

UNIVERSIDADE FEDERAL DE SANTA MARIA  
CENTRO DE TECNOLOGIA  
BACHARELADO CIÊNCIA DA COMPUTAÇÃO

Brenda Salenave Santana

**UTILIZAÇÃO DE BIG DATA COMO SUPORTE À  
MINERAÇÃO DE REGISTROS DE REDE PARA SISTEMAS  
DE DETECÇÃO DE INTRUSÃO**

Santa Maria, RS  
2017

**Brenda Salenave Santana**

**UTILIZAÇÃO DE BIG DATA COMO SUPORTE À MINERAÇÃO DE REGISTROS  
DE REDE PARA SISTEMAS DE DETECÇÃO DE INTRUSÃO**

Trabalho de Conclusão de Curso apresentado  
ao Bacharelado Ciência da Computação da Uni-  
versidade Federal de Santa Maria (UFSM, RS),  
como requisito parcial para a obtenção do grau  
de **Bacharel em Ciência da Computação**

Orientador: Prof. Dr. Raul Ceretta Nunes

**Brenda Salenave Santana**

**UTILIZAÇÃO DE BIG DATA COMO SUPORTE À MINERAÇÃO DE REGISTROS  
DE REDE PARA SISTEMAS DE DETECÇÃO DE INTRUSÃO**

Trabalho de Conclusão de Curso apresentado  
ao Bacharelado Ciência da Computação da Uni-  
versidade Federal de Santa Maria (UFSM, RS),  
como requisito parcial para a obtenção do grau  
de **Bacharel em Ciência da Computação**

**Aprovado em 14 de dezembro de 2017:**



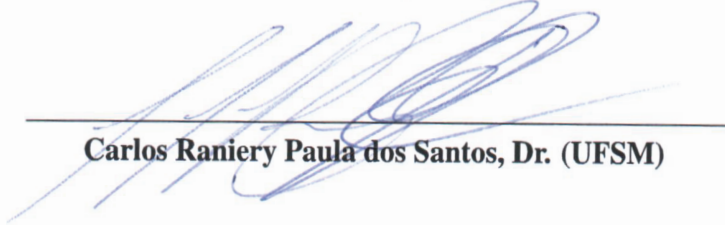
---

**Raul Ceretta Nunes, Dr.**  
(Presidente/Orientador)



---

**Ana Trindade Winck, Dr<sup>a</sup>. (UFCSPA)**



---

**Carlos Raniery Paula dos Santos, Dr. (UFSM)**

Santa Maria, RS

2017

## **DEDICATÓRIA**

*Aos meus afilhados Benhur e Isadora*



## AGRADECIMENTOS

*Nenhum estágio desta trajetória concluído sozinho. Em todos os momentos pude contar com o apoio de muitas pessoas e sou grata a todas elas. Desta forma, estes agradecimentos não poderão contemplar todos aqueles que de alguma forma contribuíram seja com este trabalho, seja em minha caminhada durante a graduação como um todo. Para esses, tenham a certeza que por mais que não expresse diretamente neste espaço, todos moram no meu coração e estão presentes em meu pensamento.*

*Primeiramente devo, antes de qualquer coisa, agradecer profundamente à minha mãe, Andria Salenave Santana, e ao meu pai, Carlos Roberto Soares Santana, que não só durante este trabalho, mas durante minha vida toda me apoiaram, me incentivaram e me animaram nos momentos difíceis. Vocês, hoje e sempre, são meus heróis e minhas referências. Agradeço também de coração, ao meu tio Wanderley Fernandes dos Santos por todo incentivo e apoio desde minha infância, e aos demais familiares.*

*Gostaria de agradecer a todos meus amigos, em especial a Karina Lopes da Silva pelos incontáveis auxílios ao longo de todos os anos de amizade, e também por todo suporte emocional dado mesmo durante as intensas madrugadas de estudo. Sou grata ainda por todo auxílio provido na execução deste trabalho, pelos meus amigos Maurício Matter Donato, Vinícius Fülber Garcia, Thales Nicolai Tavares e Luísa Perin Lucca. Agradeço também aqueles que mesmo não estando mais aqui, sempre torceram e me motivaram a chegar a este ponto.*

*A todos os professores que durante este percurso proveram os ensinamentos necessários tanto para a realização deste trabalho quanto para a vida. Tenho a honra e a felicidade de dizer que tive grandes mestres durante toda minha graduação. Agradeço em especial a professora Ana Trindade Winck por todas conversas, conselhos e ainda orientações na área a qual escolhi seguir. A esta agradeço ainda pelo exemplo profissional e de potencial representação feminina dentro da computação.*

*Por fim, faço um agradecimento ao professor Raul Ceretta Nunes que me acolheu em seu grupo de trabalho, orientando este trabalho de graduação que me permitiu um enorme crescimento.*

*“Eu tive sorte, eu tive azar. Tive motivos pra acreditar que existe gente com medo de arriscar.”*

(TÓPAZ)

## RESUMO

### UTILIZAÇÃO DE BIG DATA COMO SUPORTE À MINERAÇÃO DE REGISTROS DE REDE PARA SISTEMAS DE DETECÇÃO DE INTRUSÃO

AUTORA: BRENDA SALENAVE SANTANA

ORIENTADOR: RAUL CERETTA NUNES

Sistemas de Detecção de Intrusão referem-se aos meios técnicos de descobrir acessos indevidos em uma rede, indicando ações maliciosas. Porém, tais ferramentas requerem um alto nível de processamento, bem como elevada taxa de precisão e acurácia. Assim, sugere-se a utilização de técnicas de Mineração de Dados para análises preditivas e outros métodos de extração de informações sobre os fluxos de dados gerados. Como suporte para tal, tem-se infraestrutura de *Big Data* na solução do problema, como um conjunto de técnicas e procedimentos que abrangem a análise dinâmica do processamento de dados, provenientes de fontes distintas, em diferentes formatos. Desse modo, espera-se analisar e desenvolver uma ferramenta que faça uso dessa infraestrutura como suporte à mineração de fluxo de dados para o auxílio em Sistemas de Detecção de Intrusão. Como estudo de caso, foram utilizados dados providos por dispositivos de IoT do *Secure Water Treatment Dataset*. Assim, este trabalho destina-se à análise da adequação da infraestrutura para suportar a Mineração de Dados em tais fins. Foram estudadas então as ferramentas Spark e WEKA, aplicando os algoritmos *Naïve Bayes* e *Random Forest Tree*. Os resultados demonstram o potencial de uso de ambas as ferramentas no auxílio a detecção de anomalias em registros de rede, destacando ainda a acurácia e velocidade de processamento ao se trabalhar com Spark.

**Palavras-chave:** Mineração de Dados. Big Data. IDS. IoT.

## **ABSTRACT**

### **USE OF BIG DATA AS A NETWORK RECORD MINING SUPPORT FOR INTRUSION DETECTION SYSTEMS**

**AUTHOR: BRENDA SALENAVE SANTANA**

**ADVISOR: RAUL CERETTA NUNES**

Intrusion detection systems refer to technical means of discovering improper accesses on a network, indicating malicious actions. However, such tools require a high level of processing as well as high precision and accuracy rates. Thus, it is suggested to use techniques Data Mining for predictive analysis and other methods of extracting information on the generated data flows. As support for this, we have Big Data infrastructure in solution of the problem, as a set of techniques and procedures that cover dynamic analysis. In this way, it is expected to analyze and develop a tool that makes use of this infrastructure as support for the data flow mining for the aid in Intrusion Detection Systems. As a study case, the data is provided by IoT devices from the Secure Water Treatment Dataset. This work focuses on the data presented demonstrate the potential use of all the operations, without assistance, in order to support the data mining in such fins. Tools as Spark and Weka were studied, applying the algorithms Naïve Bayes and Random Forest Tree. The results present the potential of both tools for anomalies detection in network registers, highlighting security and speed of processing when Spark is used.

