam aptos a serem processados.

Figura 3 - As três camadas da arquitetura Lambda, aplicadas ao sistema proposto.



Fonte: o autor.

Com os dados normalizados, estes podem ser processados pela camada de velocidade, encaminhando os resultados para visualização na camada de serviço ou podem ser armazenados para serem trabalhados pela camada de lote. Com as informações armazenadas, ocorre o processamento e análise de todos os dados para sua classificação, sendo que os resultados

obtidos destes processos são salvos. A visualização dos resultados da camada de lote é possível a partir das tabelas nas quais foram armazenados.

De acordo com Figura 3 e suas etapas, foram selecionadas diferentes ferramentas que são utilizadas para atender cada processo, onde cada uma delas atende a uma funcionalidade específica e realiza a integração com as demais, conforme detalhado a seguir.

The Zeek Network Security Monitor

A primeira etapa é responsável por efetuar o monitoramento em tempo real do tráfego da rede, isto é, conforme novas conexões são realizadas, as mesmas devem ter suas informações coletadas e disponibilizadas para processamento. Para tal fim é aplicado o ZEEK[2], um *framework open-source* de segurança de rede, que tem como principal característica o monitoramento de todo o tráfego de rede, onde o fluxo monitorado é registrado em um extenso conjunto de *logs,* que incluem diversos protocolos.

Para processar os dados das conexões conforme elas ocorrem, é necessário que as informações sejam transportadas de forma ágil, o que não irá ocorrer se as mesmas forem escritas em um arquivo *log* para posterior transporte do arquivo. Desta forma utiliza-se a linguagem *Script*, por meio da qual a ferramenta permite a codificação de novas funcionalidades, em conjunto com o plugin[3] do ZEEK que efetua comunicação com o Apache Kafka. A partir disto, é feita a escrita das informações dos protocolos selecionados diretamente ao Apache Kafka no formato JSON.

Apache Kafka

Com a necessidade de transportar os dados das conexões da rede para serem processados, utiliza-se o Apache Kafka[4], uma ferramenta distribuída que possui capacidades como publicar, distribuir e receber um fluxo de dados como uma fila de mensagens em tempo real. Possui baixa latência de resposta e altas taxas de transmissão, suportando aplicativos com fluxo de dados em tempo real, o que torna viável seu uso para processamento de eventos e integração com sistemas de processamento em larga escala (GÜRCAN e BERIGEL 2018). As

[2] ZEEK: https://www.zeek.org/
[3] Plugin: https://github.com/apache/metron-bro-plugin-kafka
[4] Apache Kafka: https://kafka.apache.org/

aplicações do Kafka conseguem trabalhar com milhares de mensagens em curtos intervalos de tempo (SOBREIRO, 2018).

O Kafka atua com conceito de *producers* e *consumers,* sendo respectivamente o responsável por produzir essas informações e o que efetua o consumo destes dados. Nesse cenário, consideramos o ZEEK como *producer*, o qual efetua a escrita dos dados em determinado tópico. Enquanto o Spark *Streaming* atua *consumer,* lendo os dados deste tópico conforme novas informações são escritas. Essa característica de funcionamento do Kafka, abstrai a necessidade de *consumer* e *producer* terem conhecimento um do outro.

Apache Spark

A segunda etapa, trabalha com um fluxo de dados e para isso é necessário uma ferramenta que possua esta característica, dentro deste requisito aplica-se o Apache Spark[5], que é uma ferramenta de código aberto aplicada à computação distribuída, escrita nas linguagens de Scala, Python e Java (OUNACER et al. 2017), uma plataforma de computação voltada para utilização da memória. Spark possui uma coleção de bibliotecas que podem ser combinadas para trabalharem em conjunto, que inclui SQL, *DataFrames* e *Datasets*, MLlib para aprendizado de máquina, *GraphX* processamento de grafos e Spark *Streaming* utilizado para realizar processamento de fluxos próximo ao tempo real (STOREY e SONG, 2017; MARCHAL et al., 2014).

Dentre essas bibliotecas, Spark *Streaming*, SQL, *DataFrames* e *Datasets* são aplicadas em conjunto para efetuar o consumo, normalização e armazenamento dos dados disponíveis no Kafka. Para compreensão de quais características cada uma apresenta e para que são utilizadas nessa etapa, as mesmas são detalhadas a seguir.

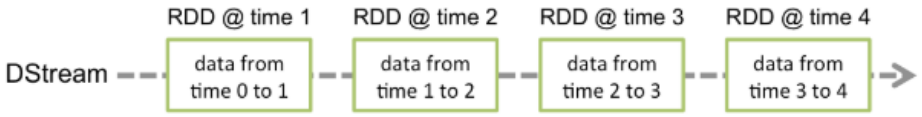
A programação no Spark tem seu conceito baseado em *Resilient Distributed Datasets* (RDD). O RDD é abstração principal fornecida pelo Spark, é uma coleção de elementos particionados em nós do *cluster* que podem ser operados em paralelo. Tem como característica principal ser imutável, podendo apenas criar um novo RDD baseado em outro ou sobre um conjunto de dados inicial.

Spark *Streaming* permite o processamento de fluxo de dados, para efetuar esse processo o mesmo recebe o fluxo e divide os dados em pequenos lotes que são processados pelo Spark. Esses pequenos lotes internamente são representados por uma sequência de RDDs, conforme

[5] Apache Spark: https://spark.apache.org/

pode ser observado na Figura 4. O Spark *Streaming* possui uma abstração de alto nível denominada *discretized stream* (*DStream),* que representa um fluxo contínuo de dados e pode ser criada a partir de fluxo de dados de entrada, nesse caso o Kafka ou a partir de outro *DStream.*

Figura 4 - Abstração do DStream.



Fonte: Apache Spark: https://spark.apache.org/.

O módulo de SQL, *DataFrames* e *Datasets* é voltado para processamento de dados estruturados, originados de diferentes fontes e tipos, nesse caso é considerado o tipo JSON e a base de dados HBase. Os resultados obtidos destas fontes são retornados como *Dataset/DataFrame*, com isso é possível agrupar as informações recebidas de diferentes fontes e processá-las por meio do Spark. Sendo que o *Dataset* é um conjunto de dados distribuídos e *DataFrame* pode ser considerado um *Dataset* organizado em colunas, equivalente a uma tabela de banco de dados relacional.

Os dados consumidos pelo Spark *Streaming* são normalizados utilizando a biblioteca SQL, *DataFrames* e *Datasets,* a partir disto os resultados obtidos são armazenados no Apache HBase. Para efetuar a comunicação com o HBase, ocorre a integração do Spark com o Apache Phoenix.

Devido as funcionalidades disponíveis no Spark, a ferramenta também é aplicada à etapa de processamento dos dados armazenados. Utilizando a biblioteca SQL, *DataFrames* e *Datasets,* para manipular os dados, em conjunto com a MLlib, que possui algoritmos de aprendizagem como classificação, regressão, agrupamentos e filtragem colaborativa, neste cenário é aplicado o algoritmo *K-means* para análise dos dados.

Apache HBase

Para armazenar um fluxo de informações significativo como o tráfego de rede é necessária uma base de dados com altas taxas de leitura e escrita, com isso o Apache HBase[6] foi selecionado. Uma base de dados escalável, distribuída e aplicada no ambiente *Big Data*, sua

[6] Apache HBase: https://hbase.apache.org/

implementação em *clusters* pode armazenar tabelas com bilhões de linhas e milhões de colunas. Sobre esta base de dados, também serão armazenados todos os resultados de processamentos efetuados sobre as informações do tráfego da rede.

Conforme Li et al. (2018) os bancos de dados *NoSQL* tem se desenvolvido rapidamente em relação aos tradicionais, pois são escaláveis, tem mais flexibilidade e uma performance melhor para grandes quantidades de dados. HBase possui tolerância a falhas e rápido acesso, é um banco de dados orientado a colunas e foi construído para executar sobre o *Hadoop Distributed File System* (HDFS), utilizado pelas aplicações que necessitam de altas taxas de escrita/leitura em grandes conjuntos de dados e por outras que possuem muitos clientes (JAHN, 2017; GÜRCAN et al., 2018;).

As tabelas possuem linhas e colunas que são armazenadas como um mapa esparso multidimensional, possui uma chave para classificar os dados, o número de colunas é arbitrário e são armazenadas em famílias de colunas que são definidas de forma estática. Por meio da chave, do nome da família da coluna e o nome da coluna é feito o acesso a célula desejada, ou seja, o campo da tabela, além desta sequência de acesso descrita *rowkey:column family:column* cada célula é associada a um *timestamp*, sendo possível com esse recurso manter várias versões da mesma célula (LI et al., 2018).

De forma rápida Bhojwani (2016) define uma tabela no HBase como um conjunto de linhas, a linha sendo uma coleção de famílias de colunas, a família de colunas sendo uma coleção de colunas e a coluna um conjunto de pares com chave e valor. As linhas são armazenadas em regiões, que agrupa um determinado intervalo da chave, os dados são distribuídos nas regiões conforme a sua chave e as regiões podem ser divididas e distribuídas automaticamente para outros recursos disponíveis (LI et al., 2018).

Apache Phoenix

Com a necessidade de efetuar a integração entre o HBase e o Spark, para armazenar os dados normalizados pelo Spark, utilizou-se o Apache Phoenix[7] uma ferramenta *open-source* desenvolvida em Java, que atua como uma camada de banco de dados relacional sobre o HBase. Ele complementa a camada de armazenamento do HBase com processamento de consultas, pois é utilizado para realizar o processamento de transações online, bem como sua análise. Com o Phoenix é possível realizar consultas em *Structured Query Language* (SQL), que no momento

[7] Apache Phoenix: https://phoenix.apache.org/index.html

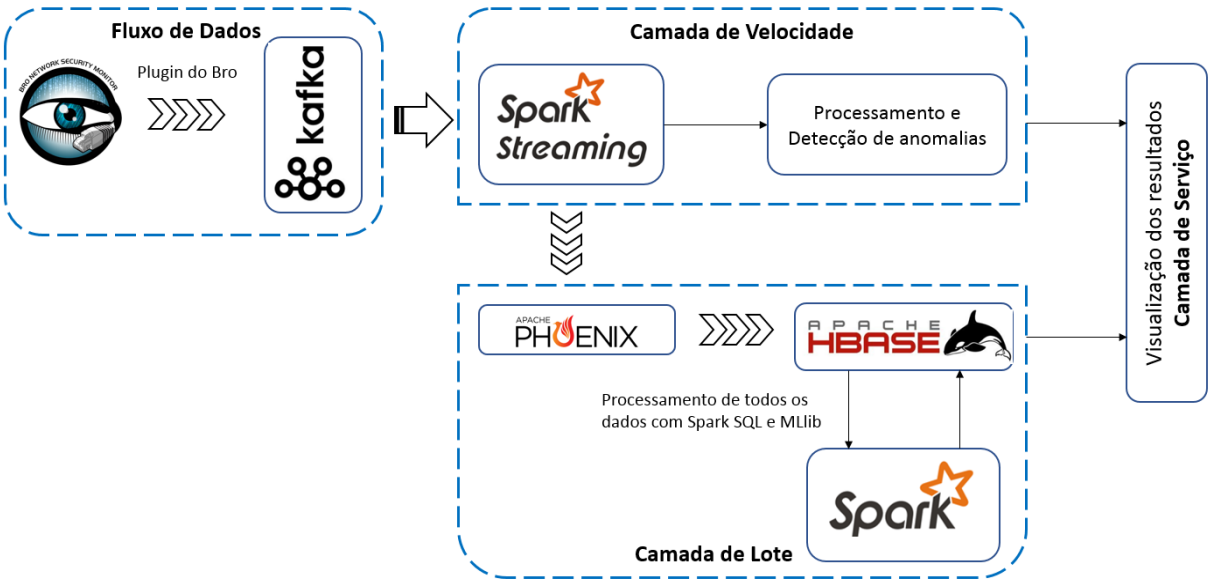
da execução no HBase são convertidas em operações de *SCAN, PUT* e *DELETE* (GUGNANI et al., 2017; VERMA et al., 2017; SHACHAM et al., 2018). O seu relacionamento acontece como um driver JBDC incorporado ao cliente que disponibiliza consultas de baixa latência nas tabelas do HBase (KATKAR, 2015).

A integração é realizada diretamente com os componentes *RDD* ou *DataFrame* do Spark, com isso é possível persistir os dados destes componentes nas tabelas do HBase sem a necessidade de algum outro tratamento ou conversão. O mesmo ocorre para efetuar a leitura dos dados das tabelas, os resultados são inseridos diretamente em um *DataFrame.*

Integração entre as ferramentas

A integração entre todas as ferramentas descritas, bem como as etapas de atuação de cada uma pode ser observada na Figura 5. A primeira parte da arquitetura denominada "Fluxo de Dados", é responsável por efetuar o monitoramento e coleta do tráfego de rede, bem como o transporte destes dados, disponibilizando-os para consumo. A segunda parte da arquitetura "Camada de Velocidade", consome as informações do tráfego de rede, normalizando os dados para estarem aptos a serem armazenados ou processados, com isso os dados são salvos na tabela para serem trabalhados pela camada em lote. A "Camada de Lote" realiza o processamento sobre os dados que estão armazenados em tabelas, nesse processo aplica o algoritmo *K-means* para detecção de anomalias.

Figura 5 - Arquitetura do sistema desenvolvido, com a integração entre as ferramentas selecionadas.



Fonte: o autor.

A aplicação do algoritmo *K-means*, nesta etapa, foi definida face aos tipos de ataques a serem analisados, DDoS, Brute Force e Scan Port, e as características que os mesmos apresentam no tráfego de rede, em torno do aumento significativo do número de conexões, conforme abordado no tópico 2.1. Deste modo, a análise a ser realizada tem como foco definir se o volume de conexões originadas pelos IPs coletados do tráfego de rede, apresentam ou não características anômalas em relação ao conjunto de dados analisado.

O *K-means* é aplicado considerando o volume de conexões, e não as características individuas de cada uma, com objetivo de classificá-los como anômalo ou não, com base no modelo gerado a partir de todas as conexões coletadas. Outro fator de sua utilização a ser mencionado é sua disponibilidade na biblioteca MLlib do Spark. Por fim, a última parte da arquitetura, a camada de serviço atua como responsável por exibir os resultados obtidos dos processamentos das camadas de velocidade e lote. O relacionamento entre as ferramentas descritas anteriormente, sobre o qual o sistema foi implementado, tem seu detalhamento apresentado nas próximas seções, com todas as interações realizadas em cada etapa para chegar ao resultado.

3.2 FLUXO DE DADOS DA REDE

A presente seção tem por objetivo apresentar todas as integrações realizadas entre as ferramentas responsáveis por trabalhar diretamente com o fluxo de dados do tráfego de rede. Para isso, é especificado no tópico 3.2.1 como ocorre o monitoramento da rede por meio do ZEEK e as configurações realizadas para que o mesmo possa coletar as informações do tráfego de rede e escrever esses dados no tópico do Kafka. Na segunda parte desta seção (tópico 3.2.2), é apresentada a integração efetuada entre Spark *Streaming* com o Kafka para que o mesmo possa consumir os dados do tópico, assim como realizar o processo de normalização das informações recebidas para armazenamento na tabela do HBase.

3.2.1 Monitoramento, coleta e transferência

A primeira etapa realizada pelo sistema é a coleta do fluxo de dados da rede, o ZEEK é responsável por essa tarefa, é configurado para realizar o monitoramento de determinada rede gerando *logs* com as informações coletadas. O arquivo de *log* gerado denominado “conn” pode ser considerado o arquivo principal, pois possui todas as conexões que foram realizadas sobre o tráfego TCP, UDP e ICMP, e cada conexão registrada nele possui um identificador único

exclusivo da conexão. Os principais campos da conexão presentes no arquivo “conn” são apresentados na Tabela 2.

Tabela 2 - Campos do *log* “conn”.

Campos	Descrição
TS	*Timestamp* referente ao primeiro pacote
UID	Identificador exclusivo da conexão
ID.ORIG_H	Endereço IP que originou a conexão
ID.ORIG_P	Número da Porta que originou a conexão
ID.RESP_H	Endereço IP que respondeu a conexão
ID.RESP_P	Número da Porta que respondeu a conexão
PROTO	Protocolo da camada de transporte
SERVICE	Identificação do protocolo de aplicação
DURATION	Duração da conexão
ORIG_IP_BYTES	Número de *Bytes* que o IP de origem enviou
RESP_IP_BYTES	Número de *Bytes* que o IP de resposta enviou

Fonte: o autor.

Diversos arquivos de *logs* são gerados pelo ZEEK durante o monitoramento da rede, além do arquivo de *log* “conn” mencionado acima, foram selecionados apenas alguns para serem utilizados, sendo eles “dns”, “http”, “dhcp” e “ssh”, considerando que cada um possui informações específicas sobre o protocolo ao qual o nome do mesmo faz referência. Todos os arquivos selecionados possuem os seis primeiros campos detalhados na Tabela 2. O campo UID pode ser utilizado para identificar todas as atividades registradas associadas a uma determinada conexão, com isso o mesmo UID pode ser encontrado em diversos arquivos de *log*. O arquivo “conn” possui apenas uma linha com determinado UID, enquanto os demais arquivos podem possuir *N* linhas com o mesmo UID, pois registram todas atividades efetuadas por aquela determinada conexão.

A segunda etapa executada pelo sistema é a transferência dos dados de cada arquivo para o tópico Kafka por meio do *plugin* de escrita do ZEEK que atua como *producer*, para isso criou-se um tópico no Kafka nomeado de “BroLog” no qual serão escritos todos *logs* selecionados. Para o ZEEK efetuar a escrita dos *logs* no tópico Kafka criado, codificou-se um *script* a ser executado, contendo todas as informações necessárias sobre os arquivos para transferência, assim como o formato e o tópico Kafka no qual será escrito. A seguir são apresentadas as configurações inseridas:

- Host Kafka: namenode.ambari.hadoop:6667

- Tópico Kafka: BroLog
- Formato dos dados: JSON
- Formato do campo TS: TS_MILLIS (*timestamp* em milissegundos)

Cada linha dos diferentes *logs* é escrita diretamente no tópico "BroLog" em formato JSON, como identificador é utilizado a *tag* primária com o nome do arquivo de *log,* para ser possível identificar a qual *log* aquela determinada linha JSON pertence. Com este processo o fluxo de dados com as informações referentes aos arquivos de *logs* especificados durante esta subseção, ficam disponíveis para serem consumidos pelo tópico "BroLog" criado no Kafka.

3.2.2 Consumo, normalização e armazenamento

O Spark atua como consumidor do tópico "BroLog", especificamente o módulo de *Streaming* do Spark, pois os dados estão disponíveis em um fluxo contínuo. O código implementado para executar esse estágio necessita de algumas informações sobre o tópico do Kafka, que tem seus valores definidos conforme descrito a seguir:

- Host Kafka: namenode.ambari.hadoop:6667
- Tópico Kafka: BroLog
- Intervalo para leitura do tópico: 30 segundos

Com essa configuração ocorre o consumo dos dados no intervalo de tempo determinado, formando lotes de dados a serem processados, o Spark cria um *DStream* que consome todas as mensagens do tópico neste intervalo de tempo, a partir dele é efetuada a leitura de cada mensagem criando outro *DStream* que contém o valor de todas as mensagens lidas em uma sequência de RDDs. Para acessar os RDDs contidos no *DStream* utilizou-se a função "*foreachRDD"*, com acesso ao RDD que possui a coleção dos dados efetuou-se sua transformação para um *dataset* no formato JSON, através dos seguintes passos:

a) Executa um *colletct* nos dados do RDD, que retorna uma lista com todos as linhas de dados;
b) Cria um *dataset* do tipo *string* a partir da lista;
c) Transforma o *dataset* do tipo *string* para o tipo JSON do Spark;
d) Cria um *dataframe* a partir do *dataset* JSON existente.

Com o resultado do último passo descrito acima, o Spark reconhece o conjunto de dados como um *dataframe*, deste modo o mesmo pode trabalhar usufruindo de todas as funcionalidades presentes neste tipo. Em seguida, ocorre a etapa de normalização dos dados para que os mesmos possam ser armazenados no banco de dados. As informações de cada *log*

são tratadas de forma separada, para isso é utilizada a *tag* primária que foi definida no *plugin* do ZEEK durante a escrita dos dados no tópico "BroLog", portanto cada um dos *logs* selecionados executa o processo descrito a seguir:

a) Filtra os dados do *dataframe,* que possui as informações do tópico Kafka, pelo nome do arquivo de *log(conn, http, ssh, dns, dhcp)* criando outro *dataframe,* apenas com os dados que respeitam o filtro;
b) Renomeia todos os campos referentes ao conjunto ID substituindo os pontos por *underline* "_";
c) Cria coluna com nome "tipo", que recebe o nome do arquivo filtrado;
d) Cria coluna com nome "ts_code", recebe o *timestamp* do sistema;
e) Cria coluna com nome "row_id", recebe um identificador único para cada linha do *dataframe* atual, gerado por meio do método "*monotonically_increasing_id"*;

A coluna "row_id" é necessária, pois pode ocorrer a multiplicidade de atividades da rede e que estas possuam o mesmo UID, TS e ID, tendo em vista o formato do campo TS que foi definido para TS_MILLIS na escrita dos *logs* do ZEEK para o Kafka. O formato original do campo TS é TS_EPOCH que possui o valor do tipo *double,* portanto ao utilizar essa nova coluna não ocorre impacto no momento do armazenamento dos dados.

Com os dados normalizados e prontos é chamado o método "*write"* para salvar os dados na tabela do HBase, utilizando o Phoenix como intermediário nesse processo abstraindo a necessidade de executar operações individuais de PUT para cada linha a ser inserida na tabela do HBase, pois o mesmo possui integração direta com o Spark possibilitando que todos os dados do *dataframe* sejam salvos diretamente na tabela desejada. Todo o processo descrito acima, é realizado de forma contínua sempre que o tópico "BroLog" possuir dados a serem consumidos.