**Keywords:** Data Mining. Big Data. IDS. IoT.

## LISTA DE FIGURAS

Figura 2.1 – Estágios de um Ciclo de Vida <i>Big Data</i> .....	16
Figura 2.2 – Componentes Apache Spark .....	17
Figura 2.3 – Arquitetura Apache Spark .....	18
Figura 2.4 – Abordagens <i>Web Mining</i> . .....	20
Figura 3.1 – Arquitetura Hogzilla IDS .....	28
Figura 3.2 – Arquitetura de IoT com KDD .....	31
Figura 3.3 – Hierarquia entre Dado, Informação e Conhecimento.....	31
Figura 3.4 – Exemplos de Anomalias .....	35
Figura 4.1 – Estrutura.....	41
Figura 4.2 – Arquitetura Utilizada .....	41
Figura 5.1 – Máquinas Virtuais.....	47
Figura 5.2 – Cenário Base .....	48
Figura 5.3 – SWaT: Visão Geral dos Processos Testados .....	50
Figura 5.4 – Tempos de Execução do Algoritmo <i>Naïve Bayes</i> .....	53
Figura 5.5 – Tempos de Execução do Algoritmo <i>Random Forest Tree</i> .....	54
Figura 5.6 – Tempos de Execução do Algoritmo <i>Naïve Bayes II</i> .....	55
Figura 5.7 – Tempos de Execução do Algoritmo <i>Random Forest Tree II</i> .....	55

## LISTA DE TABELAS

Tabela 5.1 – Dados do Tráfego de Rede.....	51
Tabela 5.2 – Fluxos de Dados .....	52
Tabela 5.3 – Tempos de Execução do Algoritmo Naïve Bayes (segundos).....	53
Tabela 5.4 – Tempos de Execução do Algoritmo Random Forest Tree (segundos) .....	54

## LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CGI	Common Gateway Interface
CPS	Cyber Physical Systems
DAG	Directed Acyclic Graph
DDoS	Distributed Denial of Service
DNS	Domain Name System
DoS	Denial of Service
DM	Data Mining
DPI	Deep Packet Inspection
ETL	Extract, Transform, Load
GB	Gigabyte
HDFS	Hadoop Distributed File System
HTML	HyperText Markup Language
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
IWC	Inverse Weight Clustering
KDD	Knowledge Discovery in Databases
MAC	Media Access Control
P2P	Peer-to-Peer
RDD	Resilient Distributed Dataset
REPL	Read–Eval–Print Loop
SCADA	Supervisory Control and Data Acquisition
SQL	Structured Query Language
UFSCPA	Universidade Federal de Ciências da Saúde de Porto Alegre
UFSM	Universidade Federal de Santa Maria
WB	WEB Mining
WEB	World Wide Web
WEKA	Waikato Environment for Knowledge Analysis
WSN	Wireless Sensor Network

## SUMÁRIO

<b>1 INTRODUÇÃO</b> .....	12
<b>2 FUNDAMENTOS</b> .....	14
2.1 INTERNET DAS COISAS .....	14
2.2 BIG DATA.....	15
2.3 DESCOBERTA DE CONHECIMENTO EM BASES DE DADOS .....	19
<b>2.3.1 WEKA</b> .....	21
2.4 SEGURANÇA DA INFORMAÇÃO .....	22
<b>2.4.1 Ataques</b> .....	24
<b>3 TRABALHOS RELACIONADOS</b> .....	26
3.1 USO DE INFRAESTRUTURA BIG DATA NO AUXÍLIO DO PROCESSO DE ANÁLISE DE DETECÇÃO DE INTRUSÃO .....	26
3.2 USO DE KDD NA MINERAÇÃO DE DADOS DE WEB LOGS .....	30
3.3 MINERAÇÃO DE DADOS PROVIDOS POR DISPOSITIVOS IOT.....	34
3.4 CONSIDERAÇÕES.....	38
<b>4 PROPOSTA</b> .....	40
4.1 ARQUITETURA .....	40
<b>4.1.1 Arquitetura</b> .....	40
<b>4.1.2 Utilização de Infraestrutura Big Data como Suporte à Mineração de Dados</b> ....	42
<b>4.1.3 Mineração de Dados em Python</b> .....	43
4.2 ALGORITMOS.....	45
<b>4.2.1 Naïve Bayes</b> .....	45
<b>4.2.2 Random Forest Tree</b> .....	46
<b>5 EXPERIMENTOS</b> .....	47
5.1 AMBIENTE BASE .....	47
5.2 BASE DE DADOS .....	48
5.3 RESULTADOS E DISCUSSÃO .....	51
<b>6 CONCLUSÃO</b> .....	57
<b>REFERÊNCIAS</b> .....	60



# 1 INTRODUÇÃO

Constantes avanços na área tecnológica viabilizam o armazenamento de inúmeros volumes de dados que vem sendo gerados. Estudos feitos por Dell EMC (2017), estimam uma taxa de crescimento de dados mundiais em torno de 40% ao ano durante a próxima década, o que no ano de 2020 deverá alcançar cerca de 44 zettabytes ( $44 \times 2^{70}$  bytes). Para Brown et al. (2011), com tal avanço tecnológico, aliado à noções como Internet das Coisas (do inglês, *Internet of Things* – IoT), a análise de dados integra-se nas mais diversas áreas de aplicação, servindo como um importante foco de pesquisa.

De acordo com Grus (2015, p. 1),

vivemos em um mundo que está soterrado por dados. Os Websites rastreiam todos os cliques de todos os usuários. Seu *smartphone* está fazendo um registro da sua localização e sua velocidade a cada segundo diariamente. Atletas avaliados usam pedômetros com esteroides que estão sempre registrando suas batidas do coração, hábitos de movimentos, dieta e padrões do sono. Carros inteligentes coletam hábitos de direção, casas inteligentes coletam hábitos de moradia e marqueteiros inteligentes coletam hábitos de compra. A própria internet representa um diagrama grande de conhecimento que contém (entre outras coisas) uma enorme enciclopédia de referências cruzadas: bases de dados específicos de domínio sobre filmes, música, resultados de esportes, máquinas de pinball, memes e coquetéis; e muitas estatísticas do governo.

Assim, ascensão de dados gerados por dispositivos IoT representa desafios como uma maior preocupação em relação à segurança e privacidade dos sensores utilizados e dos dados a serem armazenados. O advento da Internet das Coisas também pode vir a aumentar ainda os riscos envolvendo ameaças à segurança das empresas de todo o mundo. Deste modo, o uso de meios técnicos como Sistemas de Detecção de Intrusão (IDSs) auxiliam na descoberta de acessos indevidos em uma rede, que podem indicar ações de caráter malicioso. Os IDSs são usados para detectar diversos tipos de comportamentos maliciosos que podem comprometer a segurança e a confiabilidade de um sistema. Com o crescimento acentuado de tecnologias torna-se cada vez mais difícil a implantação de tais sistemas. Sistemas de Detecção de Intrusão podem ser divididos em: baseados em assinaturas e os que são baseados em anomalias. Os baseados em assinaturas identificam ataques através da análise de assinaturas de ataque definidas com base no conhecimento prévio (padrão de ataques). Já os IDSs baseados em anomalias analisam comportamentos e identificam perturbações nestes, gerando um alarme sempre que o comportamento não segue o esperado.

Conforme sistemas computacionais se tornam cada vez mais complexos, surgem fragilidades exploráveis provenientes de erros de projeto e desenvolvimento. Se faz necessário

dessa forma, o uso de técnicas que monitorem constantemente as limitações do sistema. As análises preditivas e outros métodos de extração de informações sobre um grande conjunto de dados, tratado na área de tecnologia da informação pelo termo *Big Data*, surgem como um bom recurso para tal fim.