A tabela na qual os dados normalizados são armazenados foi nomeada de "LOG", a chave de linha "*Rowkey"* é definida por nove campos que são especificados na Tabela 3, cinco famílias de colunas e oitenta e seis campos, com isso todas as informações que estão em cada um dos arquivos de *log* gerados pelo ZEEK são salvos na tabela. As famílias de colunas foram definidas com o objetivo de cada uma agrupar todas as informações referentes a um arquivo de *log* específico gerado pelo ZEEK, as mesmas são nomeadas e detalhadas a seguir:

a) Família de coluna CONN possui dezoito colunas, as quais armazenam informações referentes ao arquivo de *log* "conn", que possui todas as conexões do tráfego de rede coletadas pelo ZEEK;

b) Família de coluna denominada de DNS é composta por dezenove colunas, as quais armazenam todas as informações de consultas e respostas do protocolo DNS;
c) Família de coluna nomeada de HTTP contém vinte e três colunas que armazenam informações referentes à solicitação e resposta realizadas por meio do protocolo HTTP;
d) Família de coluna denominada SSH possui doze colunas que são responsáveis por armazenar informações pertinentes às conexões efetuadas por meio do serviço SSH;
e) Família de coluna DHCP é composta por quatro colunas que salvam informações referentes ao tráfego do protocolo DHCP coletados durante o monitoramento da rede;

Os campos que compõem a *Rowkey* são definidos com informações geradas no ZEEK e por outras agregadas, no momento da normalização dos dados, pelo Spark. A partir das definições apresentadas sobre a tabela "LOG", é possível constatar que todas as informações dos cinco *logs* escritos pelo ZEEK no tópico "BroLog", são armazenadas em suas respectivas famílias de colunas.

Tabela 3 - Tabela LOG.

Campos	**Tipo**	**Descrição**
TIPO	varchar(4)	Nome do arquivo de LOG do ZEEK
TS	timestamp not null	*Timestamp* do ZEEK referente ao primeiro pacote
TS_CODE	timestamp not null	Timestamp do sistema antes de salvar os dados
ID_ORIG_H	varchar	Endereço IP que originou a conexão
ID_ORIG_P	unsigned_long not null	Número da Porta que originou a conexão
ID_RESP_H	varchar	Endereço IP que respondeu a conexão
ID_RESP_P	unsigned_long not null	Número da Porta que respondeu a conexão
PROTO	varchar	Protocolo da camada de transporte
UID	varchar	Identificador exclusivo da conexão
ROW_ID	unsigned_long not null	Identificador atribuído a linha do *Dataframe*

Fonte: O autor.

3.3 PROCESSAMENTO E ANÁLISES

Para realizar o processamento dos dados em lote utilizou-se o Spark, a seleção dos dados da tabela "LOG" do HBase para o Spark, é efetuada por meio da integração com o Phoenix, com isso todos os dados selecionados estarão disponíveis no formato *dataframe* do Spark, aptos para manipulação. Os processamentos são realizados para obter informações sobre os totais de conexão, duração, *Megabytes* (MB) enviados e recebidos pelos protocolos, assim como definir os IPs externos que foram mais acessados a partir da rede interna e os IPs que efetuaram o maior

número de conexões da rede externa para rede interna. Entende-se por rede interna, a rede intranet e a rede externa, nesse caso, é a internet. Também é aplicado o algoritmo *K-means* para detectar anomalias referentes a ataques DDoS, força bruta e varredura de portas sobre os protocolos HTTP, SSH e TCP, respectivamente.

Em todos os processamentos realizados foram considerados os dados da tabela "LOG" que respeitam a seguinte condição "TIPO == conn", logo, apenas as informações referentes às conexões serão selecionadas, portanto não serão consideradas todas as atividades de determinada conexão, e sim a conexão em si. O resultado desta seleção é armazenado em um *dataframe* que nesse capítulo é nomeado de LT_DATA, e persistido na memória/disco com o método do Spark "*persist()",* permitindo que os dados sejam utilizados de maneira eficiente em execuções paralelas durante o processamento pelo Spark.

Essa seção detalha os procedimentos aplicados sobre os dados armazenados. O primeiro tópico 3.3.1 especifica como ocorre a manipulação dos dados para obter totais por "dd.MM.yyyy HH", sobre os protocolos de transporte e aplicação e pelo número de conexões IP de origem sobre o mesmo IP de resposta. O segundo tópico 3.3.2 especifica os passos realizados para aplicar o algoritmo *K-means* no processamento dos dados, para detectar as anomalias causadas pelos ataques DDoS, força bruta e varredura de portas. Por fim, o último tópico 3.3.3 descreve os procedimentos aplicados sobre os resultados obtidos nos tópicos anteriores, detalhando os processos efetuados para obter a linha do tempo com o número de conexões por protocolo e IPs vítimas de ataques, assim como descreve o processo realizado para obter informações sobre os IPs por meio do *Web Service* IPinfo.

3.3.1 Informações do fluxo de dados

Com os dados do LT_DATA são utilizados métodos disponíveis no módulo de SQL do Spark para processar estas informações, com esse intuito é definida uma tabela temporária a partir do LT_DATA utilizando o método "*createOrReplaceTempView",* nomeando-a de "SUM", deste modo é possível efetuar consultas SQL no LT_DATA, referenciando a tabela criada. Com o objetivo de conseguir os totais das conexões criou-se o método *M_GROUP_SUM*, que recebe um *dataframe* com os dados a serem processados, o nome dos campos que serão utilizados para agrupar as informações, um identificador que diferencia cada tipo de agrupamento e a descrição do processo. O *dataframe* passado ao método para todas as suas chamadas é o LT_DATA, os demais parâmetros passados para efetuar a totalização são detalhados na Tabela 4.

Tabela 4 - Totais por campos.

Campos	Identificador	Descrição
proto	proto	Protocolo
service	service	Serviço
id_orig_h+id_resp_h	orig_h_resp_h	IP de Origem e IP de Resposta

Fonte: O autor.

O método efetua o agrupamento dos campos que foram passados, considerando também o campo “TS” no formato de “dd.MM.yyyy HH”, utiliza os campos “DURATION”, “ORIG_IP_BYTES”, “RESP_IP_BYTES”, “ORIG_PKTS” e “RESP_PKTS” para sumarização e efetua a contagem do número de dados que foram agrupados com essas premissas, criando assim o campo “COUNT”. Com estas informações o método monta uma consulta SQL para ser executada no *dataframe* LT_DATA recebido, dessa forma cria-se o *dataframe* LT_TOTAL com o resultado desta consulta. No *dataframe* LT_TOTAL são adicionadas as seguintes colunas:

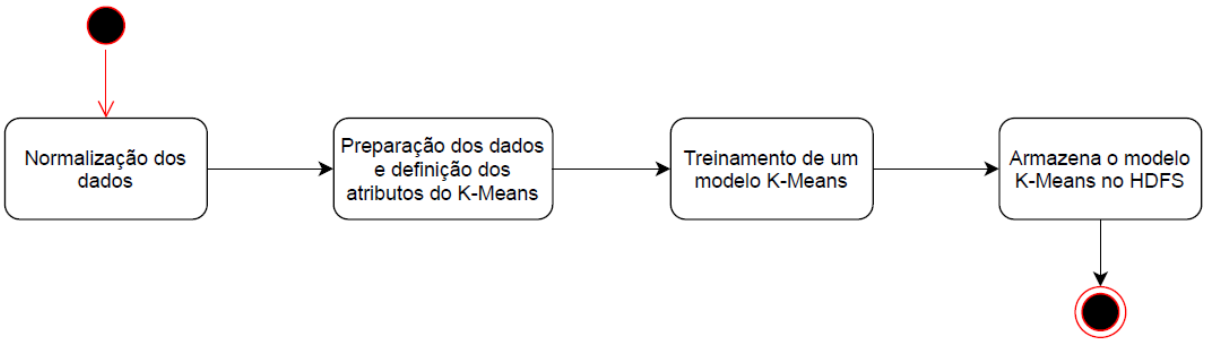
a) “TIPO” que recebe o identificador passado ao método;
b) “TS_FILTRO” recebe o filtro utilizado na seleção dos dados da tabela de LOG;
c) “TS_CODE” considera o *timestamp* do sistema;
d) “ROW_ID” recebe o valor atribuído pelo método “*monotonically_increasing_id”;*

Desta forma, no final do método *M_GROUP_SUM* o *dataframe* LT_TOTAL resultante tem os seus dados armazenados na tabela “LOG_TOTAIS” que possui dezesseis campos, sendo “TIPO”, “TS_FILTRO”, “TS_CODE” e “ROW_ID” definidos para compor a chave da linha. Os demais campos armazenam as informações de cada agrupamento realizado, com esse procedimento obtemos os totais por “dd.MM.yyyy HH” dos protocolos, serviços e IP de origem.

3.3.2 Detecção de anomalias

Com a finalidade de detectar anomalias sobre os dados da rede, são consideradas as informações selecionadas da tabela “LOG” que estão disponíveis no LT_DATA. Os processos aplicados sobre os dados para detecção de anomalias são divididos em duas etapas que são executadas separadamente, a primeira é responsável por gerar o modelo *K-means*, enquanto a segunda efetua a classificação dos dados conforme o modelo gerado, sendo que os dois primeiros processos em ambas são os mesmos. A primeira etapa é apresentada na Figura 6, a qual é responsável pela geração do modelo K-means a ser utilizado na classificação dos dados. Para cada tipo de anomalia a ser detectada estas etapas são executadas.

Figura 6 - Etapa de criação do modelo *K-Means*.



Fonte: O autor.

Primeiramente, é detalhado o processo de normalização dos dados presentes na Figura 6, iniciando pelos protocolos HTTP e SSH, seguido pelo protocolo TCP. Na sequência, o segundo processo é especificado, considerando o resultado obtido no primeiro procedimento, desta forma os dados são preparados e os atributos do *K-means* definidos. Com isso ocorre o treinamento do modelo *K-means,* e por fim, o último procedimento armazena o modelo gerado.

Com os dados do LT_DATA realiza-se a inserção de um filtro para selecionar apenas as informações onde o campo “SERVICE” igual a “http”, logo cria-se outro *dataframe* LT_HTTP com o resultado deste filtro efetua-se o procedimento para totalização e agrupamento dos dados filtrados. Esse procedimento é demostrado na Figura 7, a qual apresenta na primeira tabela uma ilustração do *dataframe* LT_HTTP, onde cada linha é considerada uma conexão, e na segunda tabela o resultado obtido após o procedimento. É possível observar que os dados são sumarizados de acordo com as informações das primeiras quatro colunas, e no resultado é adicionado a coluna “COUNT”, a qual contém o número de linhas que foram agrupadas, que representa o número de conexões.

Figura 7 – Exemplo do processo aplicado sobre os dados do *dataframe* LT_HTTP

ID_ORIG_H	ID_RESP_H	ID_RESP_P	TS	DURATION	ORIG_PKTS	ORIG_IP_BYTES	RESP_PKTS	RESP_IP_BYTES
106.13.34.4	192.168.0.30	80	10/12/18 16:28	10	6	700	22	960
106.13.34.4	192.168.0.30	80	10/12/18 16:28	12	41	400	21	250
106.13.34.4	192.168.0.30	80	10/12/18 16:28	19	32	600	30	150
106.13.34.4	192.168.0.30	80	10/12/18 16:29	8	19	1200	9	200
106.13.34.4	192.168.0.30	80	10/12/18 16:29	2	23	500	15	850
106.13.34.4	192.168.0.30	80	10/12/18 16:29	6	21	300	12	350
106.13.34.4	192.168.0.30	80	10/12/18 16:29	7	23	350	10	250

Resultado:

ID_ORIG_H	ID_RESP_H	ID_RESP_P	TS	DURATION	ORIG_PKTS	ORIG_IP_BYTES	RESP_PKTS	RESP_IP_BYTES	COUNT
106.13.34.4	192.168.0.30	80	10/12/18 16:28	41	79	1700	73	1360	3
106.13.34.4	192.168.0.30	80	10/12/18 16:29	23	86	2350	46	1650	4

Fonte: O autor

O processo acima se repete para preparação dos dados referentes ao protocolo SSH, com uma única diferença no início do procedimento, que deve considerar todos os dados onde “SERVICE” igual a “ssh”.

A preparação dos dados para detectar varredura de porta inicia-se aplicando um filtro no LT_DATA, para selecionar apenas as informações onde o campo “PROTO” igual a “TCP”. Então cria-se outro *dataframe* LT_TCP com o resultado deste filtro, efetuando, assim, o procedimento para totalização e agrupamento dos dados que foram filtrados. Esse processo é exemplificado na Figura 11, na qual as quatro primeiras colunas são utilizadas para o agrupamento das informações, enquanto as demais tem seus valores sumarizados. Ao agrupar os dados pela coluna “TS” é utilizado o formato “dd.MM.yyyy HH”, não considerando os minutos, a coluna “COUNT” é gerada no resultado para armazenar a informação do número de conexões que foram agrupadas.

Figura 8 - Exemplo do processo aplicado sobre os dados do *dataframe* LT_TCP.

ID_ORIG_H	ID_ORIG_P	ID_RESP_H	TS	DURATION	ORIG_PKTS	ORIG_IP_BYTES	RESP_PKTS	RESP_IP_BYTES
106.13.34.4	32422	192.168.0.19	10/12/18 16:27	10	6	700	22	960
106.13.34.4	32422	192.168.0.19	10/12/18 16:28	12	41	400	21	250
106.13.34.4	32422	192.168.0.19	10/12/18 16:28	19	32	600	30	150
106.13.34.4	32422	192.168.0.19	10/12/18 16:29	8	19	1200	9	200
106.13.34.5	65500	192.168.0.19	10/12/18 16:29	2	23	500	15	850
106.13.34.5	65500	192.168.0.19	10/12/18 16:29	6	21	300	12	350
106.13.34.5	65500	192.168.0.19	10/12/18 16:31	7	23	350	10	250

Resultado:

ID_ORIG_H	ID_ORIG_P	ID_RESP_H	TS	DURATION	ORIG_PKTS	ORIG_IP_BYTES	RESP_PKTS	RESP_IP_BYTES	COUNT
106.13.34.4	32422	192.168.0.19	10/12/2018 16	49	98	2900	82	1560	4
106.13.34.5	65500	192.168.0.19	10/12/2018 16	15	67	1150	37	1450	3

Fonte: O autor.

Cada preparação de dados resulta em um *dataframe* sobre o qual aplica-se o algoritmo *K-means.* Tendo em vista que o procedimento descrito abaixo é o mesmo para os três tipos de preparação dos dados, não será aplicada nenhuma distinção durante o processo para destacar as informações de “http”, “ssh” ou “tcp” até o momento do armazenamento dos resultados.

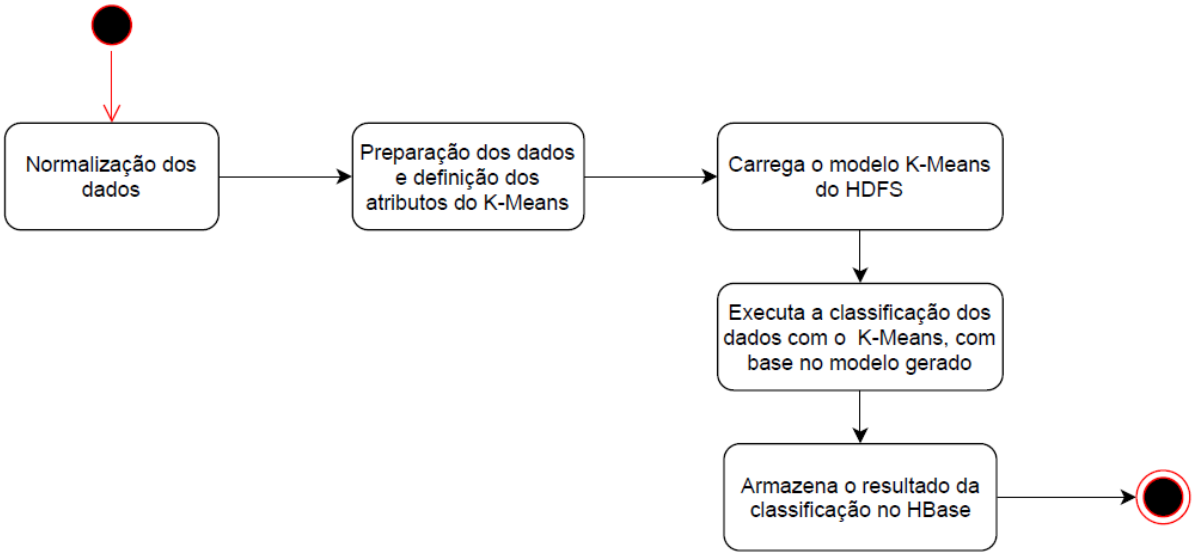
Consideramos que um último *dataframe* nomeado de LT_READY é gerado para armazenar o resultado da preparação dos dados e nele adicionam-se as colunas de “TS_FILTRO” que recebe o filtro utilizado na seleção dos dados da tabela de “LOG”, “TS_CODE” considera o *timestamp* do sistema, “ROW_ID” obtém um identificador atribuído pelo método “*monotonically_increasing_id”,* por fim, ocorre a conversão do campo “TS” para o formato *timestamp.* Utilizar-se-ão os dados deste *dataframe* LT_READY para o processamento com o algoritmo *K-means*, por esse motivo o mesmo é persistido na memória e disco com o método “*persist()”.*

Para a preparação dos dados e do *K-means,* é necessária a realização de algumas etapas, o primeiro passo é a definição do número de *clusters* a serem considerados no algoritmo, neste caso definiu-se dois *clusters*. Seguido pela definição dos campos do *dataframe* que terão seus valores utilizados no algoritmo, com isso o campo “COUNT” é o escolhido, pois possui o número de conexões realizadas. Desta forma é criado o *dataframe* LT_VECTOR, que possui todos os dados de LT_READY, e tem uma coluna a mais denominada “*FEATURES”* a qual armazena os valores de “COUNT” a serem utilizados no algoritmo.

Deste modo, é possível efetuar o treinamento de um modelo *K-means* sobre todos os dados do *dataframe* LV_VECTOR, para isso executa-se o método “*fit”* da classe *KMeansModel* definindo assim os valores dos centroides de cada *cluster*. O modelo resultante é salvo no diretório do HDFS, a definição desse diretório se dá conforme a aplicação do filtro durante a preparação dos dados, com isso cada modelo é salvo em seu respectivo diretório “http”, “ssh” ou “tcp”.

Após a conclusão da primeira etapa ao armazenar o modelo gerado, é iniciada a especificação da segunda parte conforme ilustrado na Figura 9, a qual tem nos seus dois primeiros procedimentos de normalização dos dados e preparação, o mesmo comportamento já descrito anteriormente. Deste modo, começamos a especificação a partir do *dataframe* LV_VECTOR, que é o resultado destes primeiros processos. O terceiro processo carrega o modelo armazenado no HDFS, com o modelo disponível efetua-se o procedimento de classificação dos dados, e por fim, ocorre o último processo que armazena o resultado obtido.

Figura 9 - Etapa de classificação dos dados com o *K-means*.



Fonte: O autor.

Conforme o filtro informado no momento da normalização dos dados, o modelo é selecionado do HDFS, com o mesmo à disposição efetua-se a classificação dos dados do LV_VECTOR. Para isto utiliza-se o método "*transform"* do *KMeansModel*, atribuindo cada linha do *dataframe* ao *cluster* mais próximo, de acordo com o valor do campo "COUNT" em relação ao centroide. Logo salva-se o resultado obtido em um *dataframe* nomeado LT_RES, o qual possui todas as colunas do LV_VECTOR e mais a coluna "PREDICTION", que armazena o *cluster* ao qual a linha é atribuída.

O resultado contido em LT_RES referente aos processamentos com o filtro de "SERVICE = http" ou "SERVICE = ssh" para detectar anomalias de DDoS e força bruta respectivamente, é armazenado na tabela "LOG_KMEANS_DDOS", que tem o detalhamento dos campos exibidos na Tabela 5, os primeiros sete campos foram definidos para compor a *Rowkey*.

Tabela 5 - Tabela LOG_KMEANS_DDOS.

Campos	**Tipo**	**Descrição**
SERVICE	varchar	Serviço
PREDICTION	integer not null	*Cluster* classificado
ID_RESP_H	varchar	Endereço IP que respondeu a conexão
ID_ORIG_H	varchar	Endereço IP que originou a conexão
TS_FILTRO	timestamp not null	*Timestamp* da seleção dos dados

TS_CODE	timestamp not null	*Timestamp* do sistema antes de salvar os dados
ROW_ID	unsigned_long not null	Identificador atribuído a linha do Dataframe
TS	timestamp	Timestamp do BRO referente ao primeiro pacote
COUNT	unsigned_long	Número de conexões agrupadas
ID_RESP_P	unsigned_long	Número da Porta que respondeu a conexão
DURATION	unsigned_long	Duração da conexão
ORIG_PKTS	unsigned_long	Número de pacotes que o IP de origem enviou
ORIG_IP_BYTES	unsigned_long	Número de Bytes que o IP de origem enviou
RESP_PKTS	unsigned_long	Número de pacotes que o IP de resposta enviou
RESP_IP_BYTES	unsigned_long	Número de Bytes que o IP de resposta enviou
CENTROID	varchar[]	Valor dos centroides definidos pelo K-means

Fonte: O autor.

O resultado do processamento para os dados do filtro “PROTO = tcp”, para detecção de varredura de portas é salvo na tabela “LOG_KMEANS_SCAN_PORT”. O seu detalhamento pode ser observado na Tabela 6, onde os sete primeiros campos são definidos para compor a *Rowkey*.

Tabela 6 - Tabela LOG_KMEANS_SCAN_PORT.

Campos	**Tipo**	**Descrição**
PROTO	varchar	Protocolo da conexão
PREDICTION	integer not null	*Cluster* classificado
ID_ORIG_H	varchar	Endereço IP que originou a conexão
ID_RESP_H	varchar	Endereço IP que respondeu a conexão
TS_FILTRO	timestamp not null	*Timestamp* da seleção dos dados
TS_CODE	timestamp not null	*Timestamp* do sistema antes de salvar os dados
ROW_ID	unsigned_long not null	Identificador atribuído a linha do Dataframe
TS	timestamp	Timestamp do BRO referente ao primeiro pacote
COUNT	unsigned_long	Número de conexões agrupadas
ID_ORIG_P	unsigned_long	Número da Porta que originou a conexão
DURATION	unsigned_long	Duração da conexão
ORIG_PKTS	unsigned_long	Número de pacotes que o IP de origem enviou
ORIG_IP_BYTES	unsigned_long	Número de Bytes que o IP de origem enviou
RESP_PKTS	unsigned_long	Número de pacotes que o IP de resposta enviou
RESP_IP_BYTES	unsigned_long	Número de Bytes que o IP de resposta enviou
CENTROID	varchar[]	Valor dos centroides definido pelo K-means

Fonte: O autor.