Conforme Amaral (2016), *Big Data* envolve o uso de diversos tipos de conceitos e tecnologias, tais como computação em nuvem, virtualização, infraestrutura, armazenamento, processamento, governança e gestão de projetos. A expressão *Big Data* refere-se então ao conjunto de diversas técnicas auxiliam um processo de tomada de decisão confiável devido a união de métodos previamente certificados. Por consequência, melhores decisões implicam em uma maior eficiência operacional, redução de riscos e custos. Sendo assim, uma análise adequada e minuciosa de tais conjuntos permite encontrar novas correlações entre os dados, como padrões de ocorrência de eventos, que auxiliam em uma rápida tomada de decisão. Tem-se então o aproveitamento do uso de uma infraestrutura de *Big Data* como suporte à busca de padrões em conjuntos de dados. Ainda hoje, encontram-se dificuldades no reconhecimento de acessos maliciosos na rede, porém com o uso de técnicas de Mineração de Dados torna-se viável traçar os padrões e assim descobrir invasões com maior efetividade. Desse modo, espera-se que através da utilização de tais técnicas, juntamente com a averiguação da viabilidade do uso de infraestrutura de *Big Data* na solução do problema, possa-se identificar e mapear padrões de intrusão na rede a fim de garantir segurança.

O objetivo geral deste trabalho é realizar uma análise do uso de infraestruturas de *Big Data* como suporte à prospecção de dados que auxiliem na detecção de anomalias em fluxos de dados transmitidos por dispositivos IoT de modo a prover uma maior segurança as informações ali contidas. Para isso, é proposta a utilização de ferramentas de Mineração de Dados no cenário de uma infraestrutura de *Big Data*, de modo a captar e analisar dados gerados por dispositivos IoT.

O trabalho apresenta-se dividido da seguinte forma: no Capítulo 2 são descritos os fundamentos necessários para compreensão do presente trabalho, seguidamente pelo estado da arte do tema abordado. Já no Capítulo 3, enfoca-se em trabalhos correlatos que apresentam pesquisas análogas a proposta. O Capítulo 4 apresenta a proposta de desenvolvimento elaborada. O Capítulo 5 expõe os resultados alcançados diante dos cenários de testes utilizados para os experimentos de validação da proposição exposta. Por fim, o Capítulo 6 apresenta as considerações finais do trabalho.

## 2 FUNDAMENTOS

Neste Capítulo são descritos os fundamentos necessários para o entendimento deste trabalho, bem como o nível atual do desenvolvimento de pesquisas que versam sobre o tema. Este Capítulo encontra-se organizado da seguinte maneira: a Seção 2.1 a conceituação de Internet das Coisas e o volume de dados por esta gerado; Assim a 2.2 explica um cenário *Big Data* e ferramentas de utilização neste contexto; A Seção 2.3 conceitua a Descoberta de Conhecimento em Bases de Dados (do inglês, *Knowledge Discovery in Databases* – KDD) e possibilidades de aplicação; e por fim a Seção 2.4 demonstra os desafios encontrados na área de Segurança da Informação.

### 2.1 INTERNET DAS COISAS

Computação ubíqua, ou pervasiva é a expressão utilizada para descrever a onipresença da informática no cotidiano dos indivíduos. Para Weiser (1999) o aumento nas funcionalidades e na disponibilidade de serviços de computação para os usuários finais era previsto, entretanto a visibilidade destes serviços tendia a ser menor. De acordo com este autor, a computação não seria exclusiva a um computador e, sim, a diversos dispositivos conectados entre si.

A Internet das Coisas, refere-se ao modo como dispositivos são conectados e comunicam-se entre si e com o usuário através de sensores e ferramentas que transmitem dados através em uma rede. Conforme Doody e Shields (2012), a Internet se conecta com diferentes rotinas através da rede de objetos conectados. De acordo com Said e Masud (2013), IoT pode ser definida como uma comunicação análoga a troca de informações ocorrente entre computadores comuns à Internet, porém entre dispositivos distintos. Esta permite que diferentes objetos troquem dados e informações para a fim de realizar determinadas tarefas. A Internet das Coisas tem por base de funcionamento sensores e dispositivos, e nesta torna-se possível a comunicação entre as “coisas” ligadas a uma mesma rede. No contexto de IoT, “coisas”, referem-se a quaisquer dispositivos capazes de se conectar e trocar informações através de uma rede.

De acordo com Taherkordi et al. (2017) ao se conectar um grande número de objetos físicos equipados com sensores à Internet gera-se um novo e volumoso ecossistema de dados. Os dados gerados por dispositivos IoT possuem propriedades que se adequam ao paradigma de *Big Data*, incluindo o Volume em termos de geração de massa de dados; Variedade sob a forma

de uma mistura de dados IoT estruturados e não estruturados; e Velocidade referente às diferentes frequências de geração de dados entre dispositivos IoT e requisitos de tempo diferentes para entrega de dados. Porém para Fu et al. (2011) a Internet das Coisas apresenta uma rede vulnerável a ataques maliciosos devido à abertura de implantação e recursos limitados.

## 2.2 BIG DATA

*Big Data* é o termo utilizado para referenciar o massivo crescimento da quantidade de dados gerados. Para Goldschmidt, Passos e Bezerra (2015), este termo compreende também todas as técnicas e iniciativas de tratamento, integração e análise de dados provenientes de diversas fontes em diferentes mídias e formatos.

*Big Data* é uma expressão que descreve o grande volume de dados, sendo estes estruturados ou não estruturados, que são gerados a cada segundo. Contudo, mesmo não possuindo uma definição concreta, este foi criado para atender a uma requisição cada vez mais forte e abrangente de processamentos sobre bases de dados que são caracterizados por uma ampla complexidade e, principalmente, por grandes dimensões. A essência dessa concepção se propõe a oferecer o uso de infraestrutura como suporte à este tipo de problema através de ferramentas, como *softwares* e serviços, além de paradigmas e modelos de computação.

Além do Volume de dados associados, *Big Data* e suas técnicas estão ligados também a outros fatores que aumentam a complexidade envolvida. Para Taurion (2013) tem-se ainda *Velocidade* e *Variiedade*. O primeiro termo refere-se ao tempo em que determinado problema deve ser processado mediante a constante coleta de novas informações, e o segundo é atribuído à diversidade de informações e fontes de dados com que se pode trabalhar. Os fundamentos citados, são denominados como as três dimensões básicas que constituem *Big Data*, usualmente chamados de 3V's. Outros autores mencionam ainda a existência de outros 2V's, sendo estes: Veracidade (necessidade de uso e recuperação de informações que sejam válidas no contexto em que estejam inseridas) e Valor (busca por conhecimento que agregue informações úteis ao cenário de aplicação).

Desse modo, a análise de informações contidas em um cenário de *Big Data* tem criado uma demanda cada vez maior de recursos computacionais. Por outro lado, essa exigência pode se tornar um problema, uma vez que a aquisição de infraestruturas que sejam capazes de atender aos requisitos esperados acaba, por vezes, tornando-se inviável. Além disso, as soluções para esses problemas tornam-se ineficientes à medida que o volume de dados aumenta,

principalmente com a utilização de métodos tradicionais de processamento.

Segundo Mehmood et al. (2016), o ciclo de vida *Big Data* é desenvolvido tal qual como exibido na Figura 2.1, onde na etapa de *Geração de Dados* as informações podem ser providas por fontes diversas e distribuídas. De modo geral, estes dados são extensos, diversos e complexos. O estágio de *Armazenamento de Dados*, refere-se além do recolhimento dos dados, ao gerenciamento destes em larga escala. Neste passo, também devem ser providas diversas interfaces de interação e análise dos dados armazenados. Por último, a etapa de *Processamento dos Dados* refere-se aos processos de coleta e transmissão dos dados, seu pré-processamento e extração de conhecimento.

Figura 2.1: Estágios de um Ciclo de Vida *Big Data*



Fonte: Mehmood et al. (2016).

Como mencionado, *Big Data* retrata a vasta quantidade de informação gerada diariamente através dos mais variados tipos de dispositivos eletrônicos e o tratamento analítico dessa informação através de ferramentas tecnológicas, com o intuito de se obter padrões, correlações e percepções que podem auxiliar em tomadas de decisões nas mais distintas áreas. Além de lidar com tamanhos volumes de dados, estas soluções precisam ainda lidar com distribuição de processamento e elasticidade, isto é, suportar aplicações com volumes de dados que crescem substancialmente em pouco tempo. Desta forma tem-se a utilização de tecnologias relacionadas a sua aplicação na resolução de problemas.