Todos os dados analisados com este procedimento têm seus resultados armazenados em tabelas, com isso é possível obter os dados para exibição, bem como o histórico dos resultados

por “dd.MM.yyyy HH:mm” para as duas primeiras análises sobre DDoS e força bruta, e até o nível de horas para análise de varredura de portas, e se necessário, aplicar outros processos a tais informações.

3.3.3 Procedimentos aplicados sobre os resultados

Os resultados obtidos de todos os processamentos são armazenados em tabelas, a partir delas as informações são selecionadas para exibição, em alguns casos são aplicados procedimentos para melhorar sua visualização, assim como realizar o enriquecimento destes dados. Para os resultados obtidos referentes às informações do fluxo, nos casos de totais de protocolo e serviço, possui um tratamento para visualizar o número de conexões por linha do tempo em dias e horas, o mesmo processo ocorre sobre os IPs com conexões classificadas como anômalas.

Além disso, também são efetuadas consultas sobre todos os IPs que em algum momento tiveram conexões classificadas como anomalias, assim como são consultados os IPs mais acessados e os que mais acessaram a rede interna. Esse procedimento é detalhado utilizando como exemplo os resultados por protocolo de comunicação, mas o mesmo processo é realizado para todos os dados que são exportados para visualização por linha do tempo o número de conexões realizadas.

Os dados são selecionados da tabela diretamente para um *dataframe,* e com isso são agrupados pelo *timestamp* no formato de “dd.MM.yyyy HH”, sumarizando o número de conexões presentes no campo “COUNT”, em conjunto com esses comandos utiliza-se o método *“pivot”* do *dataframe,* passando o nome da coluna que deve ter suas informações transformadas em colunas, logo o nome da coluna “PROTO” que possui os protocolos foi informada. Portanto, nesse cenário outro *dataframe* com o resultado terá a primeira coluna com a informação de “dd.MM.yyyy HH” e as demais colunas serão todos os protocolos existentes, com informação do total de conexões para aquela hora. Para obter as informações por dia é executado esse mesmo procedimento, mas com o *timestamp* no formato de “dd.MM.yyyy”. Esse processo descrito pode ser observado na Figura 10 com o uso das funcionalidades do *dataframe* do Spark.

Figura 10 - Processo de visualização por linha do tempo em "*dd.MM.yyyy HH"*.

```
lt_count = lt_res.groupBy("dd.MM.yyyy HH")
                 .pivot("PROTO")
                 .sum("COUNT");
```

Fonte: O autor.

Sobre os resultados obtidos na detecção de anomalias é aplicada a visualização por linha do tempo por hora e minuto, considerando os IPs armazenados no campo "RESP_H", esse procedimento é aplicado de forma individual sobre os dados de cada uma das análises de anomalias executadas. Os dados são selecionados diretamente para o *dataframe* nomeado de LT_ANOM, logo para definir quais os IPs que serão selecionados para esse tipo de visualização o seguinte filtro é aplicado, "PREDICTION == 1", considerando apenas as linhas onde foram classificadas como anômalas criando assim o *dataframe* LT_RES.

Com os IPs anômalos definidos é realizado um filtro no LT_ANOM aplicado por meio da funcionalidade "*join"* disponível no *dataframe,* selecionando todas as linhas onde os IPs são iguais aos presentes no LT_RES, independentemente do valor presente no campo "PREDICTION", desta forma obtemos o histórico dos IPs que, em algum momento, foram classificados como anomalia, gerando o *dataframe* LT_HIST.

Com os as informações necessárias salvas no LT_HIST é efetuado o processo de agrupamento dos dados pelo campo "TS" no formato "dd.MM.yyyy HH", sumarização dos valores do campo "COUNT" e aplicado o método "*pivot*", informando o campo "RESP_H". Desse modo o *dataframe* resultante tem a coluna "dd.MM.yyyy HH" mais "*N"* colunas com os números de IPs filtrados, onde cada uma tem a informação do número de conexões realizada por aquele IP em determinada hora. Para obter os resultados por minuto é executado este mesmo processo, utilizando o campo "TS" no formato de "dd.MM.yyyy HH:mm".

Além deste tratamento para visualizar os dados por linha do tempo, são efetuadas análises sobre os IPs de determinados processamentos, com objetivo de agregar um valor maior aos dados, adicionando informações de país, cidade, organização, latitude e longitude por meio da conexão com o *Web Service* IPinfo[8] que efetua consultas de IP retornando estas informações. Como o número de consultas é limitada a mil por dia, desenvolveu-se algumas funcionalidades,

[8] IPinfo: https://ipinfo.io

para efetuar a contagem de pesquisas realizadas no IPinfo por dia e quando atingir o limite não permitir mais consultas, armazenar o IP e o resultado obtido da consulta na tabela IP_INFO que tem seus detalhes exibidos na Tabela 7 e consultar apenas os IPs que não forem encontrados na tabela local. Consultar as informações apenas na tabela "IP_INFO" e executar essas pesquisas a partir de um *dataframe* que possui os IPs a serem consultados, agregando as informações retornadas do *Web Service* no mesmo *dataframe* para que possa ser utilizado em conjunto com os demais campos já existentes.

Tabela 7 - Tabela IP_INFO.

Campos	**Tipo**	**Descrição**
IP	varchar	IP consultado
TS_CODE	timestamp	*Timestamp* do sistema
HOSTNAME	varchar	Hostname do IP
CITY	varchar	Cidade do IP
REGION	varchar	Região
COUNTRY	varchar	País
ORG	varchar	Organização
LATITUDE	double	Latitude
LONGITUDE	double	Longitude

Fonte: O autor.

Com o objetivo de realizar as consultas a partir de um *dataframe* é necessário a utilização do mesmo em conjunto com a biblioteca do IPinfo a qual executa as pesquisas dos IPs. Para efetuar a pesquisa de forma individual por IP, que estão armazenados no *dataframe*, é utilizado o método *"map"* para acessar cada linha e com isso efetuar a consulta no *Web Service*. O resultado de cada IP é armazenado em um objeto do tipo da classe *"cl_Ipinfo"* que possui os mesmos campos retornados do IPinfo, e cada objeto é salvo no *dataframe* LT_IPINFO, que armazena o resultado de todas as consultas utilizando um codificador disponível no Spark para serialização dos objetos em um determinado tipo, nesse caso do tipo da classe *"cl_Ipinfo"*.

Logo com o resultado da consulta dos IPs disponíveis no *dataframe* LT_IPINFO é adicionada uma nova coluna denominada "TS_CODE", que recebe o *timestamp* do sistema e com isso os dados são salvos na tabela "IP_INFO". Para agregar as informações sobre os IPs do LT_IPINFO com o *dataframe* de onde os dados foram selecionados para a consulta, é utilizado o comando *"join"*, deste modo onde o valor do campo "IP" for igual em ambos

dataframes, as informações do LT_IPINFO são adicionadas ao *dataframe* original. Os IPs que serão consultados no *Web Service* são definidos a seguir.

Com os resultados obtidos ao agrupar os campos “ID_ORIG_H” com “ID_RESP_H”, os mesmos são selecionados da tabela “LOG_TOTAIS” onde o campo “TIPO” igual a “orig_h_resp_h” e inseridos no *dataframe* LT_O_R, no qual são aplicados dois filtros para que cada um gere um novo *dataframe* sobre o qual serão executadas as consultas no *Web Service* IPinfo.

O primeiro filtro deve considerar todos os dados onde “ID_RESP_H” possuir o prefixo dos IPs da rede analisada, nesse caso “192.168.”, deste modo todos os dados do LT_O_R que respeitem essa regra serão selecionados e agrupados pelo campo “ID_ORIG_H”, com os demais campos sumarizados, gerando assim um novo *dataframe* responsável por armazenar esse resultado. Então com esses dados efetua-se a consulta no *Web Service,* pesquisando os primeiros mil IPs do campo “ID_ORIG_H” que tiverem o maior número de conexões conforme o campo “COUNT”, o objetivo deste procedimento é ter informações sobre os IPs externos que mais acessam a rede interna.

O segundo filtro a ser aplicado tem como finalidade obter informações sobre os IPs que mais foram acessados por meio da rede interna, com isso o procedimento a ser aplicado é semelhante ao primeiro. Dessa forma, é aplicado o filtro no LT_O_R para considerar todos os dados onde “ID_RESP_H” possuir o prefixo dos IPs diferentes de “192.168.”, com isso um novo *dataframe* é gerado, no qual efetua-se o agrupamento dos dados pelo campo “ID_RESP_H”, com os demais campos sumarizados. A partir do *dataframe* resultante realiza-se a consulta dos IPs no *Web Service,* considerando os primeiros mil que possuírem o maior número de conexões, conforme o campo “COUNT”.

Com relação aos resultados obtidos das análises de anomalias, os mesmos são selecionados das tabelas e todas as informações consideradas anômalas, que possuem o campo “PREDICTION == 1”, tem o valor do campo “ID_ORIG_H” considerado para realizar consulta no *Web Service*.

3.4 SUMÁRIO

Este capítulo descreveu a arquitetura do sistema desenvolvido e todos os processos nela presentes. Primeiramente, apresentou-se a arquitetura e a sua estrutura lógica de funcionamento com o uso das ferramentas selecionadas. Em seguida, abordou-se o processo para realizar o monitoramento, coleta e transferência do tráfego de rede com a ferramenta ZEEK,

especificando as configurações utilizadas, bem como os arquivos de *logs* considerados para processamento. Essa primeira etapa é responsável por gerar o fluxo de dados a ser consumido pelo sistema, por isso o ZEEK realiza a escrita das informações coletadas no tópico do Kafka.

Com os dados disponíveis no Kafka os mesmos são consumidos em um fluxo contínuo pelo Spark *Streaming*, no qual ocorre a manipulação das informações com objetivo de normalizá-las, para isso usa-se *dataframes* que disponibilizam funcionalidades nativas do Spark. Com os dados preparados eles são armazenados no HBase por meio do método *"write"* do *dataframe,* que ao passar as configurações necessárias possui integração com o Phoenix, o qual trabalha diretamente com o HBase.

A última seção deste capítulo abordou os processamentos e análises praticadas sobre os dados coletados da rede. Algumas informações sobre o tráfego de rede são obtidas por meio de agrupamentos de campos e sumarização, na qual é possível identificar protocolos e serviços mais utilizados e gerar sobre estes uma linha do tempo com o número de conexões por dia e hora. Em seguida, sobre os dados aplicou-se as análises para detecção de três tipos de ataques, DDoS sobre o protocolo HTTP, força bruta pelo serviço SSH e varredura de portas com o protocolo TCP. Por fim, sobre os resultados armazenados executaram-se alguns procedimentos para agregar informações sobre os IPs, obtendo sua geolocalização e organização.

4 RESULTADOS EXPERIMENTAIS

Este capítulo apresenta os passos realizados para efetuar a execução do sistema desenvolvido, bem como os recursos de *hardware* utilizados e o *cluster* implementado. São especificados, também, os conjuntos de dados que foram processados, assim como as configurações que foram utilizadas para execução e os resultados obtidos.

O capítulo é dividido em cinco seções. Na primeira (4.1) são detalhados os recursos de *hardware* que foram utilizados e o ambiente que foi implementado para a execução do sistema. Na seção 4.2, são descritos os conjuntos de dados que foram processados, assim como as configurações utilizadas para a execução do sistema sobre cada conjunto. As seções 4.3 e 4.4, trazem os resultados obtidos de cada conjunto de dados que foram processados. Por fim, a seção 4.5 realiza a discussão sobre os resultados experimentais.

4.1 AMBIENTE DE TESTES

Para a implementação do sistema foram utilizadas duas máquinas físicas, cada uma com os seguintes recursos de *hardware*:

- 2 Processadores Intel Xeon E5-2420 1.90 GHz com 12 núcleos (totalizando 24 cores)
- 48 GB de memória RAM
- 1 TB de disco rígido

Com estes recursos disponíveis foram criadas cinco máquinas virtuais (VM's) para utilização na implementação do sistema, todas com o sistema operacional Ubuntu 16.4.3, e os recursos de *hardware* definidos para cada uma são exibidos na Tabela 8.

Tabela 8 - Recursos por Máquina Virtual.

Máquinas	**Memória RAM-GB**	**Disco Rígido-GB**	**Processadores**
Master	15	150	8
Database	8	150	8
Datanode1	25	300	12
Datanode2	32	300	12
Bro	4	50	4

Fonte: O autor.

Com as VM's definidas ocorreu a criação do *cluster* por meio do Apache Ambari[9] que atua como uma ferramenta de gerenciamento, monitoramento e provisionamento de *clusters* para a criação do *cluster* sobre o qual o sistema é implementado. Dessa forma, as quatro primeiras máquinas virtuais listadas na Tabela 8, são utilizadas na criação do *cluster* por meio do Ambari e nessas foram instaladas todas as ferramentas. Com o *cluster* definido, as ferramentas instaladas, configuradas e *online*, são criadas as tabelas no HBase e o tópico no Kafka.

Dentre as VM's criadas apenas a denominada Bro atua de forma separada do *cluste*r, pois nela foi instalado o *The Zeek Network Security Monitor* na versão 2.5.1, bem como o seu *plugin* para comunicação com o Apache Kafka. Essa máquina tem o trabalho exclusivo de executar o ZEEK, para monitorar, coletar e escrever os dados de rede no tópico Kafka.

Ocorreu a implementação de dois aplicativos Spark, o primeiro nomeado de "*integra"* responsável pelo consumo do fluxo de dados a partir do tópico "BroLog", é considerado a aplicação que atua na camada de velocidade da arquitetura Lambda, o segundo aplicativo é nomeado de *"processa"*, pois realiza o processamento de todos os dados que foram coletados e salvos, aplicando análises e armazenando os seus resultados para visualizações posteriores. Logo, a execução de ambos aplicativos pode ocorrer de forma individual ou concomitante, de acordo com o *hardware* disponível para tal.

A execução dos aplicativos ocorre por meio do comando *spark-submit,* no qual alguns parâmetros devem ser preenchidos, como o nome da aplicação que deve ser atribuído ao parâmetro "*name*", o nome da classe e o método principal do código sobre o qual iniciará a execução são informados no campo "*class*". O gerenciador de *cluster* a ser utilizado na execução do aplicativo deve ser preenchido no parâmetro "*master*", que em todos os casos foi informado o gerenciador *Yet Another Resource Negotiator* (YARN), o tipo de execução a ser utilizada local ou em *cluster* no parâmetro "*deploy-mode",* a quantidade de memória a ser atribuída ao *driver* é informada no "*driver-memory"*.

Os parâmetros *"num-executors", "executor-memory"* e *"executor-cores"* recebem, respectivamente, os valores referentes à quantidade de executores a serem utilizados na execução do aplicativo, a memória a ser alocada para cada um destes e a quantidade de *cores* a serem atribuídas a esses executores. Por fim, no último parâmetro denominado "*jars"* devem ser inseridos os nomes de todos os arquivos ".jar" que são utilizados pelo aplicativo, assim como a última informação deve ser o nome arquivo que contém o código da aplicação. O

[9] Apache Ambari: https://ambari.apache.org/

comando Spark-submit é executado na VM Master, a partir do diretório que contém o código da aplicação implementada, bem como todos os arquivos ".jars" que são utilizados.

De acordo com essas informações descritas acima, os valores atribuídos aos parâmetros que definem os recursos a serem utilizados a cada execução, são definidos conforme a disponibilidade dos mesmos. O YARN disponibiliza *virtual cores* para serem utilizadas na execução dos aplicativos, por este motivo são passados nos parâmetros no s*park-submit* um número maior de *cores* do que o sistema suportaria. Logo com o objetivo de não sobrecarregar o sistema, os seguintes recursos tiveram seus valores máximos definidos:

- *driver-memory*: 2 GB
- *num-executors*: 5
- *executor-memory:* 5GB
- *executor-cores:* 9

Dessa forma um total de 27 GB de memória e 45 *cores* são utilizados, pois cada um dos cinco executores definidos recebe os recursos informados em *executor-memory* e *executor-cores*. Esses recursos são alocados nos nós trabalhadores, nos quais as VM's Datanode1 e Datanode2 fazem parte, por isto dentre as VM's essas foram as que se destinou mais recursos.

4.2 METODOLOGIA APLICADA

Para validar o sistema em todos seus aspectos, desde o monitoramento do tráfego de rede até o resultado das análises sobre anomalias a serem detectadas, utilizou-se dados controlados. Dessa forma, os resultados obtidos a partir da execução do sistema foram validados com base nas informações disponíveis sobre o conjunto utilizado.

Nesse estudo foi considerado o conjunto de dados CICIDS2017 disponível pela *University of New Brunswick* (UNB), a descrição das informações que nele contém são apresentadas na subseção 4.2.1, assim como os passos realizados para executar o seu processamento. Após a análise experimental do sistema, o mesmo é utilizado para processar um conjunto de dados previamente coletado de uma rede local de uma instituição de ensino superior (IES), obtendo assim resultados sobre o fluxo de dados de uma rede real, sendo esse processo apresentado na subseção 4.2.2.

Sobre estes dados são realizadas análises de ataques DDoS, força bruta e varredura de porta, os resultados obtidos destas análises são visualizados por totais de conexões, duração, IPs, linha do tempo com o número de conexões realizadas por IP, totais de conexões classificadas como anomalia, número de IPs que efetuaram conexões anômalas, IPs vítimas dos

ataques e totais com o número de IPs e conexões por país e organização. Também são obtidas informações sobre o volume de dados enviados e recebidos, protocolos de aplicação mais utilizados, linha do tempo por dia e hora com o número de conexões por protocolo, aplicações mais acessadas pela rede da instituição com totais de IPs e conexões por organização e país, assim como os IPs que mais acessaram a instituição.

4.2.1 Dados da *University of New Brunswick*

Foi selecionado o conjunto de dados CICIDS2017[10] disponibilizado pela UNB criado no ano de 2017, por possuir um tráfego de rede normal e ataques mais atualizados que se assemelham aos dados verdadeiros, segundo Sharafaldin et al. (2018) a maioria dos conjuntos de dados disponíveis estão desatualizados ou sofrem com a falta de diversidade e volume do tráfego de rede, conforme os resultados obtidos de sua avaliação realizada sobre 11 conjuntos de dados, que foram criados desde 1998. O conjunto selecionado possui a coleta de dados referente a cinco dias, com início às 09:00 horas de segunda-feira, 3 de julho de 2017, terminando às 17:00 horas de sexta-feira, 7 de julho de 2017, sendo que cada dia está salvo em um arquivo ".pcap", sendo possível escolher os dados conforme o dia desejado.

Segunda-feira 03/07, é o único dia que possui o tráfego inteiramente com atividades normais, sem a presença de ataques, enquanto todos os demais sofreram algum tipo de ataque ou anomalia presente na rede. Dos dias disponíveis para o uso três foram selecionados sendo eles, 03/07 com objetivo de agregar um volume maior de dados com tráfego considerado normal, 05/07 que possui diversas anomalias causadas por ataques DoS e o dia 07/07 no qual possui o tráfego de rede com ataques DDoS e diversos tipos de varredura de portas. Os detalhes referentes ao tráfego de rede dos dias 05 e 07, bem como os períodos que foram efetuados os ataques, podem ser observados na Tabela 9.

Dessa forma, temos um conjunto de dados com um volume maior de informações para realizar a análise experimental do ambiente, sendo que de acordo com as análises aplicadas pelo sistema as atividades contidas no arquivo do dia 07 serão detectadas, pois o tráfego de rede possui anomalias geradas pelos ataques DDoS e varredura de portas. Os arquivos foram baixados diretamente na VM Bro, para que o ZEEK efetue a leitura dos mesmos, a soma do tamanho de todos os arquivos ".pcaps" utilizados é de 30 GB.

[10] Conjunto de dados CICIDS2017: https://www.unb.ca/cic/datasets/ids-2017.html

Tabela 9 - Detalhes sobre as anomalias presentes no tráfego de rede selecionado.

Dia	Período	Atacante	Vítima	Descrição
05/07/2017	09:47-10:10	172.16.0.1	192.168.10.50	DoS Slowloris
	10:14-10:35	172.16.0.1	192.168.10.50	DoS Slowhttptest
	10:43-11:00	172.16.0.1	192.168.10.50	DoS Hulk
	11:10-11:23	172.16.0.1	192.168.10.50	DoS GoldenEye
	15:12-15:32	172.16.0.11	192.168.10.51	Heartbleed Port 444
07/07/2017	10:02-11:02	172.16.0.1	192.168.10.15 192.168.10.14 192.168.10.9 192.168.10.8 192.168.10.5	Botnet ARES
	14:51-14:53	172.16.0.1	192.168.10.50	Port Scan sS
	14:54-14:56	172.16.0.1	192.168.10.50	Port Scan sT
	14:57-14:59	172.16.0.1	192.168.10.50	Port Scan sF
	15:00-15:02	172.16.0.1	192.168.10.50	Port Scan sX
	15:03-15:05	172.16.0.1	192.168.10.50	Port Scan sN
	15:06-15:07	172.16.0.1	192.168.10.50	Port Scan sP
	15:08-15:15	172.16.0.1	192.168.10.50	Port Scan s
	15:16-15:18	172.16.0.1	192.168.10.50	Port Scan as
	15:19-15:21	172.16.0.1	192.168.10.50	Port Scan s
	15:22-15:24	172.16.0.1	192.168.10.50	Port Scan sR
	15:25	172.16.0.1	192.168.10.50	Port Scan sL
	15:26-15:27	172.16.0.1	192.168.10.50	Port Scan sl
	15:28-15:29	172.16.0.1	192.168.10.50	Port Scan b
	15:56-16:16	172.16.0.1	192.168.10.50	DDoS LOIT

Fonte: O autor.