A existência de ferramentas que atuem como recursos auxiliares em soluções *Big Data*, torna-se um recurso viável de utilização de forma a aferir as informações a serem analisadas. O Apache Spark<sup>1</sup> é um *framework* de código aberto voltado para processamento de soluções em *Big Data* de forma paralela e distribuída. Esta é uma ferramenta para processamento em cluster desenvolvido usando a linguagem Scala<sup>2</sup>. A principal característica do Spark<sup>®</sup> é fornecer o pro-

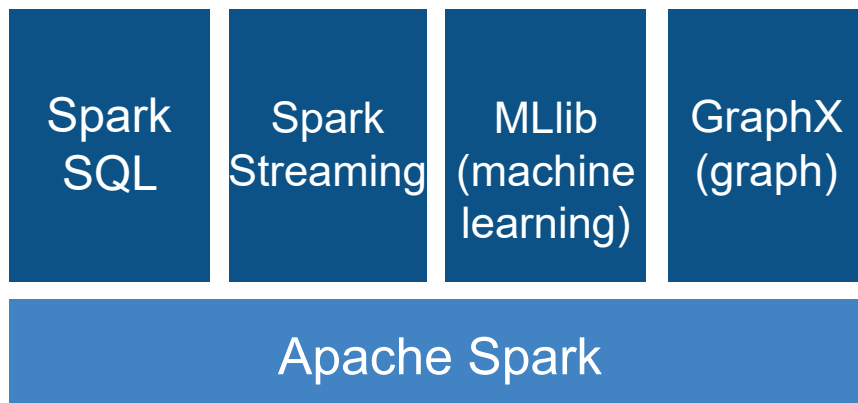
<sup>1</sup> Disponível em: <https://apache.apache.org/>.

<sup>2</sup> Disponível em: <https://www.scala-lang.org>

cessamento em memória principal, diferentemente do seu similar Hadoop que realiza diversos acessos a disco durante a execução. Ao realizar uma tarefa, o Spark mantém seus dados em memória. Essa característica resulta principalmente em um alto desempenho em processamento, por exemplo, com a execução de algoritmos iterativos em tarefas de aprendizado de máquina, onde um dado é analisado repetidas vezes.

O Spark é constituído por diversos componentes para diferentes tipos de processamentos, todos estes são construídos sobre o Spark Core. Este componente é responsável por proporcionar funções básicas para o processamento. Destes, destacam-se os elementos a seguir descritos e presentes na Figura 2.2.

Figura 2.2: Componentes Apache Spark



Fonte: <http://spark.apache.org/>.

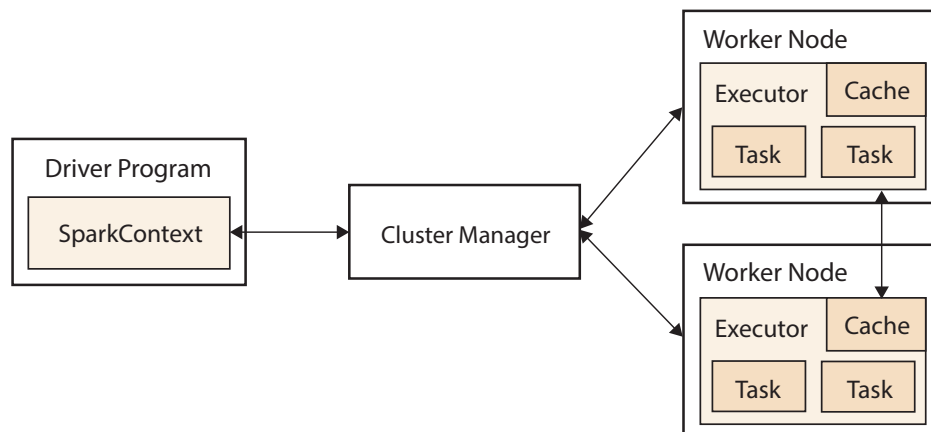
- a) Spark SQL: Possibilita utilização de SQL na realização de consultas e processamento sobre os dados no Spark. Permite a pesquisa de dados estruturados dentro dos programas Spark, utilizando SQL ou uma *Application Programming Interface* (API) DataFrame similar.
- b) Spark Streaming: Possibilita o processamento de fluxos em tempo real, ou seja, traz a API integrada do Apache Spark para o processamento de fluxos, permitindo a escrita de trabalhos de transmissão da mesma forma que em lote. Oferece ainda tolerância a falhas, recuperando-se do trabalho perdido e o estado do operador sem exigir código extra.
- c) MLib: Biblioteca de aprendizado de máquina, com diferentes algoritmos para as mais diversas atividades. O MLib se encaixa nas API da Spark e interagindo com o NumPy no

Python e as bibliotecas do R<sup>3</sup>. É possível utilizar ainda qualquer fonte de dados Hadoop, facilitando a conexão com os fluxos de trabalho deste.

- d) GraphX: Realiza o processamento sobre gráficos, unificando Extração, Transformação e Carregamento (*Extract, Transform, Load – ETL*), análise exploratória e computação iterativa em um único sistema. Permite ainda a visualização de gráficos e coleções, transformando e reunindo-os com *Resilient Distributed Datasets (RDDs)* de forma eficiente a escrever algoritmos iterativos personalizados.

Um *cluster* Spark é dividido em basicamente: *Driver Program*, *Cluster Manager*, *Work Node* tal como demonstrado na Figura 2.3. Este primeiro contém o *SparkContext*, o qual é responsável por coordenar a execução de uma aplicação no ambiente Spark. Assim, ao executar uma tarefa comunica-se com o segundo elemento, o *Cluster Manager* onde este aloca os recursos necessários aos nodos auxiliares para execução. Por último o *Work Node*, composto por Executores, recebe as demandas e as executa.

Figura 2.3: Arquitetura Apache Spark



Fonte: <http://spark.apache.org/>.

A arquitetura do Apache Spark é composta por três camadas principais: API, armazenamento dos dados e o gerenciador de recursos. A API oferece suporte para o desenvolvimento de aplicações baseadas no Spark de acordo com ferramentas para as linguagens Scala, Python<sup>4</sup> e Java<sup>5</sup>. O armazenamento do *framework* se dá pelo uso do *Hadoop Distributed File System*<sup>6</sup>

<sup>3</sup> Linguagem e ambiente de desenvolvimento integrado para cálculos estatísticos e gráficos.

<sup>4</sup> Disponível em: <https://www.python.org>

<sup>5</sup> Disponível em: <https://www.oracle.com/br/java/index.html>

<sup>6</sup> Disponível em: <https://hbase.apache.org/0.94/book/arch.hdfs.html>

(HDFS), um sistema de arquivos altamente tolerante a falhas projetado para executar em hardware padrão de baixo custo, ainda que exista suporte para uma série de serviços de armazenamento compatíveis com o Apache Hadoop<sup>7</sup>. Já o gerenciador de recursos define a ferramenta que faz a manutenção dos recursos do ambiente, sendo suportado um gerenciador Standalone do Spark, além do YARN<sup>8</sup> e a ferramenta Mesos<sup>9</sup>.

### 2.3 DESCOBERTA DE CONHECIMENTO EM BASES DE DADOS

A análise dos grandes volumes de dados torna-se cada vez mais complexa e inviável de ser realizada sem o suporte de ferramentas computacionais apropriadas. Assim, de forma a suprir esta necessidade tem-se a área denominada Descoberta de Conhecimento em Bases de Dados. Em uma tradução adaptada de Fayyad, Piatetsky-Shapiro e Smyth (1996) a definição de KDD é dada como “Um processo não trivial, interativo, iterativo, para identificação de padrões compreensíveis, válidos, novos e potencialmente úteis a partir de grandes conjuntos de dados”.

A etapa de Mineração de Dados (do inglês, *Data Mining* – DM) compreende a busca efetiva por conhecimentos úteis no contexto da aplicação de KDD. Mineração de Dados consiste no processo de explorar volumes de dados à procura de padrões consistentes, de modo a encontrar relações sistemáticas entre variáveis, detectando assim novos subconjuntos de dados. Goldschmidt, Passos e Bezerra (2015) afirma que o valor dos dados armazenados está tipicamente ligado à capacidade de se extrair conhecimento de mais alto nível a partir deles, ou seja, informação útil que sirva para apoio à tomada de decisão e/ou para exploração e melhor entendimento do fenômeno gerador de dados. Os métodos de prospecção de dados possuem portanto o objetivo de transformar informações dispersas em um conjunto de dados em conhecimento.