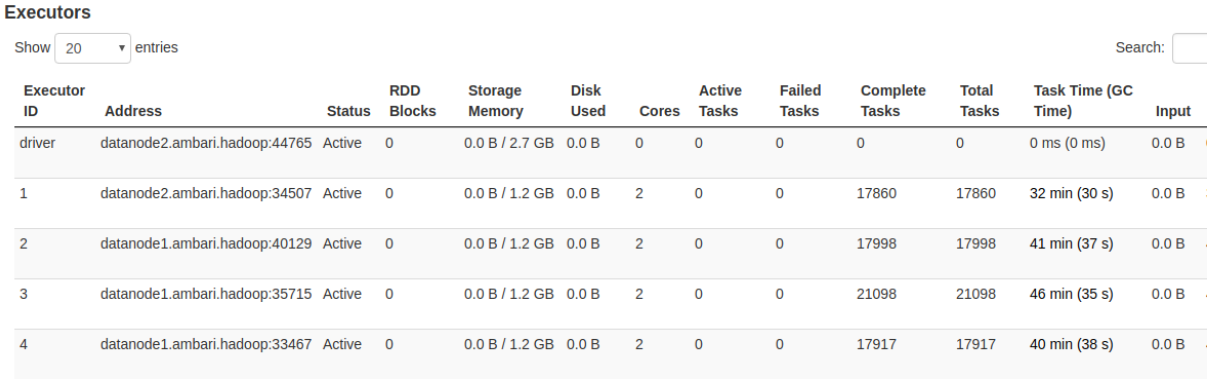
Antes de iniciar a leitura dos arquivos foi executado o aplicativo *"integra"* para que o mesmo já possa trabalhar com os dados conforme eles são escritos no Kafka, desta forma todo o processo de consumo, normalização e armazenamento do fluxo de dados gerados pelo ZEEK está ativo. A leitura dos arquivos ".pcap" é realizada na VM Bro por meio do comando "bro -r <nome do arquivo> local", como são três arquivos, esse comando é executado para cada um deles, a última informação desse comando "local" se refere ao *script* que deve ser carregado, o *script* "local" pode ser considerado o raiz pois é no qual todos os demais *scripts* são chamados quando o ZEEK é iniciado.

Desta forma, as informações foram lidas pelo ZEEK e, já, escritas no tópico do Kafka. Após a conclusão da leitura de todos os arquivos, o aplicativo *"integra"* foi finalizado, pois não teria mais dados a serem consumidos.

Ao verificar as informações do aplicativo executado na Figura 11, podemos observar os executores que foram definidos, bem como o número de tarefas que cada um realizou e o tempo consumido por cada executor, a partir desses valores temos um total de 74.843 tarefas que foram executadas em um tempo de 02h38min. Foram realizados quinhentos processos de escrita no HBase, no que resultou em um total de 1,05 GB de dados armazenados na tabela e quatro regiões criadas. Uma configuração realizada no aplicativo apenas durante a análise experimental do sistema, impactou diretamente no tempo para efetuar o processo de consumo, normalização e armazenamento dos dados pelo Spark *Streaming*, foi o tempo de dez segundos definido como intervalo para efetuar a leitura das mensagens do tópico Kafka. Deste modo, foram criadas mais tarefas para execução, pois quanto menor o intervalo de tempo mais tarefas são criadas com um número menor de informações a processar.

Com todos os dados disponíveis, executou-se o aplicativo *"processa"*, responsável por efetuar o processamento e análise dos dados coletados, essa aplicação foi executada três vezes, sendo que cada uma delas efetuou um tipo de procedimento sobre os dados. A primeira processa os totais obtendo as informações sobre o tráfego de rede, a segunda e a terceira execução aplicam as análises com o algoritmo *K-means* para detectar anomalias referentes a ataques DDoS e varredura de portas respectivamente. Os recursos utilizados para cada uma destas execuções foram: *driver-memory*: 2 GB; *num-executors*: 5; *executor-memory:* 2,5 GB; *executor-cores:* 2.

Figura 11 - Informações sobre os executores utilizados nos dados da UNB.



Executors

Show 20 entries Search:

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input
driver	datanode2.ambari.hadoop:44765	Active	0	0.0 B / 2.7 GB	0.0 B	0	0	0	0	0	0 ms (0 ms)	0.0 B
1	datanode2.ambari.hadoop:34507	Active	0	0.0 B / 1.2 GB	0.0 B	2	0	0	17860	17860	32 min (30 s)	0.0 B
2	datanode1.ambari.hadoop:40129	Active	0	0.0 B / 1.2 GB	0.0 B	2	0	0	17998	17998	41 min (37 s)	0.0 B
3	datanode1.ambari.hadoop:35715	Active	0	0.0 B / 1.2 GB	0.0 B	2	0	0	21098	21098	46 min (35 s)	0.0 B
4	datanode1.ambari.hadoop:33467	Active	0	0.0 B / 1.2 GB	0.0 B	2	0	0	17917	17917	40 min (38 s)	0.0 B

Fonte: O autor.

Após essas execuções os resultados estão salvos nas tabelas, logo ocorre a importação desses valores para visualização dos resultados obtidos.

4.2.2 Dados coletados de Instituição de Ensino Superior

Com a finalidade de criar um conjunto de dados com o tráfego de rede real, foi realizado o monitoramento de uma rede local da Universidade Federal de Santa Maria (UFSM) por meio da ferramenta TCPDUMP[11], a qual cria arquivos ".pcap" com todas as informações coletadas da rede. O período da coleta foi de 04/12/2018 às 14:00 horas até o dia 11/12/2018 às 14:00 horas, totalizando oito dias de tráfego de rede, sendo um total de 148 GB de dados coletados e armazenados em arquivos ".pcap", os quais foram lidos pelo ZEEK.

Antes do ZEEK consumir o tráfego de rede, o aplicativo *"integra"* foi iniciado pelo comando "spark-submit" definindo os seguintes recursos para a sua execução, *driver-memor = 2GB*, *num-executors* = 5, *executor-memory* = 5GB e *executor-cores* = 2, para que ocorra o consumo dos dados conforme eles são escritos no Kafka, após esse comando iniciou-se a leitura dos arquivos pelo ZEEK.

Após todos os arquivos ".pcap" lidos, foram obtidas as informações do Spark referentes a execução do aplicativo, sendo um total de 3.421 trabalhos com 23.346 tarefas executadas em 2h40min. Esses arquivos resultaram em 6,82 GB de informações armazenadas na tabela "LOG".

Para realizar o processamento de todas essas informações, foi necessário dividir os dados, pois como a aplicação mantém todos os dados selecionados da tabela em memória, seria inviável a sua execução considerando toda a tabela. A divisão dos dados realizada considera na primeira execução de cada procedimento todas as informações onde o campo "TS" menor que 09/12/2018, enquanto a segunda execução seleciona os dados onde "TS" maior ou igual 09/12/2018.

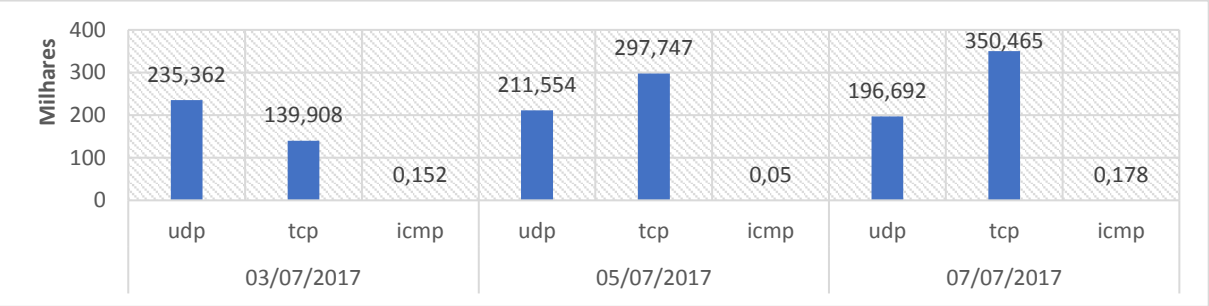
Além desta divisão de dados, o procedimento que calcula os totais é executado de forma individual, assim como a análise para detecção de varredura de portas, a análise de DDoS sobre HTTP e força bruta SSH são executadas na mesma aplicação. Desta forma, o aplicativo "*processa*" é executado três vezes conforme a divisão por procedimento descrito. Após todas as execuções, os resultados são importados das tabelas para serem visualizados.

[11] TCPDUMP: https://www.tcpdump.org

4.3 RESULTADOS DA UNB

Os três arquivos ".pcap" utilizados totalizaram 1.432.108 de conexões, processadas pelo aplicativo *"integra"* em um intervalo de tempo de uma hora e três minutos. Esse número de conexões está dividido em três protocolos, ICMP com 380, TCP possui 788.120 e UDP com 643.608, a distribuição das conexões por dia é representada na Figura 12, na qual pode se observar um volume maior de conexões TCP no dia sete, no qual ocorre o ataque DDoS e varredura de portas.

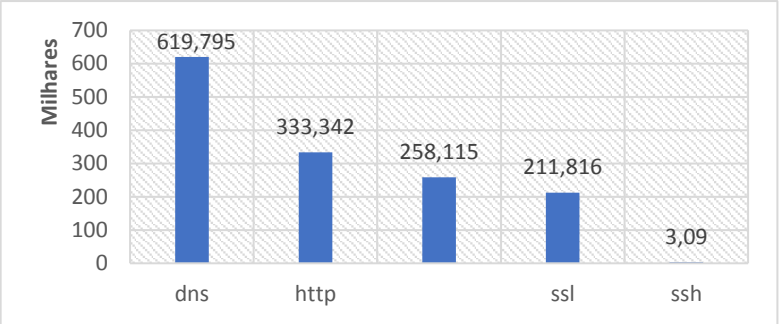
Figura 12 - Totais de conexões por dia e protocolo – UNB.



Fonte: o autor.

Deste total de conexões foram identificados oito tipos de protocolos da camada de aplicação, conforme classificação realizada pelo ZEEK, são eles, DNS, HTTP, SSL, SSH, *Kerebos* (KRB), FTP, *Distributed Computing Environment/Remote Procedure Calls* (DCE_RPC) e *Datagram Transport Layer Security* (DTLS). Nos valores apresentados na Figura 13, estão presentes apenas os protocolos de aplicação que obtiveram mais de três mil conexões, o maior número foi DNS, seguido por HTTP e outros. Um ponto a ser observado é o número de conexões que não tiveram identificação do protocolo utilizado, pois representam portas altas acima de 1024.

Figura 13 - Conexões por protocolo de aplicação – Dados UNB.



Fonte: o autor.

4.3.1 Ataques DDoS

Conforme abordado no capítulo 3, para detecção de ataques DDoS foi considerado o protocolo HTTP para aplicar o processamento com o algoritmo *K-means*. O total de conexões computadas foram de 333.342 das quais 248.088 foram classificadas como anômalas, o que representa 74% de todas as conexões, sendo que um total de 14 IPs foram analisados e em todos esses casos o ataque ocorre de um único IP de origem que é tratado pelo *firewall* com valor de 172.16.0.1, a vítima tem o IP 192.168.10.50. Se considerar o total de conexões realizadas apenas pelo IP atacante temos 261.748, sendo que destas, apenas, 13.660 não foram classificadas como anômalas.

Tiveram dois períodos de atividade do IP 172.16.0.1 que foram classificados como anomalia, o seu primeiro caso ocorreu no dia 05/07 às 10:43 até as 11:00, esses dezessete minutos de atividades foram caracterizadas como anomalia. O segundo caso foi identificado no dia 07/07 das 15:57 até as 16:15, um total de dezoito minutos de atividades com características de ataque DDoS.

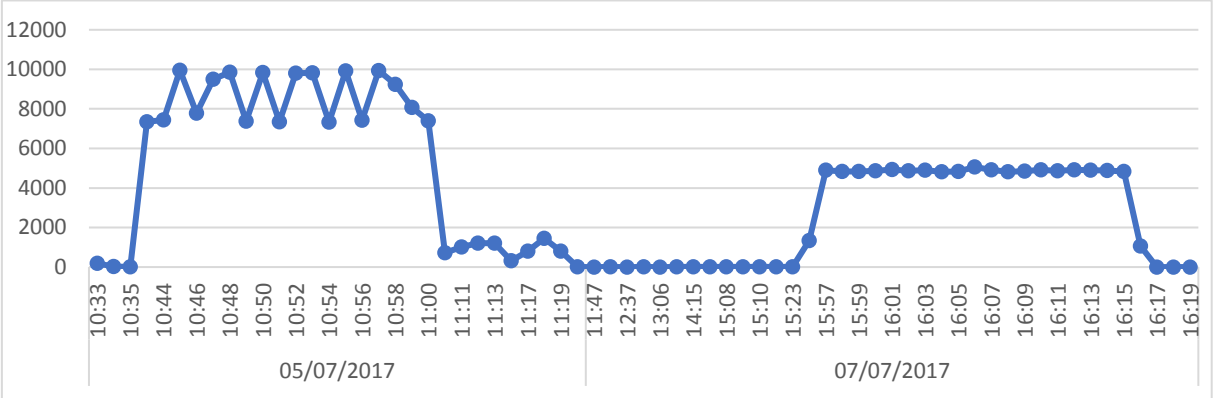
Com o intuito de verificar o minuto no qual se iniciou as anomalias, a Figura 14 traz a linha de tempo referente ao IP 172.16.0.1 por dia, hora e minuto, pois conforme as informações sobre o conjunto de dados, o ataque DDoS iniciou-se às 15:56 e termina às 16:16, em vista disso seriam vinte minutos consecutivos de conexões classificados como ataques DDoS, desta forma ocorreu uma diferença de dois minutos entre o tráfego que foi classificado como anômalo pelo sistema e o resultado esperado. Nesse caso, há uma diferença de 2395 conexões que não foram classificadas como anomalia, referentes ao número de conexões realizadas das 15:56 às 16:16. Conforme pode ser observado na Figura 14, o intervalo classificado como anômalo tem

uma variação do número de conexões por minuto muito pequena, variando de 4811 a 5055 conexões.

Outros detalhes que podem ser observados na Figura 14 é que o ataque DoS praticado das 10:43 até as 11:00 do dia 05/07, foi o intervalo de tempo com o maior número de conexões registradas, variando de 7343 até 9955 por minuto, sendo que todas as conexões deste período foram classificadas como anomalias pelo sistema. O período que teve um número de conexões consideradas normais iniciou-se às 11:47 do dia 07/07 até as 15:23, possuindo de uma a quinze conexões por minuto.

Os valores dos centroides definidos pelo algoritmo *K-means* foram de 2,63 e 6705,08, como é possível observar, são valores muito distintos. Todas as conexões atribuídas ao segundo centroide são consideradas anomalias.

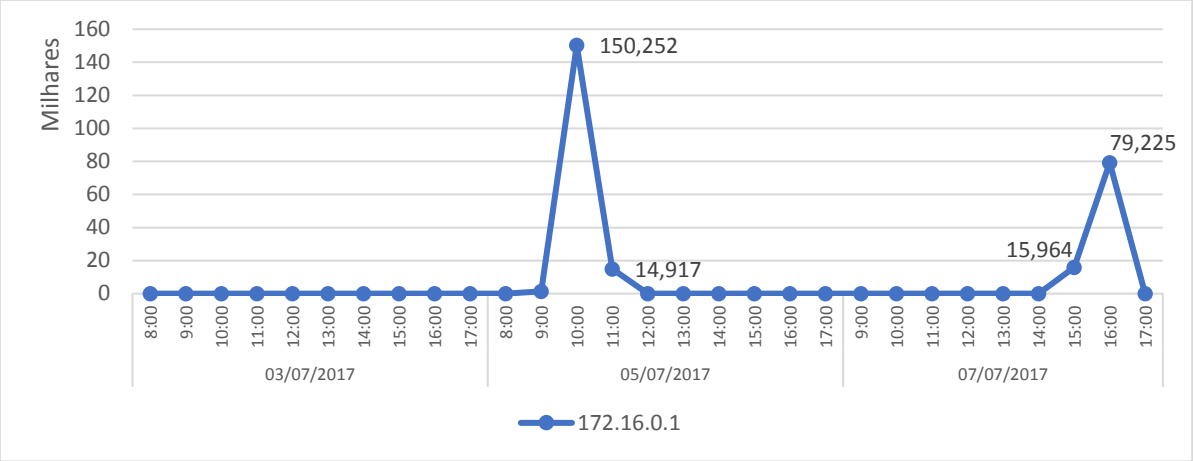
Figura 14 - Número de conexões por minuto do IP 172.16.0.1 – Dados UNB.



Fonte: o autor.

Com a Figura 15 é possível observar a linha do tempo por dia e hora com o número de conexões realizadas pelo IP de origem que efetuou os ataques. É possível identificar que no dia 05/07 às 10:00 ocorreu o maior pico do número de conexões com um total de 150.252, já no dia 07/07, o qual tem as anomalias detectadas pelo sistema, é possível visualizar que às 15:00 o IP apresentou o início de um pico de conexões com 15.964 e às 16:00 atingiu o máximo com 79.225 conexões.

Figura 15 - Linha do tempo com o número de conexões por dia e hora do IP 172.16.0.1 – Dados UNB.



Fonte: o autor.

4.3.2 Varredura de portas

O volume de dados analisado para detectar varredura de portas foi consideravelmente maior do que o da análise anterior, pois foram consideradas as 788.120 conexões TCP coletadas. O resultado ao aplicar o algoritmo *K-means* foi de 78.870 conexões classificadas como varredura de portas, todas referentes ao dia 07/07 a partir das 14:51 com variações entre conexões normais e anormais até as 15:23. Os mesmos IPs atacante 172.16.0.1 e vítima 192.168.10.50 identificados na análise DDoS, são considerados nessas respectivas funções nos resultados apresentados.

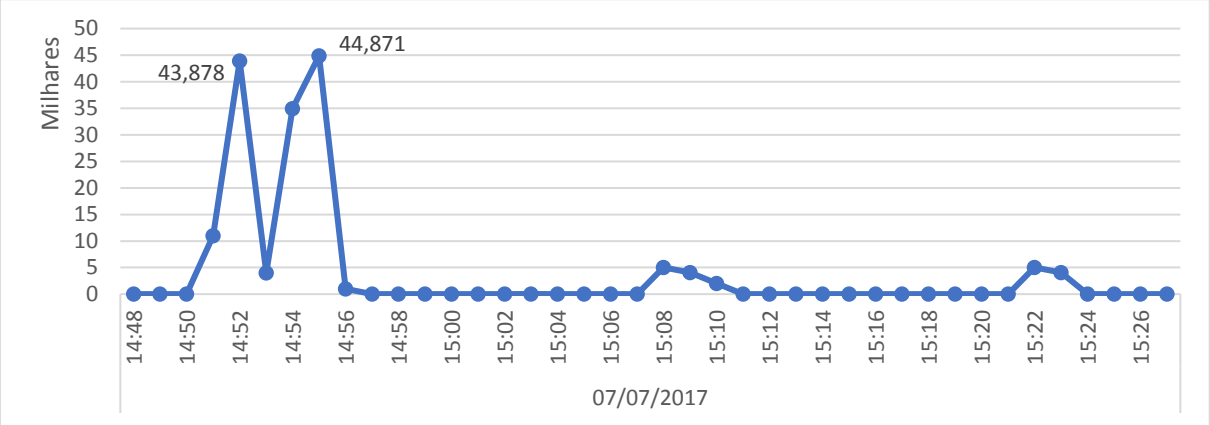
Nessa análise, os valores definidos para os dois centroides pelo *K-means* foram de 1,11 e 997,08, logo consideramos as conexões que forem classificadas ao segundo centroide como varredura de porta, nesse caso três períodos do tráfego de rede foram identificados com as características de varredura, sendo eles, 14:51 às 14:53, 15:08 às 15:10 e 15:22 às 15:23, em todos casos conforme mencionado acima o IP 172.16.0.1 é o atacante e o IP 192.168.10.50 a vítima. A média de conexões por minuto por IP de origem e porta para um mesmo destino nesses períodos é de 997, enquanto que para essas mesmas condições as conexões consideradas normais possuem uma média de 1,45.

A Figura 16 apresenta a linha do tempo com o fluxo de conexões por minuto considerando apenas o IP de origem responsável pelo ataque, logo todas as conexões referentes ao IP naquele minuto foram sumarizadas independente da porta de origem utilizada, por esse motivo os valores são altos, mesmo assim, é possível identificar uma variação maior nos períodos

anômalos. Não são apresentados os demais IPs juntamente na figura, pois seria imperceptível a diferença do número de conexões, já que dos IPs classificados no primeiro centroide, o de maior valor apresenta 383 conexões em determinado minuto. Considerando apenas esses três períodos identificados pelo sistema, foram 79 portas de origem utilizadas pelo IP 172.16.0.1 para efetuar 78.870 conexões a vítima.

Podemos observar na Figura 16 que o primeiro período classificado como varredura de porta inicia às 14:51 até as 14:53, com o seu auge às 14:52 com 43.878 conexões. O período das 14:54 até 14:56 não foi classificado como anômalo pelo sistema, mas apresenta um grande número de conexões que foram realizadas a partir do IP 172.16.0.1, com um auge de 44.871 conexões às 14:55.

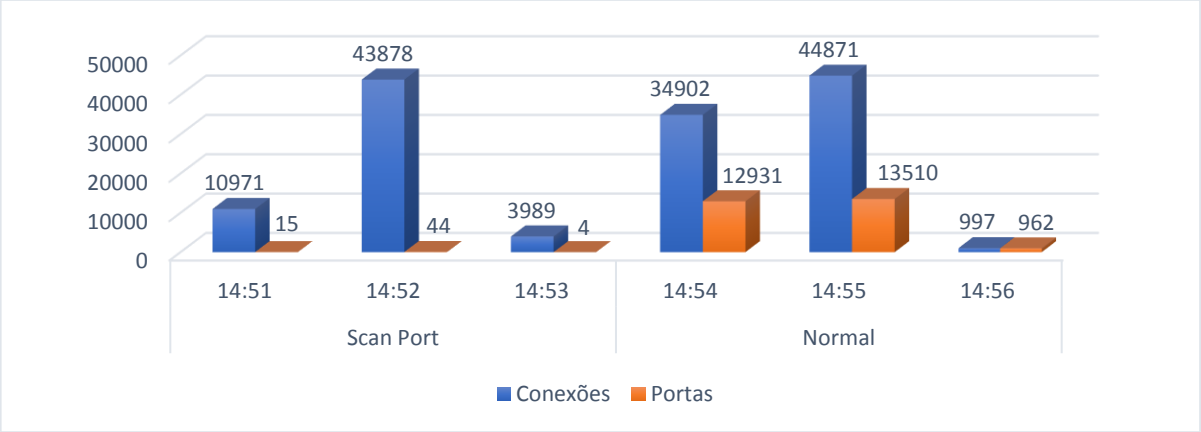
Figura 16 - Linha do tempo do IP 172.16.0.1 que realizou varredura de porta – Dados UNB.



Fonte: o autor.

Isso ocorre, pois nesse intervalo de tempo realizou-se um total de 80.770 conexões, sendo que a média de conexões considerando o IP de origem e a porta para um mesmo destino, foi de 2,92 conexões, logo pode considerar-se que ocorreu o uso de um grande número de portas pelas quais o IP 172.16.0.1 iniciou a conexão, nesse caso foi um total de 14.067 portas distintas utilizadas nesse período. Portanto, com a Figura 17, é possível observar a diferença do número de portas distintas utilizadas por minuto no período das 15:51 até as 15:54, bem como o número de conexões que foram realizadas.