A Web é uma das maiores e mais heterogêneas bases de dados disponíveis. Assim, mensurar o valor das informações contidas traz grandes desafios. Técnicas existentes de KDD destinam-se a auxiliar o processo de obtenção de conhecimento a partir de tais bases. Neste contexto, tais técnicas são denominadas *Web Mining* (WB). Estas, agrupam ainda abordagens distintas, cada qual com seus métodos e ferramentas, sendo elas: *Web Mining* de Conteúdo, *Web Mining* de Estrutura e *Web Mining* de Uso, tal como ilustrado na Figura 2.4.

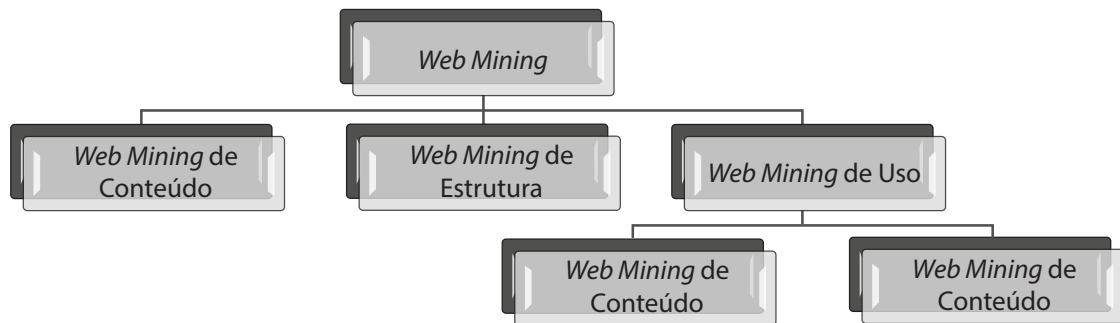
<sup>7</sup> Disponível em: <https://hadoop.apache.org>

<sup>8</sup> Tecnologia de gerenciamento de *clusters*.

<sup>9</sup> Projeto de código aberto da Apache, utilizado para gerenciamento de *clusters* de computadores.



Figura 2.4: Abordagens *Web Mining*.



Fonte: Goldschmidt, Passos e Bezerra (2015).

Para Chakrabarti (2002), *Web Mining* de Conteúdo constitui-se na aplicação de técnicas de Mineração de Dados para descoberta de conhecimento em documentos Web. Ou seja, o processo de mineração do conteúdo da Web consiste em analisar textos, imagens e outros componentes presentes em documentos Linguagem de Marcação de Documentos (do inglês, HyperText Markup Language – HTML) costumam apresentar extensas informações, a fim de formatar visualmente o conteúdo. Para fins de aplicação desta técnica, tais informações são ignoradas e apenas seu conteúdo é considerado como base de dados para o processo de descoberta de conhecimento. Esta técnica é essencialmente utilizada como meio de facilitar o acesso ao conteúdo predominantemente desestruturado encontrado nestes tipos de documento. Dentre as principais utilizações, destacam-se a categorização automática de páginas HTML e indexação do conteúdo.

Já a mineração de estrutura de *links* da Web propõe-se ao estudo do relacionamento entre páginas da Web através de seus *hiperlinks*, ou seja, visa o desenvolvimento de técnicas para aproveitar o julgamento coletivo da qualidade de páginas Web que está implícito em na estrutura de ligações. Alguns hiperlinks são utilizados de forma a organizar uma massiva quantidade de informações ou mesmo para facilitar a navegação do próprio site, já outros apontam para páginas de outros sites. Quando diversas referências apontam a uma mesma página, esta é considerada uma fonte de conteúdo de qualidade superior as demais páginas que não recebem tantos apontamentos. Atualmente, os principais motores de busca fazem uso desta informação para auxiliar o processo de ordenação de resultados de uma pesquisa.

Dentro das abordagens de WB, o *Web Mining* de Uso objetiva-se a realizar a análise de dados gerados por sistemas de informação baseados na Web relacionados com o acesso de

páginas. Tais análises envolvem a descoberta de padrões sobre dados muitas vezes armazenados em *logs* de servidores Web e de aplicação. De acordo com Grace et al. (2011), o conteúdo de arquivos de *log* são comumente utilizados neste tipo de mineração. O processo de WB de Uso dividido em três etapas principais: pré-processamento (tratamento dado ao conjunto de *logs* a serem analisados), mineração dos dados (aplicação dos algoritmos) e pós-processamento (análise e interpretação dos dados).

### 2.3.1 WEKA

Waikato Environment for Knowledge Analysis<sup>10</sup> (WEKA) é uma coleção de algoritmos de aprendizagem de máquina para tarefas de Mineração de Dados. Os algoritmos podem ser diretamente aplicados à um conjunto de dados ou chamados de seu próprio código Java. O WEKA contém ferramentas para realização do pré-processamento de dados, classificação, regressão, agrupamento, regras de associação e visualização. Sendo adequado também ao desenvolvimento de novos esquemas de aprendizado de máquinas.

A ferramenta é desenvolvida através da linguagem de programação Java e possui uma interface de interação com arquivos de dados e produção de resultados visuais. Este *software* possui também uma API geral, permitindo assim sua incorporação a aplicativos de forma a realizar tarefas de Mineração de Dados automatizadas. O WEKA fornece ainda, via linha de comando e interface gráfica, o que facilita a manipulação de algoritmos.

O *workbench* do WEKA inclui métodos para os principais problemas de Mineração de Dados. A partir disto, todos os algoritmos implementados realizam contribuições na forma de uma única tabela relacional que pode ser lida a partir de um arquivo ou gerada por uma consulta de banco de dados, onde armazenam seus resultados.

De acordo com Eibe et al. (2016), uma maneira de utilizar o WEKA é aplicar um método de aprendizagem a um conjunto de dados e analisar sua saída para saber mais sobre os dados. Outro é usar modelos aprendidos para gerar previsões em novas instâncias. E um terceiro é aplicar diversos vários *learners* distintos e comparar seu desempenho para eleger um para previsão. Muitos métodos apresentam parâmetros ajustáveis, acessíveis através da interface interativa da ferramenta. De acordo com os desenvolvedores, a ferramenta ainda é capaz de lidar com o processamento de informações volumosas, características de ambientes *Big Data*.

---

<sup>10</sup> Disponível em: <https://www.cs.waikato.ac.nz/ml/weka/>

## 2.4 SEGURANÇA DA INFORMAÇÃO

Segurança da informação é um conceito que abrange o uso de ferramentas diversas para proteger informações sigilosas e garantir que estas estejam a disposição apenas para aqueles autorizados. De acordo com Soares et al. (2013) segurança da informação compreende um amplo conjunto de técnicas que visam fornecer serviços de segurança, como confidencialidade, integridade, e autenticação, a informação a ser protegida. Assim conceito pode ser estendido e aplicado a sistemas computacionais, sendo um sistema seguro se fornecer serviços de segurança desejados às informações que administra. Desta forma comprovar a segurança sistema de informação é seguro não é fácil.

Frequentemente, novas técnicas para comprometer ambientes computacionais são criadas, assim o mercado de segurança da informação deve acompanhar esta velocidade, e até mesmo estar a frente para não atuar de forma reativa. Deste modo, a implementação de uma política apropriada de proteção é fundamental em uma arquitetura de segurança.

Sistemas de Detecção de Intrusão referem-se aos meios técnicos de descobrir acessos indevidos em uma rede, que podem indicar ações de caráter malicioso. Com o acentuado crescimento das tecnologias de infraestrutura, tanto nos serviços quanto nos protocolos de rede a implantação de Sistemas de Detecção de Intrusão torna-se cada vez mais difícil. Esta pode ser definida como um processo de monitoramento de eventos que ocorrem em um sistema de computação ou em uma rede, com o objetivo de analisar possíveis incidentes, violações ou iminências de destas às regras de segurança do ambiente monitorado. Tais eventos podem ser provenientes desde a ação de *malwares* (código nocivo) até mesmo à ataques que visam o ganho não autorizado do ambiente em questão.

A utilização de IDSs como sistema de identificação pode envolver desde alertas ao administrador da rede e exames preventivos até a obstrução de uma conexão suspeita. Ou seja, o processo de detecção de intrusão envolve ainda ações responsivas de maneira preventiva à atividades suspeitas que possam interferir nos princípios da integridade, confiabilidade e disponibilidade. Sistemas de Detecção de Intrusão geralmente analisam arquivos locais em busca de registros de tentativas mal-sucedidas de conexão à máquina.

Em um modelo básico de IDS é possível incluir elementos, de modo a melhorar as funcionalidades do mesmo. Assim, primeiramente as decisões provenientes do sistema são baseadas sob a coleta de dados realizada, sendo as decisões tomadas somente na existência

de valores significativos de informações em sua base que confirmam a maliciosidade daquele computador. Os dados são armazenados por tempo indefinido, podendo ser utilizados como referência, ou mesmo para um processamento posterior. Dados coletados com informações homogêneas, são fundamentais para o trabalho do IDS, dada a necessidade de comparação em alguns modelos. Assim, um ou mais algoritmos são executados, de forma a buscar evidências para a tomada de decisão contra as atividades suspeitas.

Geralmente os IDS são controlados por configurações que especificam as possíveis ações a serem tomadas. Estas, ditam onde os dados serão coletados para análise, assim como qual deverá ser a resposta a cada tipo de tentativa intrusão. Um melhor ajuste de configurações auxilia uma maior proteção ao ambiente, porém, deve-se ainda observar o sobreajuste (*overfitting*) de forma a evitar que o modelo se adapte exclusivamente ao conjunto de dados anteriormente observado, se mostrando ineficaz na previsão de novos resultados. Estes cuidados devem ser tomados, principalmente ao se tratar de dados muito heterogêneos, como é o caso de informações geradas por IoT.

Sistemas de Detecção de Intrusão são responsáveis por todo o tipo de saída, ou seja, desde respostas automáticas, alerta de atividades suspeitas ao administrador até notificações ao usuário. Entretanto estes alertas não devem ser totalmente conclusivos, podendo haver erros tanto de análise como de configuração, o que pode gerar os chamados falsos positivos, que são alertas, ou ações, em resposta a evidências encontradas porém de forma equivocada. Uma configuração frágil pode gerar falsos negativos, que se conceitua pela falta de alerta ou decisão para um ataque real. Assim, busca-se que o IDS obtenha o menor número de falsos positivos e falsos negativos possível.

De acordo com Di Pietro e Mancini (2008), IDSs baseados em assinatura utilizam-se de técnicas de correspondência de padrões contidos banco de dados de assinaturas de ataques conhecidos e buscam combinar essas assinaturas com os dados analisados gerando um alarme ao encontrar correspondências. Em contrapartida atualmente conforme os mesmos, ataques visam principalmente explorar vulnerabilidades a nível de aplicação: assim, a carga útil contém a informação mais importante para diferenciar o tráfego normal da atividade anômala. Deste modo Sistemas de Detecção de Intrusão de rede são considerados uma segunda linha de defesa eficaz contra ataques baseados em rede dirigidos a sistemas computacionais. Por sua vez, IDSs baseados em anomalias podem levar em consideração cabeçalhos de pacotes, a carga útil ou mesmo uma combinação de ambos. Os autores argumentam ainda que as abordagens baseadas

na carga útil estão se tornando os métodos mais eficazes para detectar ataques.

Para Lee e Stolfo (1998), a prevenção de intrusão por si só não é suficiente pois, à medida que os sistemas se tornam cada vez mais complexos, sempre há fraquezas exploráveis devido a erros de projeto e programação ou várias técnicas de invasão “socialmente criadas”. Se faz necessário, dessa forma, o uso de técnicas que monitorem constantemente as limitações do sistema. As análises preditivas e outros métodos de extração de informações sobre um grande conjunto de dados (estruturados ou não), tratado na área de tecnologia da informação pelo termo *Big Data*, surgem como um bom recurso para tal fim.

### 2.4.1 Ataques

A segurança da informação trata diretamente da proteção de um conjunto de dados, no sentido de preservar o valor que estes possuem para um indivíduo ou mesmo uma organização. As propriedades básicas desta área, visam contemplar a confidencialidade, integridade, disponibilidade e autenticidade das informações. Entretanto, rotineiramente redes de computadores recebem ataques que buscam ferir tais propriedades. Tais ataques podem ser:

- a) DoS: Estes podem ser interpretados como “Ataques de Negação de Serviços”, do inglês *Denial of Service* (DoS). Consiste em tentativas de fazer com que computadores, servidores *World Wide Web* (Web) por exemplo, tenham dificuldade ou mesmo sejam impedidos de executar suas tarefas. Assim, ao invés de ‘invadir’ o computador ou mesmo infectá-lo o atacante faz com que a máquina receba tantas requisições que esta chega ao ponto de não conseguir processá-las. Assim Tanenbaum (2003), apresenta estes como ataques em que o objetivo do intruso é desativar o destino em vez de roubar dados, fazendo com que em geral, os pacotes solicitados possuam endereços de origem falsos, para que o intruso não possa ser rastreado com facilidade.
- b) DDoS: *Distributed Denial of Service* (DDoS) é um tipo de ataque DoS de grandes dimensões, ou seja, que utiliza múltiplos (até mesmo milhares) de computadores para atacar uma determinada máquina, distribuindo a ação entre elas. Tanenbaum (2003), apresenta este como uma variante ainda pior do primeiro, aumentando o poder de ataque do intruso, e ainda reduzindo a chance de detecção, pois os pacotes provém de um grande número de máquinas pertencentes a usuários insuspeitos.
- c) Força Bruta: Consiste em adivinhar, por tentativa e erro, um nome de usuário e senha e,

assim, executar processos e acessar sites, computadores e serviços com o nome e com os mesmos privilégios deste usuário. De acordo com Knudsen e Robshaw (2011), este é o único ataque que sempre pode ser realizado contra qualquer cifra de bloco na busca de chaves de acesso. Fazendo assim com que nem mesmo um *design* inteligente de implementação possa evitá-lo.

- d) Spoofing: É um ataque que consiste em mascarar (*spoof*) pacotes do protocolo de Internet (*Internet Protocol – IP*) ou mesmo do *Domain Name System* (DNS) utilizando endereços de remetentes falsificados. Tanenbaum (2003), apresenta então a ação de enganar um servidor DNS fazendo-o instalar um falso endereço IP como *spoofing* de DNS. Devido às características do protocolo IP, o reencaminhamento de pacotes é feito com base numa premissa muito simples: o pacote deverá ir para o destinatário e não há verificação do remetente. Assim, a falsificação do endereço de origem é feita através de uma manipulação simples do cabeçalho IP. Ou seja, vários computadores podem enviar pacotes fazendo-se passar por um determinado endereço de origem, fato que representa uma séria ameaça para os sistemas baseados em autenticação pelo endereço IP.

Além destes, existem ainda diversos outros tipos de ataques que podem vir a comprometer a segurança dos dados. Vale ressaltar também que a ocorrência de diferentes tipos de ataques, podem ocorrer de forma conjunta. Estes tipos de investidas, podem proporcionar ocorrências de anomalias no tráfego de uma rede, os quais podem vir a serem detectados por meio do uso de IDSs.

### 3 TRABALHOS RELACIONADOS

Atualmente, o uso de dispositivos IoT vem crescendo de forma notável devido ao impacto em potencial destes. Deste modo a preocupação com a segurança dos dados gerados torna-se crucial, sendo assim alvo de diversos estudos na área. Neste Capítulo são descritos trabalhos correlatos os quais realizaram pesquisas análogas aos objetivos do presente trabalho. Este Capítulo encontra-se organizado da seguinte maneira: a Seção 3.1 trata sobre estudos envolvendo a utilização de infraestrutura de *Big Data* para o suporte à detecção de intrusões; a Seção 3.2 aborda estudos direcionados a aplicação de técnicas de Mineração de Dados em registros de rede; a Seção 3.3 concentra-se em algoritmos e métodos alvíres para detecção de intrusão em redes IoT; e por fim, a Seção 3.4 destina-se a uma breve síntese dos trabalhos citados e perspectivas sobre o tema.