Figura 17 - Número de conexões realizadas pelo IP 172.16.0.1 por minuto e a quantidade de portas utilizadas – Dados UNB.



Fonte: o autor.

A partir da Figura 17 é possível observar que o número de conexões realizadas utilizando um baixo número de portas origem por minuto foram classificadas como anômalas, enquanto que ao realizar conexões com o uso de mais portas o sistema não classifica como anomalias, pois o volume de conexão por IP e porta de origem é menor.

4.3.3 Tempo de processamento

A execução do aplicativo “processa” sobre os dados de testes foi realizada apenas uma vez para cada tipo de análise, sendo eles, processar os totais, análise DDoS e detecção de varredura de portas, pois o volume de dados é suportado pelos recursos que estavam sendo utilizados. De forma mais específica, foi possível carregar todos os dados a serem utilizados pelo Spark para a memória, efetuando o processamento de modo paralelo a partir deles.

Os tempos de processamento de cada execução são apresentados a seguir em “mm:ss”, sendo 13:54 para calcular os totais, a análise de varredura de porta sobre todas as conexões consumiu 08:33 e a detecção de ataques DDoS sobre o protocolo HTTP levou 03:57 sendo esse o menor tempo de execução. O tempo total de processamento destes dados foi de 26:24.

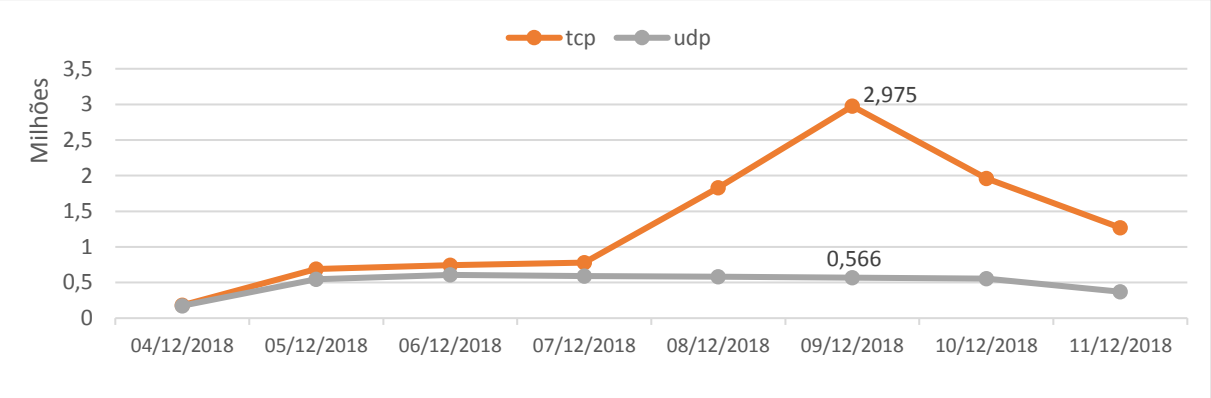
4.4 RESULTADOS DE INSTITUIÇÃO

Os oito dias de coleta de dados resultaram em um total de 14.520.161 conexões, distribuídas em três protocolos TCP com 10.431.040 sendo o mais utilizado, seguindo pelo UDP com um total de 3.995.577 e por último encontra-se o ICMP com 93.544 conexões. A

distribuição destas conexões ao longo dos oito dias de tráfego monitorados pode ser observada na Figura 18, que não considera o protocolo ICMP pois não seria possível identificar informações sobre o mesmo, por possuir um número baixo em relação aos outros. Isso ocorre, pois, o *firewall* filtra os pacotes deste protocolo, contudo destaca-se seu ápice de utilização no dia 09/12/2018 com 15.721 conexões.

O número de conexões realizadas pelos protocolos TCP e UDP possuem uma média muito semelhante nos dias 04, 05, 06 e 07, entretanto nos demais dias isso não ocorre, pois o protocolo TCP tem um aumento considerável de conexões após o dia 07 chegando em seu auge no dia 08 com quase três milhões de conexões, enquanto o protocolo UDP manteve média estável durante todos os dias.

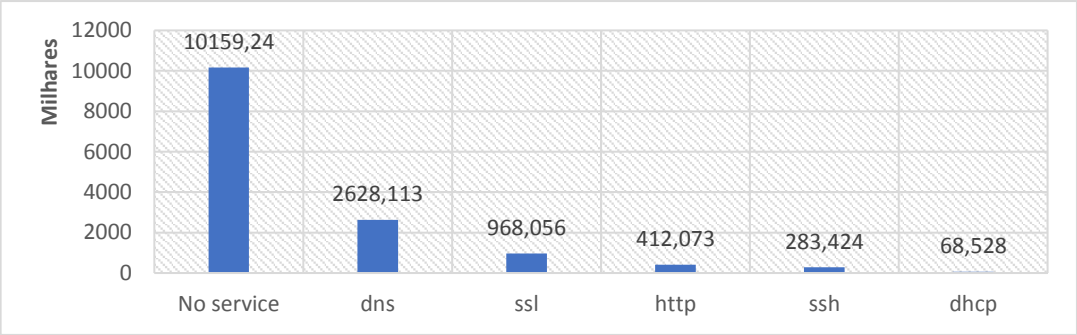
Figura 18 - Conexões por dia e protocolo – Dados IES.



Fonte: o autor.

Sobre essas conexões TCP e UDP coletadas, foram identificados pelo ZEEK dez tipos de protocolos da camada de aplicação, sendo eles, SSL, HTTP, DNS, SSH, DHCP, *Extensible Messaging and Presence Protocol* (XMPP), DCE_RPC, *NT LAN Manager* (NTLM), KRB e *Simple Network Management Protocol* (SNMP). As conexões que não tiveram protocolo de aplicação identificado representam portas altas acima de 1024, essas se destacam por serem a maioria, com mais de dez milhões de conexões. Conforme pode ser observado na Figura 19, a qual exibe os protocolos com mais de sessenta mil conexões, bem como o agrupamento de todas as conexões que não tiveram o protocolo identificado, logo por esse caso possuir mais conexões teve um volume maior de dados recebidos e enviados sendo 5 GB e 30 GB respectivamente, e sobre a qual teve o maior tempo de conexões totalizando 27.280 horas.

Figura 19 - Conexões por serviço – Dados IES.

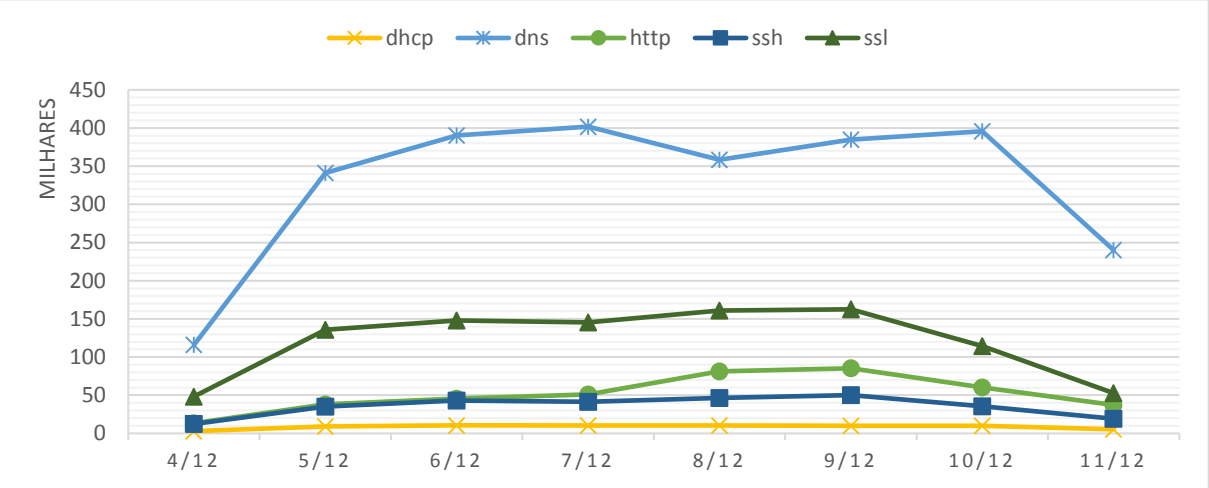


Fonte: o autor.

Os serviços identificados tiveram um volume de dados enviados e recebidos bem abaixo em relação as conexões que não tem identificação. Sendo que foi por meio do SSL que mais ocorreu o envio e recebimento de dados, totalizando 19,54 GB e 4 GB respectivamente, seguido por HTTP com 12,74 GB enviados e 579 MB recebidos, enquanto os demais serviços tiveram valores inferiores a 500 MB tanto de envio quanto de recebimento.

A linha do tempo por dia dos cinco serviços com o maior número de conexões, pode ser observada na Figura 20, na qual foi omitido a linha das conexões que não possuem serviço, por apresentar um volume grande de informações tornando as demais imperceptíveis. É possível concluir que todos mantém uma média de utilização por dia, sem significativas variações.

Figura 20 - Utilização dos serviços por dia – Dados IES.



Fonte: o autor.

Foi por meio das conexões realizadas sem a identificação do serviço, que ocorreu o aumento significativo da atividade do protocolo TCP apresentados nos dias 08, 09, 10 e 11 na Figura 18.

4.4.1 Anomalias de ataques DDoS

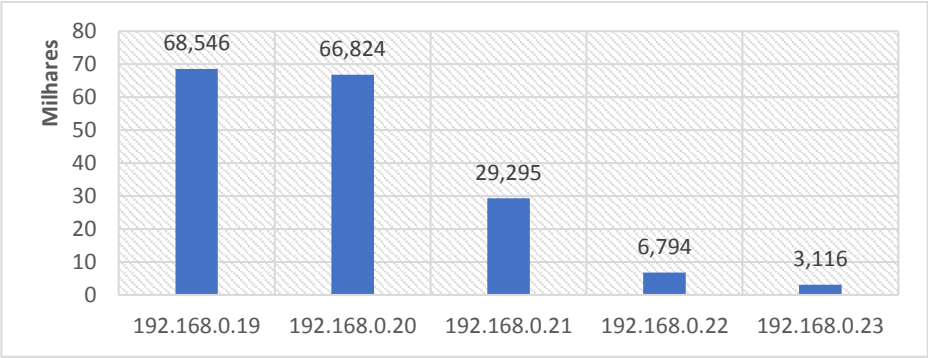
Considerando apenas o protocolo HTTP para essa análise, foram processadas 412.073 conexões, realizadas por 4.338 IPs distintos, esses IPs utilizaram um total de dez portas para iniciar a conexão, que foram identificados pelo campo ID_ORIG_P do ZEEK. A análise considerou o campo ID_ORIG_H, portanto esse total de IPs engloba internos, assim como IPs externos que se conectam à rede da instituição. Deste modo, a análise pode identificar tanto anomalias com características DDoS praticadas contra a rede interna, como as praticadas por ela.

Deste total de conexões HTTP analisadas, 181.520 foram classificadas como anômalas, o que representa 44,05% de todas as conexões realizadas por esse protocolo, das quais foram identificados 102 IPs distintos responsáveis por iniciar a conexão, que no caso podem ser denominados como atacantes, originados de dez países Brasil (BR), Canadá (CA), China (CN), Egito (EG), Reino Unido (GB), Hong Kong (HK), Romênia (RO), Arábia Saudita (SA), Seicheles (SC) e Estados Unidos (US).

Os centroides obtidos pelo algoritmo *K-means* sobre essa análise foram de 1,86 e 21,24, logo as conexões atribuídas ao segundo valor se caracterizam como anomalias. Esses valores foram obtidos quando foi realizado o treinamento do modelo *K-means,* sobre todas as conexões HTTP.

Destes 102 IPs responsáveis pelos ataques, 45 são IPs internos, os quais são responsáveis por 179.549 conexões classificadas como anômalas, direcionadas a 51 IPs distintos. Os cinco IPs internos que realizaram mais de mil conexões classificadas como anomalia podem ser observados na Figura 21, na qual os dois primeiros possuem valores semelhantes e uma diferença significativa para o terceiro IP.

Figura 21 - Número de conexões originadas por IPs internos classificados como anomalia – Dados IES.



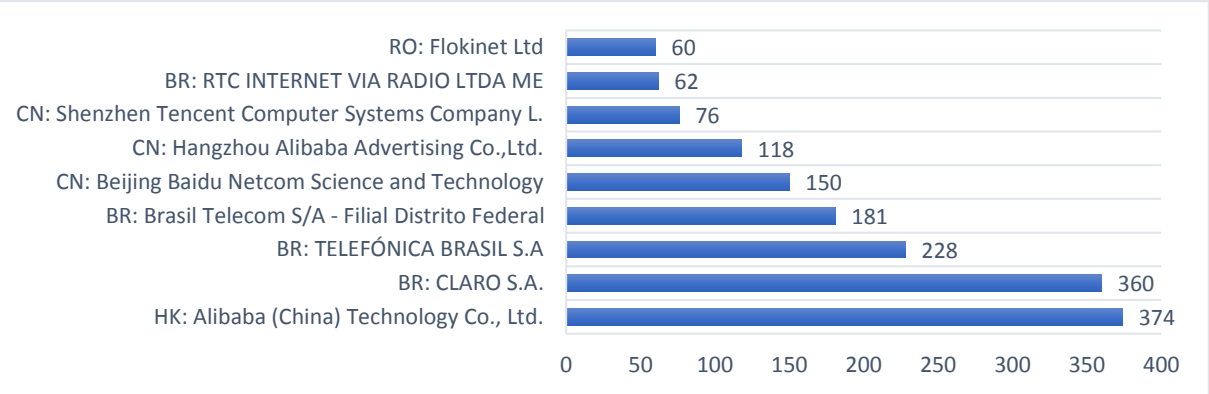
Fonte: o autor.

Ao analisar as conexões anômalas realizadas pelo IP 192.168.0.19, foi identificado que três são os IPs de destino, sendo que dois recebem doze conexões cada, e o terceiro IP 178.62.242.235 é que concentra todo o volume de conexões com um total de 68.522, o total de dados enviados e recebidos por esse IP de origem é de 53 MB e 64 MB respectivamente. O segundo IP com mais conexões teve vinte e seis IPs de destino, enviando um total de 55 MB e recebendo 268 MB de dados sendo o IP que mais enviou e recebeu informações.

Ao considerar apenas os IPs externos que se conectaram à rede local são 57 IPs distintos, que efetuaram 1.971 conexões classificadas como anômalas sobre 9 IPs de destino, o que representa 1,1% de todas as conexões HTTP classificadas como anomalia. Para identificar o número de IPs e conexões por organização essas informações foram agrupadas, identificando assim 25 organizações, das quais a com o maior número de IPs é a Claro S.A. com um total de treze, seguido por Telefónica Brasil e Brasil Telecom, respectivamente, com dez e sete IPs, as demais obtiveram três ou menos IPs por organização.

Também foram identificadas as organizações responsáveis pelo maior número de conexões anômalas. Esse resultado pode ser observado na Figura 22, que possui nas duas primeiras letras a abreviação do país correspondente.

Figura 22 - Nº de conexões HTTP por organização classificadas como anomalia – Dados IES.



Fonte: o autor.

4.4.2 Ataque de força bruta sobre o serviço SSH

O processamento para identificar tentativas de acesso com conexões SSH, analisou um total de 238.424 conexões, originadas por 3.744 IPs distintos que efetuaram conexões tendo como destino 19 IPs. Na Tabela 10 estão presentes os onze IPs externos que mais realizaram conexões pelo serviço SSH no período analisado. Destes totais, obtivemos 29.266 conexões classificadas como anomalias, o que representa 12,25% de todas as conexões SSH, as quais foram geradas por 118 IPs de origem com tentativas de acesso a sete IPs da rede. Os valores dos centroides definidos pelo *K-means* nessa análise foram de 1,98 e 8,57, logo todas as conexões atribuídas ao segundo *cluster* são classificadas como anomalias.

Tabela 10 - IPs externos que mais realizaram conexões SSH – Dados IES.

IP de Origem	País	Organização	Conexões
218.92.1.136	CN	CHINANET-BACKBONE	42984
200.178.102.24	BR	CLARO S.A.	31110
200.195.171.75	BR	COPEL Telecomunicações S.A.	31002
200.195.171.74	BR	COPEL Telecomunicações S.A.	30974
58.242.83.32	CN	CHINA UNICOM China169 Backbone	13787
218.92.1.155	CN	CHINANET-BACKBONE	11014
150.165.85.5	BR	Associação Rede Nacional de Ensino e Pesquisa	10522
58.242.83.36	CN	CHINA UNICOM China169 Backbone	8365
185.246.128.25	SE	ICME LIMITED	5034
193.201.224.218	UA	PE Tetyana Mysyk	4032
200.20.164.160	BR	Fundação Carlos Chagas Filho de Amparo a Pesquisa	3060

Fonte: o autor.

Podemos observar na Tabela 11 os IPs que efetuaram o maior número de conexões classificadas como anomalias, em direção a rede interna. O IP 223.111.182.119 chegou a realizar em determinado minuto 102 conexões, enquanto o IP com o maior número de conexões com 2.730 manteve sempre a mesma média de seis conexões por minuto.

Tabela 11 - IPs externos com o maior número de conexões SSH classificados como anomalia – Dados IES.

IP de Origem	País	Organização	Conexões
200.178.102.24	BR	CLARO S.A.	2730
200.195.171.74	BR	COPEL Telecomunicações S.A.	2688
200.195.171.75	BR	COPEL Telecomunicações S.A.	2682
185.246.128.25	SE	ICME LIMITED	2080
193.201.224.218	UA	PE Tetyana Mysyk	925
222.233.52.185	KR	SK Broadband Co Ltd	889
150.165.85.5	BR	Associação Rede Nacional de Ensino e Pesquisa	888
115.238.245.2	CN	CHINANET-BACKBONE	818
223.111.182.119	CN	China Mobile communications corporation	786
122.226.181.164	CN	DaLi	770
61.184.247.4	CN	CHINANET-BACKBONE	732
118.123.15.142	CN	CHINANET-BACKBONE	714

Fonte: o autor.

O número total de conexões SSH recebidas por cada IP da instituição é apresentada na Tabela 12, assim como o número de IPs externos que tentaram conexões sobre cada um. O que pode ser observado é o volume significativo de conexões e IPs, que realizaram tentativas de acessos.

Tabela 12 - Total de conexões SSH recebida por IP da instituição.

IP	Conexões Recebidas	Nº de IP's Externos
192.168.0.24	133188	1662
192.168.0.25	34051	1491
192.168.0.26	29887	2059
192.168.0.27	29436	2113
192.168.0.28	29126	2092
192.168.0.29	16025	333
192.168.0.30	11652	1519

Fonte: o autor.

A Tabela 13 apresenta as informações referentes aos IPs da instituição que receberam tentativas de conexões SSH classificadas como anômalas, contém o número IPs distintos que tentaram conexão sobre cada um e o total de conexões anômalas recebidas. Os IPs internos que mais receberam conexões classificadas como anômalas foram, 192.168.0.24 com um total de 16.503 e 192.168.0.25 com 10.110, seguido pelos demais IPs.

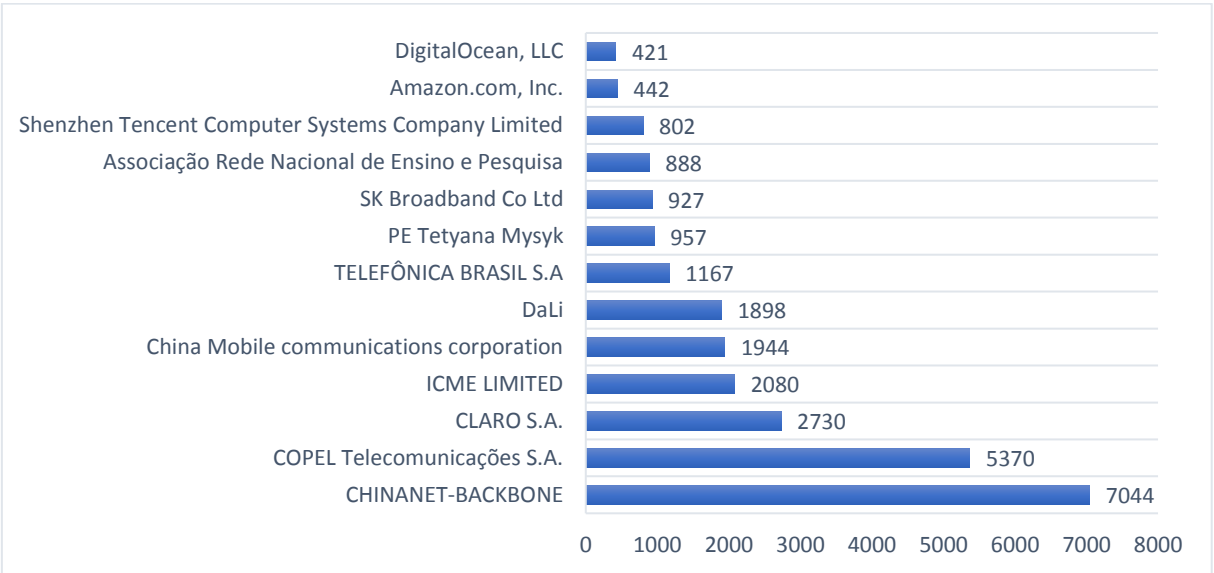
Tabela 13 - IPs da instituição que sofreram tentativas de acesso SSH e o total de conexões classificadas como anomalias.

IP	Conexões	Nº de IP's Externos
192.168.0.24	16503	74
192.168.0.25	10110	24
192.168.0.30	901	12
192.168.0.29	696	6
192.168.0.28	506	6
192.168.0.27	502	8
192.168.0.26	42	5

Fonte: o autor.

Como pode ocorrer de vários IPs serem de uma mesma organização, foram obtidos os totais de conexões por organização, conforme pode ser observado na Figura 23, que exibe as onze primeiras com mais conexões. Sendo um total de quarenta e duas organizações identificadas.