#### 3.1 USO DE INFRAESTRUTURA BIG DATA NO AUXÍLIO DO PROCESSO DE ANÁLISE DE DETECÇÃO DE INTRUSÃO

Em Suthaharan (2014), apresenta-se um trabalho realizado com enfoque no problema da classificação de intrusão de dados volumosos gerados em um tráfego de rede. Neste, os autores discutem os desafios do sistema apresentados pelos problemas do *Big Data* associados à predição de intrusão de rede. A previsão de um possível ataque deste tipo em uma rede exige a coleta contínua de dados de tráfego e a aprendizagem de suas características. Porém, o processo de coleta leva a problemas causados pelas propriedades de volume, variedade e velocidade de *Big Data*. A aprendizagem das individualidades da rede requer técnicas de aprendizado de máquina que capturem o conhecimento global dos padrões de tráfego. Assim, os autores discutem os problemas e os desafios na gestão da classificação do *Big Data* usando técnicas de aprendizagem de representação geométrica e as modernas tecnologias de rede *Big Data*. Em particular, este artigo aborda as questões relacionadas à combinação de técnicas de aprendizagem supervisionada, técnicas de aprendizagem de representação, técnicas de aprendizagem ao longo da vida e tecnologias *Big Data* (por exemplo, Hadoop, Hive e Cloud) para resolver problemas de classificação de tráfego de rede.

Assim, Suthaharan (2014) sugere a integração de ferramentas modernas tais como Hadoop e tecnologias de nuvem, com o uso de máquinas de suporte de vetores para predição de

intrusões em redes de computadores através de estratégias de classificação *Big Data*. Além disso, os autores sugerem adotar uma estrutura de aprendizagem ao longo da vida da máquina para resolver os problemas associados ao parâmetro de continuidade.

O Hogzilla<sup>11</sup> é um Sistema de Detecção de Intrusão de código aberto, que faz uso de tecnologias *Big Data* de forma a capacitar os recursos necessários para detecção. Esta plataforma é apoiada por ferramentas como *Snort*, *SFlows*, *GrayLog*, *Apache Spark*, *HBase* e *libnDPI*, que fornece a detecção de anomalia de rede. Hogzilla também fornece visibilidade da rede. A arquitetura definida, procura utilizar componentes estáveis e conhecidos. Assim, de acordo com os desenvolvedores as diretrizes seguidas fornecem um desenvolvimento rápido, estável e com foco na implementação dos algoritmos de detecção, não despendendo tempo com recursos subsidiários. Demais, o ambiente criado visa processar uma grande quantidade de dados. Neste cenário, o Hogzilla IDS é suportado pelas seguintes partes:

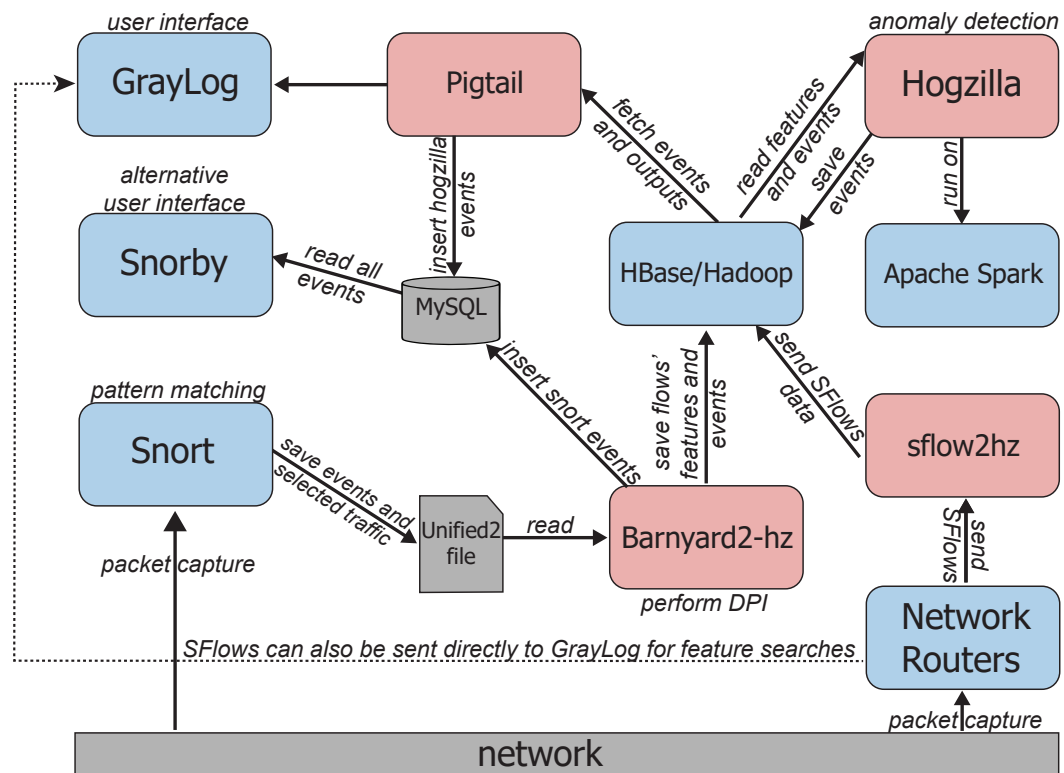
- a) *Apache Spark*: Utilizado para o processamento de rotinas desenvolvidas em Scala;
- b) *HBase/Hadoop*: Responsável pelo armazenamento de dados a seres processados e já processados;
- c) *Snort*: Coleta pacotes de rede, associados às vezes com os respectivos eventos gerados pelo *Snort* salvando-os em um arquivo unificado;
- d) *Barnyard2-hz*: Lê o arquivo unificado (pacotes e alguns eventos), executa o *Deep Packet Inspection* (DPI) e salva os recursos dos fluxos no *HBase*.
- e) *lib nDPI*: Usado para DPI e para identificar algum tráfego conhecido. Tais informações podem ser utilizadas em rotinas de detecção.
- f) *GrayLog*: Um “Gerenciamento de log de código aberto que realmente funciona”;
- g) *Snorby*: Um console de monitoramento alternativo. Recomenda-se o uso do *GrayLog*.
- h) *SFlow2Hz*: Um binário simples usado para inserir *sFlows* no *HBase*.

A Figura 3.1 apresenta a arquitetura estruturada para o desenvolvimento da ferramenta. De acordo com os desenvolvedores, tal arquitetura possui baixo acoplamento permitindo assim alterações no módulo sem muitos problemas. Sendo útil assim para manutenção das tecnologias atualizadas.

<sup>11</sup> Disponível em: <http://ids-hogzilla.org>



Figura 3.1: Arquitetura Hogzilla IDS



Fonte: Hogzilla IDS (2017).

Com o uso de tal estrutura, os autores afirmam que é possível a detecção de ataques como: *Spammers*, *Malware*, Vazamento de dados, Comunicações *Peer-to-Peer* (P2P), *DDoS*, Servidores hackeados, Varredura de portas (interna, externa, horizontal ou vertical) e outros mais.

De forma a lidar com o crescimento exorbitante da geração de dados, causado pelo aumento do número de usuários Rathore et al. (2016) propõe em seu trabalho um Sistema de Detecção de Intrusão de alta velocidade, capaz de executar em um ambiente *Big Data*. O design do sistema proposto contém quatro camadas, consistindo em camada de captura, filtração e camada de balanceamento de carga, camada de processamento e a camada de tomada de decisão. O IDS proposto foi implementado utilizando o paradigma de programação *MapReduce*<sup>12</sup> usando um único nó Hadoop, que processa o arquivo de sequência e calcula os valores dos parâmetros. A contribuição deste trabalho consiste em uma arquitetura baseada no Hadoop como proposta para um IDS capaz de detectar quaisquer tipos de ameaças de forma precisa, efici-

<sup>12</sup> Modelo de programação e *framework* proposto pelo Google para suporte de computações paralelas em grandes conjuntos de dados em *clusters* de computadores.

ente e com alta velocidade utilizando classificadores de aprendizagem de máquina. Além disso, para realização de análises em tempo real, o Apache Spark é usado como ferramenta no ecossistema Hadoop. Então os valores das características são enviados para servidores de decisão. Estes servidores possuem implementações de diversos classificadores, tais como como J48<sup>13</sup>, REPTree<sup>14</sup>, SVM<sup>15</sup>, entre outros.