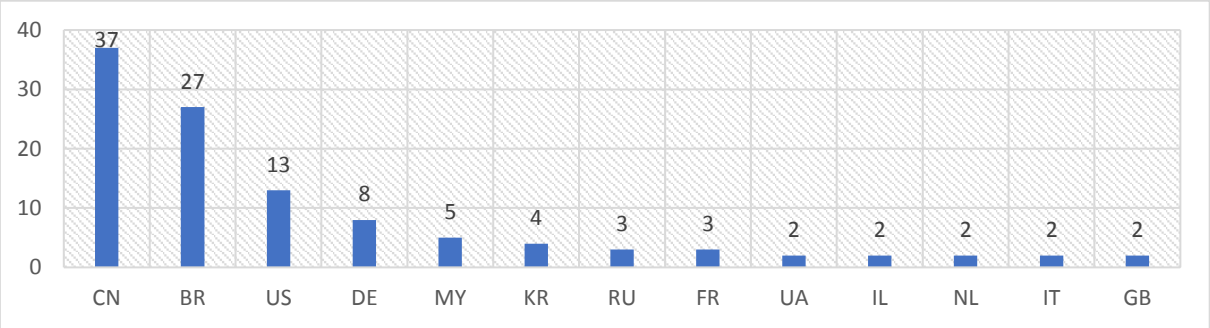
Figura 23 - Número de conexões SSH originadas por organização classificadas como anomalias – Dados IES.



Fonte: o autor.

A partir destes números foram analisados os países dos IPs de onde todas essas conexões anômalas foram geradas, sendo um total de vinte, dos quais os treze países que mais tiveram IPs detectados são exibidos na Figura 24.

Figura 24 - Número de IPs por país que efetuaram conexões SSH a rede da instituição.

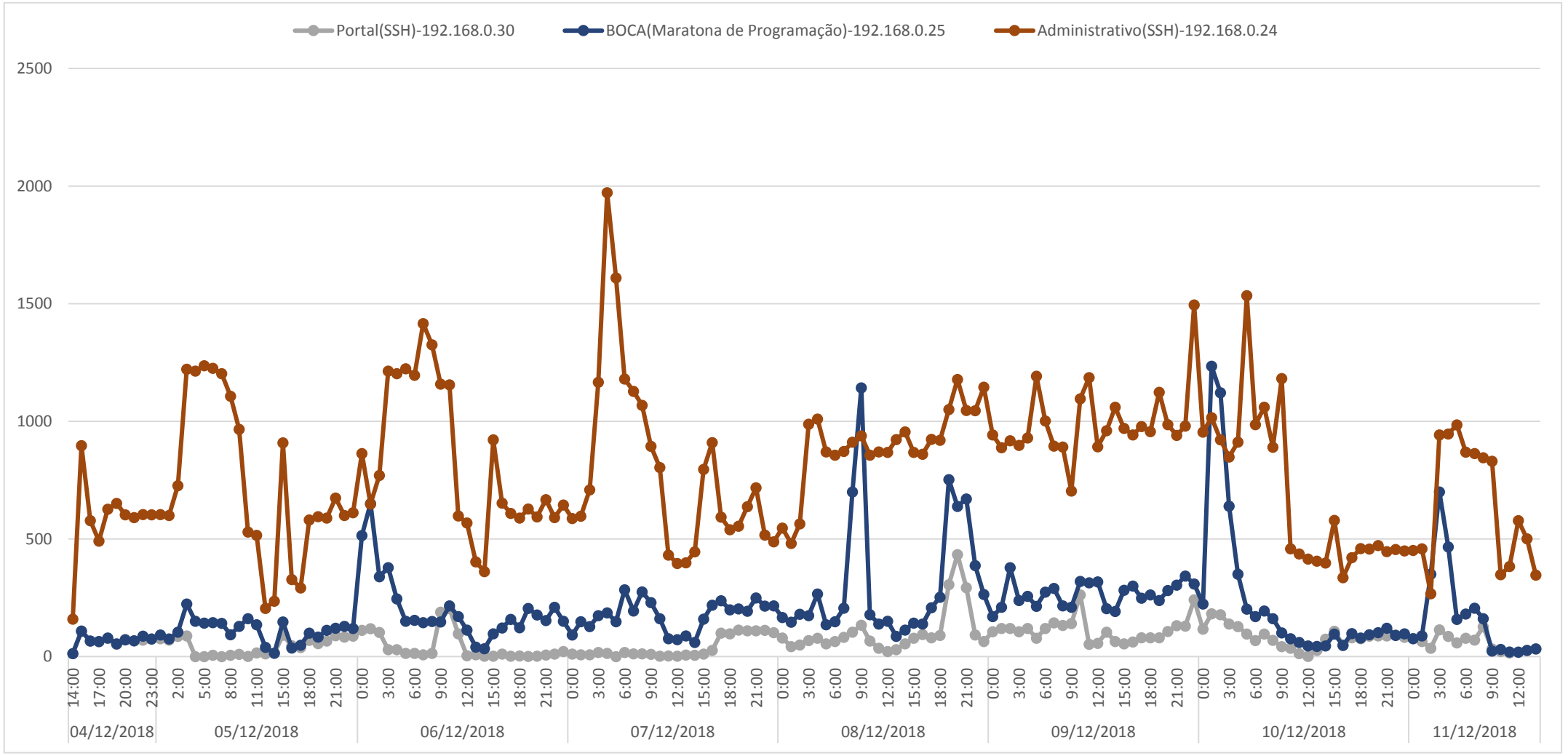


Fonte: o autor.

Foi gerada uma linha do tempo com o fluxo de conexões com os três IPs mais atacados, na qual foram consideradas os dois tipos de conexões, as com e sem anomalias, para identificar os períodos nos quais ocorreram as tentativas de acesso, conforme pode ser observado na Figura 25. Essa linha do tempo foi gerada por dia e hora, logo todas as conexões efetuadas em direção à vítima são obtidas, nesse caso foi considerado o campo RESP_ORIG para agrupamento.

O período que ocorreu o auge de conexões com destino ao IP 192.168.0.24 foi o dia 07/12/2018 às 04:00 horas, onde um total de quatorze IPs efetuou 1.972 conexões, das quais 906 foram classificadas como anomalias praticadas por cinco IPs. Nesse horário teve um intervalo de sete minutos das 04:34-04:41 que a média foi de 68 conexões por minuto praticadas pelo IP 223.111.182.119, logo este foi um dos motivos desta hora ter um número tão elevado de conexões. Outra característica que pode ser observado na Figura 25, é o grande volume de tentativas de acesso de forma continua, uma informação referente a essas tentativas é que apenas uma conexão foi realizada com apenas uma tentativa e teve uma duração 01:13 horas, se consideramos todas as demais, a média de duração por conexão é de cinco segundos.

Figura 25 - Linha do tempo dos IPs internos que mais sofreram tentativas de conexões SSH – Dados IES.



Fonte: o autor.

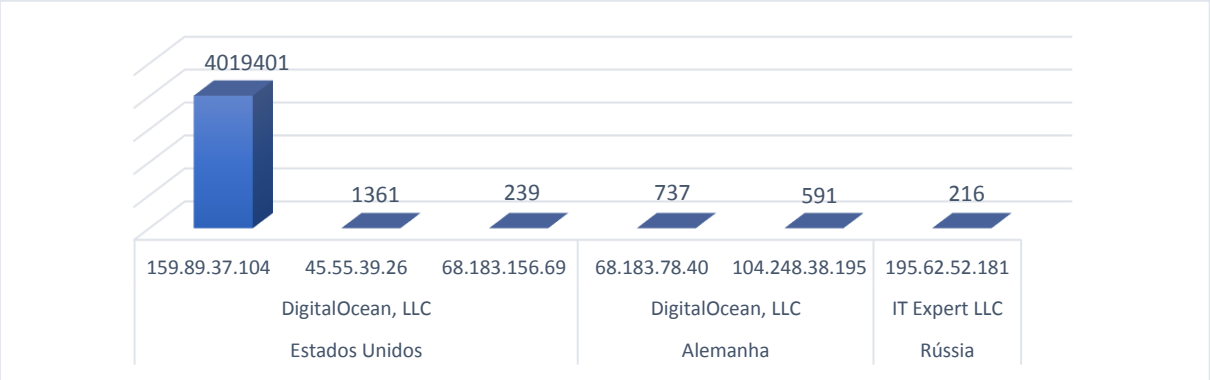
4.4.3 Anomalias de varredura de portas

O volume de dados analisados para identificar esse tipo de anomalia foi de 10.431.040 conexões referentes ao protocolo TCP, sobre as quais foram classificadas 4.022.545 conexões com características de varredura de portas, o que representa 38,56% de todas as conexões que foram efetuadas pelo protocolo TCP durante os onze dias que os dados foram coletados. Um total de seis IPs de origem utilizando seis portas distintas sendo elas “22619, 25515, 32422, 59326, 65500 e 9215” que efetuaram todas essas conexões caracterizadas como varredura de porta em direção a 243 IPs da rede.

As primeiras conexões detectadas foram no dia 06/12/2018 das 05:00 até as 06:00, no dia 07/12/2018 não ocorreu nenhum caso, retornando no dia 08/12/2018 que continuaram a aparecer até o dia 11/12/2018. Ocorrendo no dia 09/12/2018 o maior volume de conexões classificadas, com um total de 1.574.609, em um intervalo que teve início às 10:00 horas e foi até as 23:00 horas apresentando conexões classificadas como varredura de portas, sendo que a média destas conexões por hora foi de 444.

O algoritmo *K-means* nessa análise definiu o valor do primeiro centroide de 1,56 enquanto o segundo tem o valor de 418,57, consequentemente, as conexões atribuídas ao segundo *cluster* foram classificadas como anomalias, recordando que para o processamento dos dados durante a análise de varredura de portas os mesmos foram agrupados por “dd.MM.yyyy HH”. Sobre os IPs identificados foram obtidos seus respectivos países e organizações, conforme pode ser observado na Figura 26, onde apenas um IP de origem concentra 99,92% de todas as conexões classificadas como anômalas.

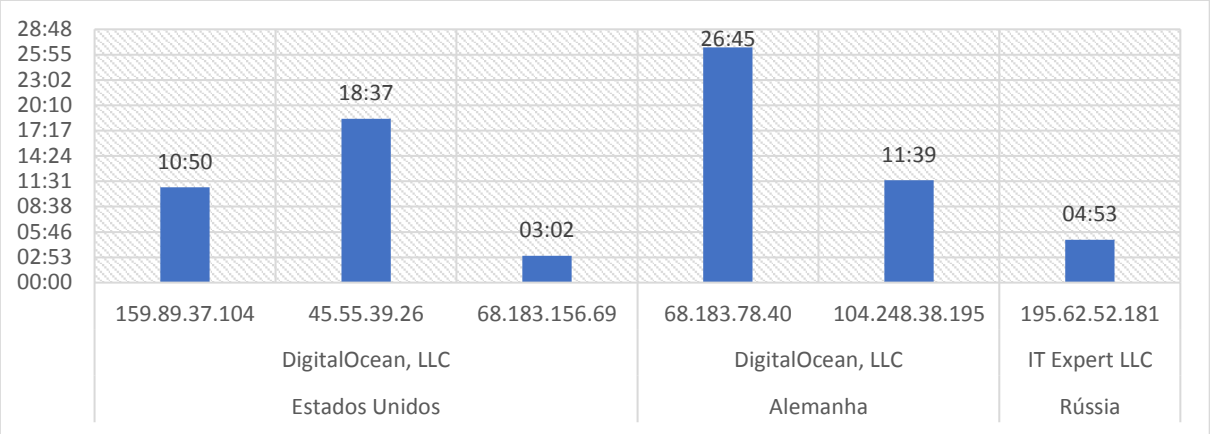
Figura 26 - Totais de conexões TCP classificadas como varredura de porta por IP externo – Dados IES.



Fonte: o autor.

Todas essas conexões tiveram uma duração de apenas uma hora e quinze minutos, ao visualizar essa informação por IP, identificamos que o IP com o maior número de conexões não foi o que teve a maior duração, mesmo que a média de duração por conexão seja muito baixa para todos os IPs. Os tempos de duração por IP e minutos são exibidos na Figura 27, que agrupa por país, IP e organização apenas referente às conexões classificadas como varredura de portas.

Figura 27 - Duração de todas conexões anômalas por minuto originadas por IPs externos – Dados IES.

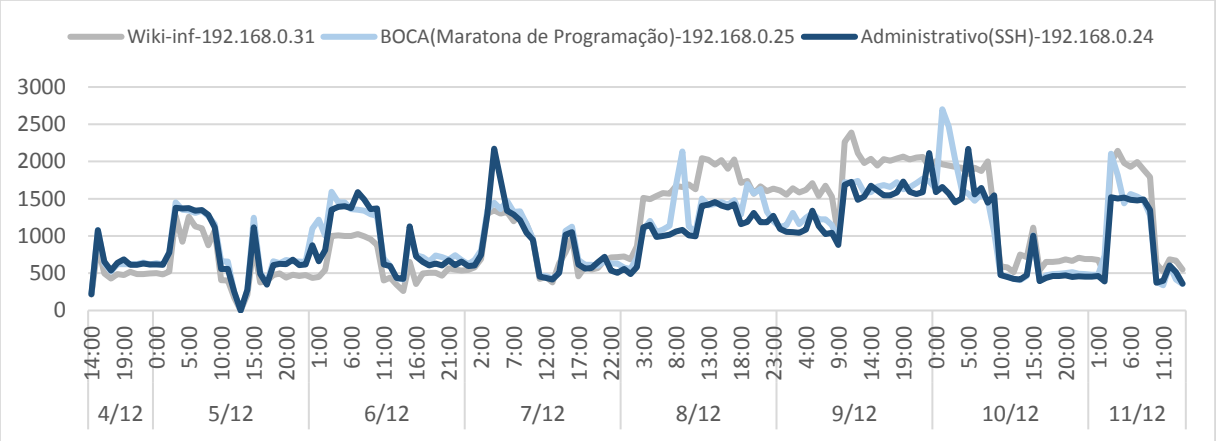


Fonte: o autor.

Dentre os IPs que foram vítimas destas conexões foram selecionados os cinco que tiveram os maiores picos de conexões, independentemente se naquele momento ocorreram ou não anomalias, para exibir a linha de tempo com o número de conexões que cada um destes recebeu, sendo possível assim uma melhor visualização, conforme apresentado na Figura 28.

Conforme pode ser observado os dois maiores picos de conexões foram em períodos que não foram registradas anomalias, de acordo com as informações apresentadas no início desta subseção. Entretanto, é possível verificar um aumento de conexões no dia 08 em de todos IPs em determinado período, o que ocorre novamente iniciando no dia 09 até o dia 10, com cada IP mantendo um padrão do número de conexões muito semelhantes, sendo exatamente neste período que ocorreu o maior volume de conexões anômalas.

Figura 28 - IPs da instituição que foram vítimas de varredura de porta e tiveram o maior pico de conexões TCP.

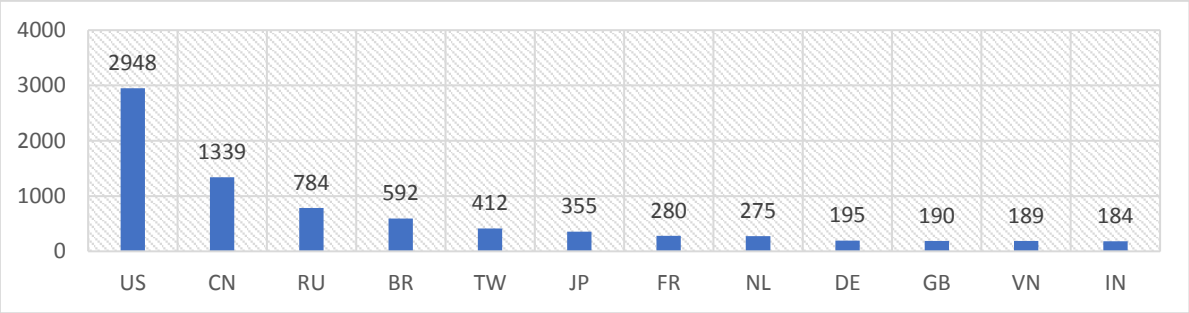


Fonte: o autor.

4.4.4 Análise sobre IPs

Com o objetivo de efetuar a análise dos IPs que mais realizaram acessos à rede da instituição independente do protocolo utilizado, foram selecionados 9.910 IPs que mais realizaram conexões, que juntos totalizam 8.298.264 conexões, e, a partir deles, foram feitas as análises que serão apresentadas. Foram identificados 126 países nos quais esses IPs são originados, dos quais pode ser observado na Figura 29 os doze países que possuem o maior número de IPs.

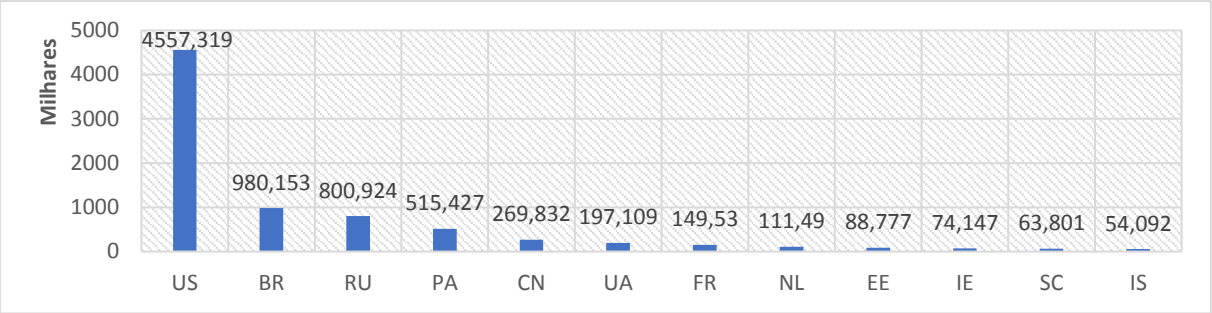
Figura 29 - Número de IPs por país que acessaram a rede da instituição.



Fonte: o autor.

Com o número de IPs por país foi verificado quais possuem o maior número de conexões, apresentando os dez primeiros na Figura 30. A soma do volume de dados enviados e recebidos por estes IPs são de1094 MB e 3080 MB respectivamente.

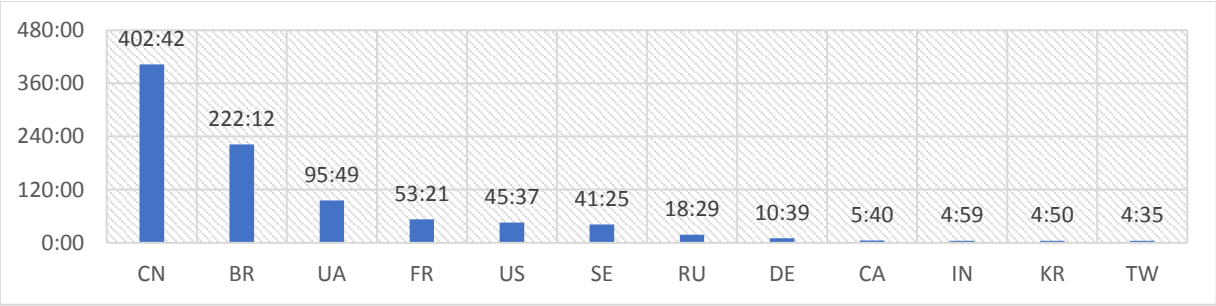
Figura 30 - Número conexões realizadas por país com destino a rede da instituição.



Fonte: o autor.

Como pode ser observado nessas informações acima, os Estados Unidos é o país que possui o maior número de IPs e conexões, e na segunda posição do número de IPs aparece a China, mas não representa nesse caso o segundo com mais conexões aparecendo apenas na quinta posição. Logo não significa que o país com um número maior de IPs vai necessariamente possuir mais conexões, em três países dos doze que foram exibidos ocorreu a permanência na mesma posição, sendo Estados Unidos, França (FR) e Holanda (NL). Por fim, a Figura 31 apresenta por país, a duração em horas das conexões que foram realizadas, exibindo os doze países com o maior tempo de conexão.

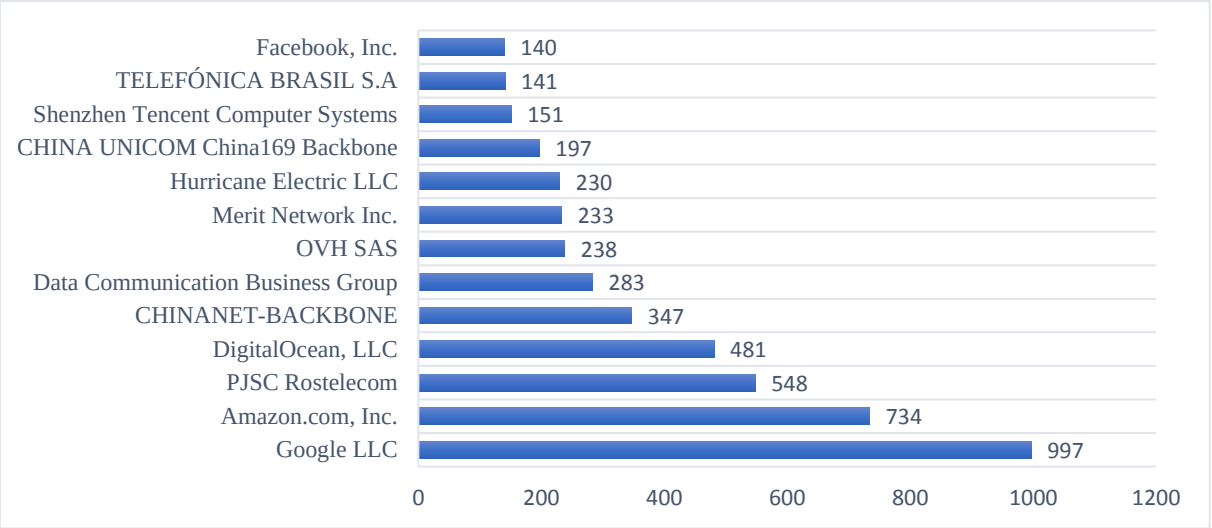
Figura 31 - Duração das conexões realizadas por país que acessaram a instituição.



Fonte: o autor.

Após as informações por país, foram identificadas as organizações, sendo um total de 1.337, destas foram selecionadas as treze que utilizaram o maior número de IPs distintos para se conectar à rede. Essa informação apresentada na Figura 32.

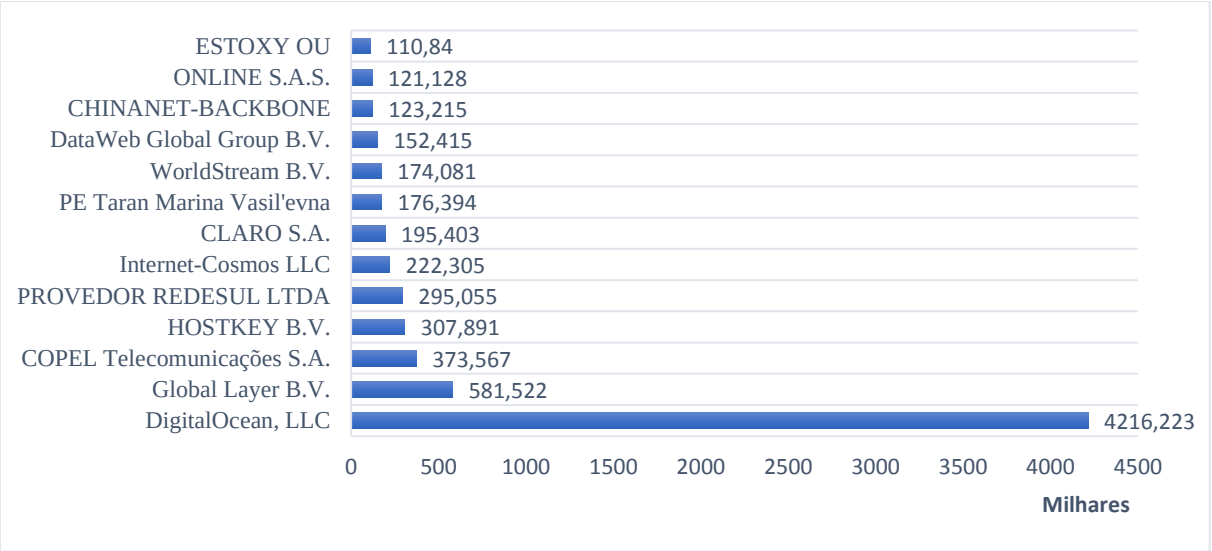
Figura 32 - Número de IPs utilizados por organização com destino a rede da instituição.



Fonte: o autor.

A partir das organizações foram identificadas as treze que mais efetuaram conexões, conforme pode ser observado na Figura 33, que apresenta a DigitalOcean como responsável por mais de quatro milhões de conexões. Se somar as conexões de todas as outras 1.336 organizações totalizam 4.082.041, não chegando ao número de conexões da DigitalOcean.

Figura 33 - Número de conexões realizadas por organização com destino a rede da instituição.



Fonte: o autor.

Além destas informações referentes aos IPs que mais acessaram a rede, também foi efetuada a análise de quais os IPs que mais foram acessados a partir da instituição, independente do protocolo utilizado, logo foram analisados 3.491 IPs, referentes a cinquenta e um países

diferentes que somam um total de 2.529.512 conexões. Os países com o maior número de IPs são exibidos na Tabela 14, enquanto os países com o maior número de conexões na Tabela 15, as duas ordenadas em ordem decrescente.

Tabela 14 - Países com maior Nº de IPs externos acessados.

País	IPs
Estados Unidos	2583
Brasil	316
Alemanha	87
Irlanda	82
China	47
Canadá	45
Holanda	41
Reino Unido	40
França	27
Singapura	25

Fonte: o autor.

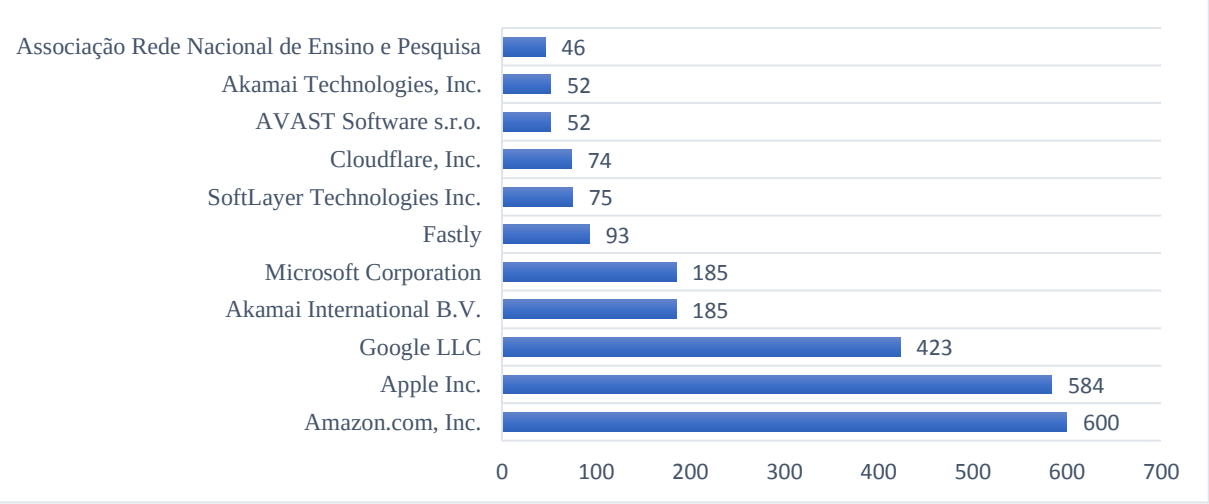
Tabela 15 - Países com maior Nº de conexões acessados pela instituição.

País	Conexões
Estados Unidos	1182579
Holanda	910984
Brasil	224752
Irlanda	39417
Reino Unido	24987
Alemanha	20613
Canadá	15052
Dinamarca	12320
África do Sul	12235
Seicheles	11678

Fonte: o autor.

Ao realizar a análise por organização foram identificadas 299, das quais onze são exibidas na Figura 34, as mesmas agrupam 2.369 IPs que representa 67,86% do total, onde Amazon, Apple e Google fazem parte do grupo que possui mais de 400 IPs, enquanto as demais organizações apresentam menos da metade deste valor.

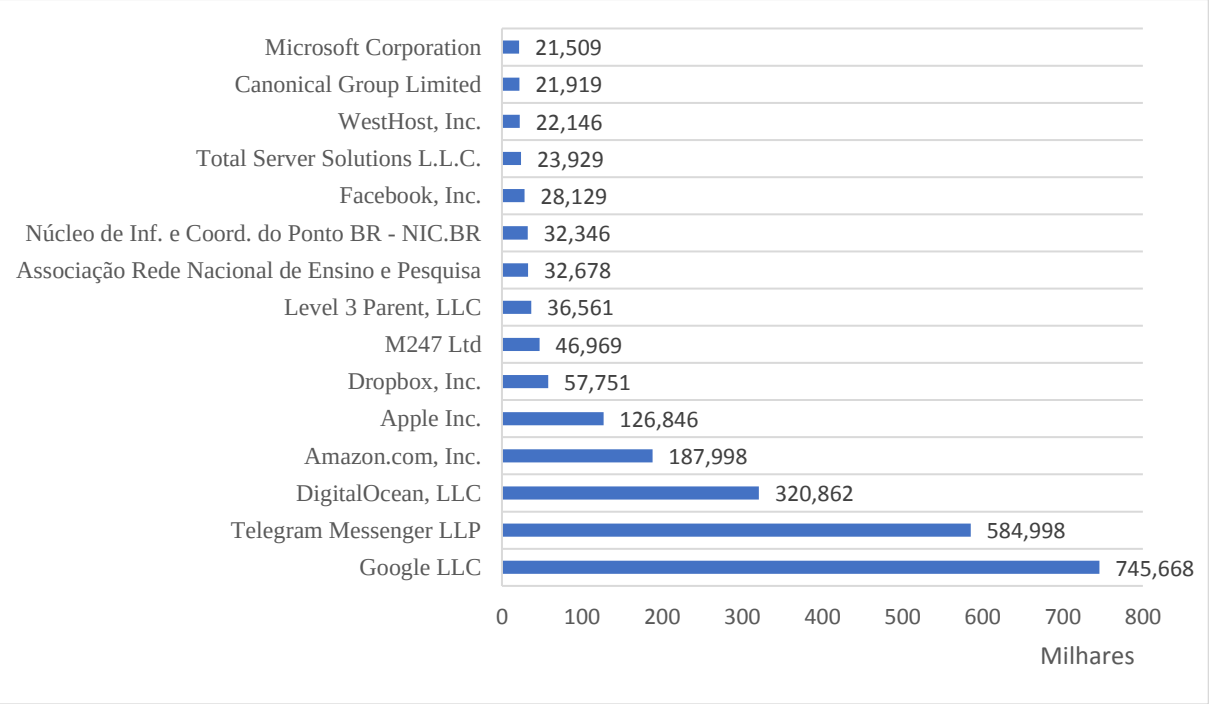
Figura 34 - IPs por organização que foram acessados pela instituição.



Fonte: o autor.

O número de conexões realizadas em direção a dezoito organizações totalizam 2.290.309 que representa 90,54% de todas as conexões efetuadas pelos IPs que estão sendo analisados, essas organizações podem ser visualizadas na Figura 35. Se considerarmos todas as conexões obtemos um total de 46,62 GB de dados que foram recebidos das mesmas e foram enviados 6,64 GB de dados.

Figura 35 - Organizações mais acessadas pela instituição.



Fonte: o autor.

4.4.5 Tempo de processamento

As execuções realizadas do aplicativo *"processa"* para obter estes resultados, são apresentadas na Tabela 16, com os tempos de cada execução. Sendo que o tempo total de processamento dos dados foi de 2h55min.

Tabela 16 - Tempos de processamento dos dados em "mm:ss" – Dados IES.

Descrição	Tempo
Totais UFSM LT	16:43
Totais UFSM GT	20:44
Gerar modelo DDoS/SSH	15:37
Processa K-Means DDoS/SSH LT	15:23
Processa K-Means DDoS/SSH GE	15:10

Gerar modelo TCP	35:22
Processa K-Means Varredura de Portas LT	29:28
Processa K-Means Varredura de Portas GE	26:51

Fonte: o autor.

4.5 DISCUSSÃO

Conforme pode ser visto no decorrer das subseções referente à análise experimental do sistema, a detecção de ataques DDoS aplicada aos dados da UNB obtiveram os resultados esperados, com uma diferença de dois minutos em relação ao período que ocorreu o ataque e do que foi detectado. Outro resultado apresentado foram os ataques DoS referentes ao dia 05/07/2017, detectados devido ao grande volume de conexões realizadas no período, pois os demais ataques presentes nesse dia que são informados na Tabela 9, não foram classificados como ataques, isso ocorre porque os demais geram anomalias diferentes das que foram analisadas pelo sistema. A análise DDoS considera o número de conexões que foram realizadas por um determinado IP de origem utilizando *N* portas de origem em direção ao um mesmo IP de resposta e porta, logo os resultados obtidos são referentes aos ataques que geram um grande volume de conexões.

Os resultados obtidos da análise de varredura de portas sobre o conjunto de dados de teste, teve três períodos identificados dos treze possíveis, sendo detectados os que geraram um volume maior de conexão considerando o mesmo IP de origem e porta para se conectarem em um mesmo IP destino com *N* portas de respostas. Característica que não foi apresentada pelo segundo período com maior volume de conexões, logo o mesmo não teve nenhuma conexão classificada como anomalia nesse intervalo das 14:54 até as 14:56 conforme detalhado nos resultados. Nesse caso específico ocorreram 80.770 conexões distribuídas nesses três minutos, todas efetuadas pelo mesmo IP de origem mas com diversas portas de origem, conforme pode ser observado na Figura 17.

Com os resultados obtidos durante a análise experimental do sistema, foi possível constatar que a forma como é aplicado o processamento utilizando o algoritmo *K-means* não irá detectar todos os tipos de varredura de portas aplicados nesse conjunto de dados, pois seria necessário analisar os dados em si e não apenas o fluxo de conexões. Outro objetivo que foi alcançado durante essas execuções foi a validação da integração realizada entre todas as ferramentas, principalmente o armazenamento de forma correta dos dados, verificando se o número de conexões que se encontravam no arquivo de *log* gerado pelo ZEEK, era o mesmo

número que foi armazenado. Durante essa análise experimental foi identificado a necessidade de inserir o campo *"row_id"* como chave na tabela HBase.

Após a etapa de análise experimental e resultados obtidos sobre o conjunto de dados da UNB, foram aplicados os dados coletados da UFSM para processamento no sistema. Ao gerar a linha de tempo com as conexões realizadas por protocolo TCP e UDP foi possível identificar a ocorrência de uma anomalia em determinado período na rede, que foi comprovada nos resultados exibidos dentro de cada análise efetuada. Na análise de ataques DDoS foram classificadas 179.549 conexões anômalas realizadas pela instituição com destino a IPs externos. Também foram identificados 57 IPs externos que realizaram 1.971 conexões classificadas como anômalas com destino à rede da instituição.

Os resultados obtidos sobre o protocolo SSH demonstraram um volume significativo de tentativas de conexões originadas por IPs externos com destino a vinte IPs da rede, além das quase trinta mil conexões classificadas como anomalia pelo alto número de tentativas de conexões em um curto período de tempo. Também é possível observar na linha de tempo referente aos dois IPs que mais sofreram ataques apresentada nos resultados, que o número de conexões recebidas é alto e contínuo. Cada conexão estabelecida pode efetuar várias tentativas de acesso com login e senha, dependendo da configuração aplicada na máquina ao utilizar o serviço SSH. Esses ataques de força bruta SSH são direcionados a IPs que possuem a porta 22 aberta, o que deve ser evitado.

Com os resultados sobre a varredura de portas obtidos, foi possível compreender o responsável pelo grande volume de conexões TCP, conforme visto na linha do tempo apresentado na Figura 18. Nessa análise foram classificadas mais de quatro milhões de conexões como anomalias, sendo que apenas um IP concentra quase todo esse volume, com uma média de conexões por hora acima do normal, de modo que os centroides definidos nessa análise pelo *K-means* foram os que apresentaram a maior diferença de valores, sendo 1,56 e 418,57.

Os resultados referentes à análise dos IPs obtiveram apenas três países e duas organizações identificadas, já que são referentes apenas a seis IPs, sendo estas organizações DigitalOcean e IT Expert responsáveis por prestação de serviços em nuvem. Ao apresentar a linha do tempo com o número de conexões que foram recebidas pelos IPs da rede interna, observa-se que o período com maior volume de conexões recebidas de forma contínua foi nos mesmos dias que o protocolo TCP apresentou o maior volume de conexões.

Um dos casos que pode ser destacado dos IPs vítimas é o 192.168.0.26 atribuído a máquina que é utilizada na implementação do sistema, a mesma não é utilizada para mais nenhum outro serviço dentro da rede da instituição, que poderia vir a gerar conexões tendo como destino essas organizações ou algum motivo para recebê-las. Mesmo desta forma, do período de 08/12/2018 até 11/12/2018 a mesma recebeu 15.960 conexões do IP 159.89.37.104 e porta 65500, com uma média de 420 conexões por hora nos períodos classificados como anomalia.

As análises efetuadas sobre os IPs mais acessados pela instituição e os que mais efetuaram acessos à rede interna, obtiveram inúmeros países de origem e de destino, bem como diversas organizações. Pelos resultados apresentados, é possível constatar que grande parte dos IPs que acessam a rede interna são originados de provedores de internet ou prestadores de serviço em nuvem. Enquanto os IPs que foram mais acessados a partir da instituição são de organizações que prestam serviços online ou referentes a mídias sociais, foram as conexões realizadas pela instituição que geraram o maior volume de dados enviados e recebidos com um total de 6,64 GB e 46,62 GB respectivamente.

4.6 SUMÁRIO

Nesse capítulo apresentou-se os recursos de *hardware* e máquinas virtuais utilizadas para implementação do sistema. Como foi possível observar dois conjuntos de dados serviram como base para o processamento, utilizou-se os dados com o tráfego de rede controlado para a validação de todas as etapas realizadas pelo sistema, desde a coleta até a análise dos dados que obteve os resultados esperados, identificando as anomalias causadas pelos ataques DDoS e varredura de porta. O segundo conjunto de dados foi criado a partir do monitoramento da rede da instituição durante oito dias, o qual tem mais de quatorze milhões de conexões coletadas, sobre esses dados foram aplicadas as análises de ataques DDoS considerando o protocolo HTTP, força bruta SSH e varredura de portas sobre o protocolo TCP. Sobre os resultados foram obtidas informações do número de IPs responsáveis pelos ataques, assim como os IPs que foram as vítimas, totais de conexões classificadas com anomalias, linha do tempo com o número de conexões por IP e totais por organização e país dos IPs atacantes.

Ao visualizar os resultados obtidos sobre os dados da instituição, observou-se que as anomalias identificadas da análise de ataques DDoS, não foram praticados contra a rede interna, mas sim por ela. No entanto, o resultado de força bruta trouxe um número significativo de IPs externos que efetuaram conexões em direção à instituição, assim como apresentou um volume

contínuo de conexões realizadas em direção a determinados IPs internos. Observou-se que os resultados obtidos da análise de varredura de portas foi o que classificou o menor número de IPs externos como anômalos, entretanto foram os que mais realizaram conexões com anomalias, nesse caso, varredura de portas.

Exibiu-se ainda informações relacionadas aos IPs mais acessados pela instituição, com os totais por organização e país, e o total de dados enviados e recebidos. As mesmas informações também foram obtidas dos quase dez mil IPs externos analisados, que mais acessaram a instituição.

5 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho teve como objetivo realizar o desenvolvimento de um sistema com base na arquitetura Lambda para efetuar o processamento do fluxo de dados da rede, implementando, assim, as camadas de *speed, batch* e *service*. Com a finalidade de executar análises sobre os dados coletados, para identificar anomalias que são causadas por ataques DDoS por meio do protocolo HTTP, ataques de força bruta pelo serviço SSH, varreduras de portas realizadas mediante o protocolo de transporte TCP. A partir dos resultados destas análises, obter informações dos IPs que causaram essas anomalias, sendo esse detalhe um dos diferenciais do sistema.

Além destas análises aplicadas à detecção de anomalias, também são obtidas informações referentes ao tráfego de rede, como os IPs mais acessados por meio da rede interna e os que mais efetuaram acessos à rede local, número de conexões por protocolo e volume de dados enviados e recebidos. Dentre esses aspectos o sistema implementado visa atender a necessidade de monitoramento do fluxo de dados do tráfego de rede, trabalhando com os dados conforme os mesmos são gerados a cada conexão realizada na rede, de uma forma com baixa latência, para efetuar sua normalização e armazenamento, para que, posteriormente, possam ser efetuadas análises sobre as informações coletadas.

No decorrer deste estudo detalhou-se a arquitetura utilizada no sistema, as ferramentas necessárias para sua implementação e a integração realizada entre as mesmas, bem como a estrutura lógica de funcionamento do sistema. Conforme a especificação do sistema evidenciada no decorrer do trabalho, foi possível notar que a camada de velocidade da arquitetura Lambda não foi implementada de forma completa, sendo utilizada de forma parcial para receber o fluxo de dados, normalizá-los e efetuar o armazenamento dos mesmos na tabela do HBase, sem efetuar as análises que foram implementadas na cama em lote.

Após os primeiros testes realizados sobre o conjunto de dados controlados, foi possível validar o funcionamento da estrutura lógica do sistema, bem como a integração entre as ferramentas escolhidas. Ao analisar os resultados obtidos constatou-se que as anomalias apresentadas por ataques DDoS foram identificadas, assim como em determinado caso foi possível classificar um tipo de ataque DoS. Entretanto, sobre as conexões que efetuaram varredura de porta não ocorreu a detecção de todos os tipos de varreduras do conjunto, isso ocorre devido ao tipo de análise implementada, com foco sobre o volume de conexões realizadas e não sobre as características individuais de cada uma.

Posteriormente a análise experimental executada sobre o conjunto de dados controlados, o sistema foi aplicado sobre um conjunto de dados reais, coletados de uma rede local da IES. Com volume significativo de conexões, o sistema foi capaz de classificar conexões anômalas dentre os tipos considerados no trabalho, DDoS, força bruta e varredura de portas. Outros trabalhos que implementaram a arquitetura Lambda, propuseram diferentes tipos de análises sobre as características individuais de cada conexão, enquanto esse estudo atua sobre o volume de conexões para identificar as anomalias. Apesar deste estudo caminhar no mesmo sentido, difere-se dos demais pois sobre os resultados classificados como anômalos, são realizadas consultas sobre os IPs para se obter informações dos responsáveis, classificando-os de acordo com país e organização.

Todos os resultados obtidos das análises, assim como as informações do tráfego de rede são armazenadas em tabelas, logo esses resultados podem ser utilizados posteriormente sem a necessidade de reprocessamento. De acordo com os objetivos definidos neste trabalho, pode-se afirmar que foram alcançados, pois durante as análises experimentais sobre o conjunto de dados controlados e reais, o sistema foi capaz de trabalhar com o fluxo de dados no momento da leitura dos mesmos, para armazená-los e posteriormente efetuar sua análise.

Como trabalhos futuros sugere-se a implementação de forma completa da camada de velocidade, que já está trabalhando com o fluxo de dados, mas não efetua a análise dos dados. A aplicação do sistema para análise em tempo real da rede monitorada, bem como a aplicação de outros tipos de analises, que verifiquem as informações de cada conexão de forma individual com objetivo de identificar os demais tipos de varreduras de portas e detectar também ataques DoS, obtendo assim uma variedade maior de anomalias a serem identificadas.

Destaca-se como principal sugestão de continuidade deste sistema a integração das ferramentas do Elasticsearch[12] com o Spark, por possuírem compatibilidade e tornarem possível a visualização dos resultados das camadas de velocidade e lote em *dashboards*. Além da continuação deste sistema adicionando os novos recursos descritos acima, sugere-se a implementação de um novo sistema utilizando como base a arquitetura Kappa, que é direcionada ao processamento de fluxos em tempo real sem a presença da camada em lote.

[12] Elasticsearch: https://www.elastic.co/

REFERÊNCIAS

ANGELOV, P. Outside the Box: An Alternative Data Analytics Frame-work. **Journal of Automation Mobile Robotics and Intelligent Systems.** [S.l.], v. 8, n. 2, p. 29-35, 2014. DOI: 10.14313/JAMRIS2-2014/16. Disponível em: <http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.baztech-1e4a2ccb-270b-45e1-b7f7-0d45deb4fba7/c/Angelov_outside_the_box_2_14.pdf.> Acesso em: 17 nov. 2018.

BERMAN, J. J. **Principles of Big Data**: Preparing, Sharing, and Analyzing Complex Information. Boston: Morgan Kaufmann, 2013. 288 p.

BERTRAN, P. F. Lambda Architecture: A state-of-the-art. **Datasalt.** [S.l.], jan. 2014. Disponível em: <http://www.datasalt.com/2014/01/lambda-architecture-a-state-of-the-art/> Acesso em: 12 out. 2018.

BHOJWANI, N.; SHAH, V. A survey on hadoop hbase system. **International Journal of Advance Engineering and Research Development.** [S.l.], v. 3, n. 1, p. 82-87, 2016. Disponível em: <http://www.ijaerd.co.in/papers/finished_papers/A%20SURVEY%20ON%20HADOOP%20HBASE%20SYSTEM-13335.pdf>. Acesso em: 15 dez. 2018.

CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Anomaly detection: A survey. **ACM computing surveys (CSUR)**. [S.l.], v. 4, p. 15:1-15:58, 2009. Doi: 10.1145/1541880.1541882. Disponível em: <https://www.vs.inf.ethz.ch/edu/HS2011/CPS/papers/chandola09_anomaly-detection-survey.pdf>. Acesso em: 30 out. 2018.

CHARLES, P. J.; BHARATHI, S.T.; SUSMITHA, V. Big Data – Concepts, Analytics, Architectures – Overview. **International Research Journal of Engineering and Technology (IRJET).** [S.l.], v. 5, n. 2, p. 125-129, fev. 2018. Disponível em: <https://www.irjet.net/archives/V5/i2/IRJET-V5I231.pdf>. Acesso em: 16 nov. 2018.

CHEN, M.; MAO, S.; LIU, Y. Big data: A Survey. **Mobile Networks and Applications**. [S.l.], v. 19, n. 2, p. 171-209, abr. 2014. Doi: 10.1007/s11036-013-0489-0. Disponível em: < https://link.springer.com/article/10.1007/s11036-013-0489-0>. Acesso em: 07 ago. 2018.

CHEN, Z.; ZHANG, H.; HATCHER, W.G.; NGUYEN, J.; YU, W. A Streaming-Based Network Monitoring and Threat Detection System. In: *14th International Conference on Software Engineering Research, Management and Applications (SERA), 2016,* Towson, MD. **Anais**… IEEE, jun. 2016. p. 31-37. Doi: 10.1109/SERA.2016.7516125

DANESH, S. **Big data - From Hype to Reality**. 2014. Dissertation (Master's Degree in Statistics)-Örebro University School of Business, Örebro, 2014. Disponível em: < http://oru.diva-portal.org/smash/record.jsf?pid=diva2%3A752309&dswid=-6780>. Acesso em: 03 set. 2018.

DAS, S.; BEHERA, R.K.; KUMAR, M.; RATH, S.K. Real-Time Sentiment Analysis of Twitter Streaming data for Stock Prediction. **Procedia Computer Science**, [S.l.], v. 132, p. 956-964, 2018. Disponível em: <https://doi.org/10.1016/j.procs.2018.05.111>. Acesso em: 18 set. 2018.

DEMERTZIS, K.; TZIRITAS, N.; KIKIRAS, P.; SANCHEZ, S. L.; ILIADIS, L. The Next Generation Cognitive Security Operations Center: Adaptive Analytic Lambda Architecture for Efficient Defense against Adversarial Attacks**.** ***Big Data Cogn. Comput****.* [S.l.], 2019. Disponível em: <https://doi.org/10.3390/bdcc3010006>. Acesso em: 16 fev. 2019.

DIVAKARAN, D.M.; FOK, K.W.; NEVAT, I.; THING, V.L.L. Evidence gathering for network security and forensics. **Digital Investigation**, [S.l.], v. 20, p. 56-55, mar. 2017. Disponível em: <https://doi.org/10.1016/j.diin.2017.02.001>. Acesso em: 08 jul. 2018.

DUMBILL, E. **Planning for big data.** Sebastopol, CA: O'Reilly Media, 2012.84 p.

GANTZ, J.; REINSEL, D. Extracting value from chaos. **IDC iView**. [S.l.], jun. 2011. Disponível em: <https://www.emc.com/collateral/analyst-reports/idc-extracting-value-from-chaos-ar.pdf>. Acesso em: 11 out. 2018.

GUGNANI, S.; LU, X.; QI, H.; ZHA, L.; PANDA, D. K. D. K. Characterizing and accelerating indexing techniques on distributed ordered tables. In: *International Conference on Big Data (Big Data),* 2017, Boston, MA. **Anais**… IEEE, dez. 2017, p. 173-182. Doi: 10.1109/BigData.2017.8257925.

GÜRCAN, F.; BERIGEL, M. Real-Time Processing of Big Data Streams: Lifecycle, Tools, Tasks, and Challenges. In: *2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT),* 2018, Ankara, Turkey. **Anais**… IEEE, out. 2018, p. 1-6. Doi: 10.1109/ISMSIT.2018.8567061.

GURUSAMY, V.; KANNAN, S.; NANDHINI, K. The Real Time Big Data Processing Framework: Advantages and Limitations. **International Journal of Computer Science and Engineering (JCSE)**, [S.l.], v. 5, n. 12, p. 305-312, dez. 2017. Doi: 10.26438/ijcse/v5i12.305312. Disponível em: https://www.researchgate.net/publication/322550872_The_Real_Time_Big_Data_Processing_Framework_Advantages_and_Limitations. Acesso em: 08 out. 2018.

HAN, D.; BI, K.; LIU, H.; JIA, J. A DDoS Attack Detection System Based on Spark Framework. **Computer Science and Information Systems**, [S.l.], v. 14, n. 3, p. 769-788, 2017. Disponível em: < https://doi.org/10.2298/CSIS161217028H>. Acesso em: 14 ago. 2018.

HASANIFARD, M; LANDANI, B. T. DoS and port scan attack detection in high speed networks. In: *11th International ISC Conference on Information Security and Cryptology*, 2014, Tehran, Iran. **Anais**… IEEE, set. 2014, p.61-66. Doi: 10.1109/ISCISC.2014.6994023

HASHEM, I. A. T.; YAQOOB, I.; ANUAR, N. B.; MOKHTAR,S.; GANI, A.; KHAN, S. U. The rise of "big data" on cloud computing: Review and open research issues. **Information Systems**. [S.l.], v. 47, p. 98-115, jan. 2015. Disponível em: <https://doi.org/10.1016/j.is.2014.07.006.>. Acesso em: 21 out. 2018.

HOQUE, N.; BHATTACHARYYA, D. K.; KALITA, J. K. A novel measure for low-rate and high-rate DDoS attack detection using multivariate data analysis. In: *8th International*

Conference on Communication Systems and Networks (COMSNETS), 2016, Bangalore. **Anais**… IEEE, jan. 2016, p. 1-2. Doi: 10.1109/COMSNETS.2016.7439939.

HUN, S. H.; NASRIDINOV, A.; RYU, K. H. Information Flow Monitoring System. **IEEE Access**, [S.l.], v. 6, p. 23820-23827, abr. 2018. Doi: 10.1109/ACCESS.2018.2829495.

HUSÁK, M.; ČERMÁK, M.; JIRSÍK, T.; ČELEDA, P. HTTPS traffic analysis and client identification using passive SSL/TLS fingerprinting. **EURASIP Journal on Infomation Security**, [S.l.], p. 1-14, 2016. Disponível em: <https://doi.org/10.1186/s13635-016-0030-7>. Acesso em: 17 ago. 2018.

JAHN, G. F. **Uma proposta de arquitetura para tratamento de dados não estruturados no âmbito dos institutos federais de educação.** 2017. p. 130. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Pernambuco, Recife, PE, 2017. Disponível em: < https://repositorio.ufpe.br/handle/123456789/28002>. Acesso em: 04 set. 18.

KARIMI, A.M.; NIYAZ, Q.; SUN, W.; JAVAID, A.Y.; DEVABHAKTUNI, V.K. Distributed Network Traffic Feature Extraction for a Real-time IDS. In: *International Conference on Electro Information Technology*, 2016, Grand Forks, ND. **Anais**… IEEE, mai. 2016, p. 522-526. Doi: 10.1109/EIT.2016.7535295.

KATKAR, J. **Study of big data architecture lambda architecture**. 2015. p. 458. Dissertation (Degree Master of Science) - San Jose State University, San Jose, CA, 2015. Disponível em: < https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1458&context=etd_projects>. Acesso em: 04 jul. 2018.

KAUSHIK, A. K.; PILLI, E. S.; JOSHI, R. C. Network forensic system for port scanning attack. In: *2nd International Advance Computing Conference (IACC),* 2010, Patiala, India. **Anais**... IEEE, fev. 2010, p. 310-315. Doi: 10.1109/IADCC.2010.5422935.

KHAN, M. A.; UDDIN, M. F.; GUPTA, N. Seven V's of Big Data understanding Big Data to extract value. In: *Proceedings of the 2014 Zone 1 Conference of the American Society for Engineering Education*, 2014, Bridgeport, CT, 2014. **Anais**… IEEE, abr. 2014, p. 1-5. Doi: 10.1109/ASEEZone1.2014.6820689.

KIEYZUN, A.; GUO, P. J.; JAYARAMAN, K.; ERNST, M. D. Automatic creation of SQL Injection and cross-site scripting attacks. In: *31st International Conference on Software Engineering,* 2009, Vancouver, BC. **Anais**… IEEE, mai. 2009, p. 199-209. Doi: 10.1109/ICSE.2009.5070521.

KIRAN, M.; MURPHY, P.; MONGA, I.; DUGAN, J.; SINGH, S. B. Lambda Architecture for Cost-effective Batch and Speed Big Data processing. In: *International Conference on Big Data (Big Data)*, 2015, Santa Clara, CA. **Anais**… IEEE, out. 2015, p. 2785-2792. Doi: 10.1109/BigData.2015.7364082.

KOZIK, R. Distributed System for Botnet Traffic Analysis and Anomaly Detection. In: *International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom)*

and IEEE Smart Data (SmartData), 2017, Exeter, UK. **Anais**… IEEE, jun. 2017, p. 330-335. Doi: 10.1109/iThings-GreenCom-CPSCom-SmartData.2017.55.

KREPS, J. Questioning the Lambda Architecture. **O'Reilly**, [S.l.], jul. 2014. Disponível em: < https://www.oreilly.com/ideas/questioning-the-lambda-architecture>. Acesso em: 05 ago. 2018.

LOPEZ, M. A.; LOBATO, A. G. P.; DUARTE, O. C. M. B.; PUJOLLE G. An Evaluation of a Virtual Network Function for Real-time Threat Detection Using Stream Processing. In: *Fourth International Conference on Mobile and Secure Services (MobiSecServ),* 2018, Miami Beach, FL. **Anais**... IEEE, fev. 2018, p. 1-5. Doi: 10.1109/MOBISECSERV.2018.8311440.

LIU, D.; ZHANG M.; LI T. Network Traffic Analysis Using Refined Bayesian Reasoning to Detect Flooding and Port Scan Attacks. In: *International Conference on Advanced Computer Theory and Engineering,* 2008, Phuket, Thailand. **Anais**… IEEE, dez. 2008, p. 1000-1004. Doi: 10.1109/ICACTE.2008.44.

LI, Y.; MANOHARAN, S. A performance comparison of SQL and NoSQL databases. In: *Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM),* 2013, Victoria, BC. **Anais**… IEEE, ago. 2013, p. 15-19. Doi: 10.1109/PACRIM.2013.6625441.

MARCHAL, S.; JIANG, X.; STATE, R.; ENGEL,T. A Big Data Architecture for Large Scale Security Monitoring. In: *International Congress on Big Data,* 2014, Anchorage, AK. **Anais**… IEEE, jul. 2014, p. 56-63. Doi: 10.1109/BigData.Congress.2014.18.

MARTINS, R. S.; ANGELOV, P.; COSTA, B. S. J. Automatic Detection of Computer Network Traffic Anomalies based on Eccentricity Analysis. In: *International Conference on Fuzzy Systems (FUZZ-IEEE),* 2018, Rio de Janeiro. **Anais**... IEEE, jul. 2018, p. 1-8. Doi: 10.1109/FUZZ-IEEE.2018.8491507.

MARUHASHI, K.; GUO, F.; FALOUTSOS, C. MultiAspectForensics: Pattern Mining on Large-Scale Heterogeneous Networks with Tensor Analysis. In: *International Conference on Advances in Social Networks Analysis and Mining*, 2011, Kaohsiung. **Anais**… IEEE, jul. 2011, p. 203-210. Doi: 10.1109/ASONAM.2011.80.

MARZ, N. How to beat the CAP theorem. In: *Thoughts from the red planet*. [S.l.], out. 2011. Disponível em: <http://nathanmarz.com/blog/how-to-beat-the-cap-theorem.html>. Acesso em: 05 ago. 2018.

MARZ, N.; WARREN, J. **Big Data:** Principles and best practices of scalable real time data systems. [S.l.]: Manning Publications, 2015. 328 p.

MAZEL, J.; FONTUGNE, R.; FUKUDA, K. A taxonomy of anomalies in backbone network traffic. In*: International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2014, Nicosia. **Anais**… IEEE, ago. 2014, p. 30-36. Doi: 10.1109/IWCMC.2014.6906328.

MUKADDAM, A.; ELHAJJ, I.; KAYSSI, A.; CHEHAB, A. IP Spoofing Detection Using Modified Hop Count. In: *28th International Conference on Advanced Information Networking*

and Applications, 2014, Victoria, BC. **Anais**… IEEE, mai. 2014, p. 512-516. Doi: 10.1109/AINA.2014.62.

NUPAIROJ, N.; TANGSATJATHAM, P. Hybrid big data architecture for high-speed log anomaly detection. In: *13th International Joint Conference on Computer Science and Software Engineering (JCSSE),* 2016, Khon Kaen, Thailand. **Anais**… IEEE, jul. 2016, p. 1-6. Doi: 10.1109/JCSSE.2016.7748933.

OJENIYI, J.A.; BALOGUN, M.O.; SANJO, F.; UGOCHUKWU, O. Development of a traffic analyzer for the detection of DDoS attack source. In: International Conference on Information and Communication Technology and its Applications (ICTA), 2016, Minna, Nigeria. **Anais**… ICTA, 2016, p. 111-116. Disponível em: <https://www.academia.edu/33450138/Development_of_a_Traffic_Analyzer_for_the_Detection_of_DDoS_Attack_Source>. Acesso em: 09 dez. 2018.

OUNACER, S.; TALHAOUI, M. A.; ARDCHIR, S.; DAIF, A.; AZOUAZI, M. A New Architecture for Real Time Data Stream Processing. **International Journal of Advanced Computer Science and Applications (IJACSA)**. [S.l.], v. 8, n 11, p. 44-51, 2017. Disponível em: < http://dx.doi.org/10.14569/IJACSA.2017.081106>. Acesso em: 03 set. 2018.

OUNACER, S.; TALHAOUI, M. A.; ARDCHIR, S.; DAIF, A.; AZOUAZI, M. Real-time Data Stream Processing Challenges and Perspectives. **International Journal of Computer Science Issues (IJCSI)**. [S.l.], v. 14, n 5, p. 6-12, set. 2017. Disponível em: <https://doi.org/10.20943/01201705.612>. Acesso em: 03 set. 2018.

PATEL, S. K.; SONKER, A. Rule-Based Network Intrusion Detection System for Port Scanning with Efficient Port Scan Detection Rules Using Snort. **International Journal of Future Generation Communication and Networking**. [S.l.], v. 9, p. 339-350, 2016. Doi: 10.14257/ijfgcn.2016.9.6.32. Disponível em: < https://www.researchgate.net/publication/305424244_Rule-Based_Network_Intrusion_Detection_System_for_Port_Scanning_with_Efficient_Port_Scan_Detection_Rules_Using_Snort>. Acesso em: 04 nov. 2018.

PERSICO, V.; PESCAPÉ, A.; PICARIELLO, A.; SPERLÍ, G. Benchmarking big data architectures for social networks data processing using public cloud platforms. **Future Generation Computer Systems**. [S.l.], v. 89, p. 98-109, 2018. Disponível em: < https://doi.org/10.1016/j.future.2018.05.068.>. Acesso em: 21 ago. 2018.

SAHNI, G.; SHARMA, S. Study of Various Anomalies and AnomalyDetection Methodologies in Wireless Sensor Network. **International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE).** [S.l.], v. 3, n. 5, p. 700-703, mai. 2013. Disponível em: <https://pdfs.semanticscholar.org/4d43/736fe63964cb487eace40929c22b0813f17b.pdf>. Acesso em: 16 set. 2018.

SCHONBERGER, V. M.; CUKIER, K. **Big Data**: A Revolution That Will Transform How We Live, Work, and Think. Boston: Houghton Mifflin Harcourt, 2014. 252 p.

SHACHAM, O.; GOTTESMAN, Y.; BERGMAN, A.; BORTNIKOV, E.; HILLEL, E.; KEIDAR, I. Taking omid to the clouds: fast, scalable transactions for real-time cloud

analytics. **Proceedings of the VLDB Endowment**. [S.l.], v. 11, p. 1795-1808, ago. 2018. Doi: 10.14778/3229863.3229868. Disponível em: <http://www.vldb.org/pvldb/vol11/p1795-shacham.pdf>. Acesso em: 05 jan. 2019.

SHARAFALDIN, I.; LASHKARI, A. H.; GHORBANI, A. A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP), 2018. **Anais**… Science and Technology Publications, 2018, p. 108-116. Doi: 10.5220/0006639801080116. Disponível em: < http://www.scitepress.org/Papers/2018/66398/66398.pdf>. Acesso em: 25 ago. 2018.

SILVA, E. C. da. **Detecção de Ataques de Negação de Serviço Utilizando Aprendizado de Máquina**. 2016. p. 128. Dissertação (Mestrado em Informática) - Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2016. Disponível em: < http://www.repositorio-bc.unirio.br:8080/xmlui/bitstream/handle/unirio/11522/MI%2011-2016.pdf?sequence=1&isAllowed=y>. Acesso em: 07 ago. 2018.

SOBREIRO, S. A. R. **Estudo de tecnologias para sistemas de Big Data**. 2018. p. 132. Dissertação (Mestrado em Engenharia da Informática) - Instituto Superior de Engenharia do Porto, Porto, 2018. Disponível em: < http://recipp.ipp.pt/bitstream/10400.22/11936/1/DM_SauloSobreiro_2018_MEI.pdf>. Acesso em: 24 set. 2018.

SONEWAR, P. A.; THOSAR, S. D. Detection of SQL injection and XSS attacks in three tier web applications. In: *International Conference on Computing Communication Control and automation (ICCUBEA),* 2016, Pune, India. **Anais**… IEEE, ago. 2016, p. 1-4. Doi: 10.1109/ICCUBEA.2016.7860069.

STOREY, V. C.; SONG, Il-Yeol. Big data technologies and Management: What conceptual modeling can do. **Data e Knowledge Engineering.** [S.l.], v. 108, p. 50-67, fev. 2017. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0169023X17300277?via%3Dihub>. Acesso em: 14 Out. 2018.

TREURNIET, J. A Network Activity Classification Schema and Its Application to Scan Detection. ***IEEE/ACM Transactions on Networking***. [S.l.], v. 19, n. 5, p. 1396-1404, out. 2011. Doi: 10.1109/TNET.2011.2109009.

VERMA, A.; KUMAR, D. Evaluating and Enhancing Efficiency of Recommendation System using Big Data Analytics. **International Research Journal of Engineering and Technology (IRJET).** [S.l.], v. 4, n. 3, p. 1518-1529, mar. 2017. Disponível em: <https://www.irjet.net/archives/V4/i3/IRJET-V4I3346.pdf>. Acesso em: 19 dez. 2018

WANG, B.; ZHENG, Y.; LOU, W.; HOU, Y. T. DDoS attack protection in the era of cloud computing and Software-Defined Networking. **Computer Networks.** [S.l.], v. 81, p.308-319, abr. 2015. Disponível em: <https://doi.org/10.1016/j.comnet.2015.02.026.> Acesso em: 21 nov. 2018.

WELZEL, A.; ROSSOW, C.; BOS, H. On measuring the impact of ddos botnets. In: EuroSec '14 Proceedings of the Seventh European Workshop on System Security, 2014, Amsterdam.

Anais… ACM New York, abr. 2014. Doi: 10.1145/2592791.2592794. Disponível em: < https://www.christian-rossow.de/publications/ddosbotnets-eurosec2014.pdf>. Acesso em: 12 ago. 2018.

WINGERATH, W.; GESSERT, F.; FRIEDRICH,S.; RITTER, N. Real-time stream processing for Big Data. **Information Technology**. [S.l.], v. 58, n. 4, p. 186–194, jun. 2016. Disponível em: <https://doi.org/10.1515/itit-2016-0002.>. Acesso em: 06 jan. 2019.

WU, X.; KUMAR, V.; QUINLAN, J.R.; GHOSH, J.; YANG, Q.; MOTODA, H.; MCLACHLAN, G.J.; NG, A.; LIU, B.; YU, P.S.; ZHOU, Z.; STEINBACH, M.; HAND, D.J.; STEINBERG, D. Top 10 algorithms in data mining. **Knowledge and Information Systems**, [S.1], v. 14, p. 1-37, 2008. Disponível em: <https://doi.org/10.1007/s10115-007-0114-2>. Acesso em: 03 out. 2018.

YAO, C.; LUO, X.; ZINCIR-HEYWOOD, A. N. Data analytics for modeling and visualizing attack behaviors: A case study on SSH brute force attacks. In: *Symposium Series on Computational Intelligence (SSCI),* 2017, Honolulu, HI. **Anais**… IEEE, dez. 2017, p. 1-8. Doi: 10.1109/SSCI.2017.8280913.

ZHANG, J. Z.; ZHOU, X. Z.; ZHANG,W. Z.; LIANG, Q. L.; XIANG, F. X. An Adaptive Filtration Based Defense Framework against Ddos. In*: International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC),* 2017, Guangzhou, China. **Anais**…IEEE, dez. 2017, p. 729-736.Doi: 10.1109/ISPA/IUCC.2017.00113.

ZHANG, K. W.; FAN, K. W.; EDWARDS, A.; YU, P. S. RS-Forest: A Rapid Density Estimator for Streaming Anomaly Detection. In: *International Conference on Data Mining,* 2014, Shenzhen, China. **Anais**… IEEE, dez. 2014, p. 600-609. Doi: 10.1109/ICDM.2014.45.

ZHOU, B.; LI, J.; GUO, S.; WU, J.; HU, Y.; ZHU, L. Online internet traffic measurement and monitoring using spark streaming. In: Global Communications Conference (GLOBECOM), 2017, Singapore. **Anais**… IEEE, dez. 2017, p. 1-6. Doi: 10.1109/GLOCOM.2017.8255000.