Para realização de análise sobre um tráfego de rede e possibilidades de intrusão neste, os autores fizeram uso do Darpa Dataset<sup>16</sup> (2000). Os atributos utilizados como parâmetros, envolvem: duração completa do fluxo ou sessão, protocolo utilizado, serviço (serviço particular que o host está utilizando), número de *roots* envolvidos, número de pacotes, taxa de pacotes/-segundo para um fluxo em particular, tamanho médio do tamanho dos pacotes, desvio padrão de tamanhos e a faixa dos tamanhos dos pacotes. Entre as várias abordagens de aprendizagem em máquina, o sistema proposto funciona bem no REPTree e J48 usando os recursos propostos. Diversas destas abordagens são utilizadas como forma de proporcionar uma distinção entre as intrusões e os fluxos normais. Após, os classificadores REPTree e J48 são selecionados para classificação de intrusão usando os recursos propostos com base em seus resultados de desempenho. Os resultados da avaliação do sistema e da comparação mostram que o sistema possui uma melhor eficiência e precisão, em comparação com os sistemas existentes, com uma taxa de verdadeiros positivos de 99,9% e menos de 0,001% de falsos positivos usando o algoritmo REPTree.

Para Seelammal et al. (2016), os ataques cibernéticos vem sofrendo um aumento, parte disto se dá na presença de atividades recentes ocorrentes no trecho monitorado e anteriormente desconhecidas, onde a taxa de detecção de intrusão é imprecisa e baixa. Assim, em todas as infraestruturas de segurança os Sistemas de Detecção de Intrusão de Rede cresceram em um componente padrão e separável. Por esse motivo, o autor propõe um novo modelo com base em dados volumosos para detectar ataques desconhecidos. A ideia central do trabalho é de analisar a atividade dos usuários de uma rede e classificar as ações entre usuários regulares e anomalias. Para a implementação os autores utilizaram o *Snort* como ferramenta de captura do comportamento online dos usuários da rede. Após a coleta do comportamento da rede, o conjunto de dados foi analisado com o *framework* Hadoop utilizando o algoritmo de

<sup>13</sup> Implementação de código aberto em Java do algoritmo C4.5 na ferramenta WEKA.

<sup>14</sup> Algoritmo de aprendizagem que utiliza da árvore de decisão rápida.

<sup>15</sup> Modelos de aprendizagem supervisionados com algoritmos de aprendizagem associados que analisam dados utilizados para análise de classificação e regressão.

<sup>16</sup> Mais informações em: <http://askcypert.org/node/10>

classificação C4.5<sup>17</sup>.

O desempenho do projeto foi verificada junto ao 99KDDcup<sup>18</sup> *Dataset*. Assim, os autores afirmam que a proposta melhora o desempenho do IDS e sua capacidade de fornecer respostas rápidas a vários tipos de ataques de rede. O trabalho proposto então por Seelammal et al. (2016), desenvolve um IDS com plataforma de dados Hadoop, para analisar grandes volumes de tráfego de rede. Neste os autores, acreditam que os grandes dados têm uma promessa considerável, onde há enormes e potenciais desafios que podem ser superados para reconhecer seu verdadeiro potencial.

Em Sharma et al. (2016) analisam-se técnicas de aprendizado de máquina para detecção de intrusão em sistemas. Existem diversos algoritmos que se pode optar de acordo com a necessidade requerida pelo sistema a ser implantado. Neste trabalho, os autores fazem uso do classificador *Naïve Bayes* e *K-Nearest Neighbor*<sup>19</sup> na estrutura MapReduce. A implementação destes algoritmos foi realizada em um cluster de múltiplos nós Hadoop 2.6.0, com um nó mestre e oito nós escravos. Para fins de teste, foi utilizado o *dataset* NSL-KDD<sup>20</sup>, o qual de acordo com os autores não atende aos critérios de volume *Big Data*, porém propicia dados satisfatórios. Para análise do desempenho do sistema proposto, os autores realizam comparações do uso de implementações dos algoritmos citados utilizando o paradigma MapReduce com as implementações WEKA.

Os estudos de casos realizados por Sharma et al. (2016), analisam características como acurácia, precisão, sensibilidade, taxa de falsos positivos e a métrica F1 (responsável por determinar o equilíbrio entre precisão e revocação). Assim, os autores afirmam que tal experimento realizado demonstra que a plataforma MapReduce é mais rápida do que o WEKA e as mudanças feitas nos algoritmos conduzem resultados mais eficientes para as métricas analisadas.

### 3.2 USO DE KDD NA MINERAÇÃO DE DADOS DE WEB LOGS

Em Tsai et al. (2014) é apresentada uma discussão acerca de dados de Internet das Coisas e mineração aplicada nos mesmos. Neste trabalho é apresentada não somente uma descrição sistemática a respeito de mineração de dados, mas também como aplicar tais técnicas à IoT. A arquitetura apresentada na Figura 3.2, representa a integração de Internet das Coisas com a Des-

<sup>17</sup> Algoritmo utilizado para criação de árvores de decisão

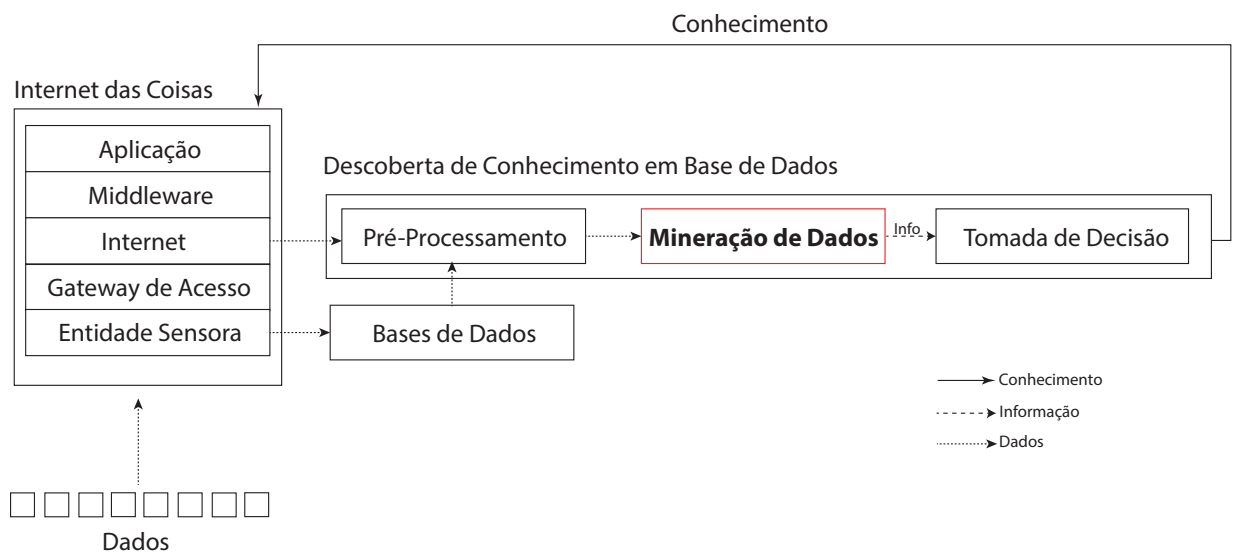
<sup>18</sup> Mais informações em: <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>

<sup>19</sup> Método não paramétrico utilizado para classificação e regressão.

<sup>20</sup> Mais informações em: <http://www.unb.ca/cic/datasets/nsl.html>

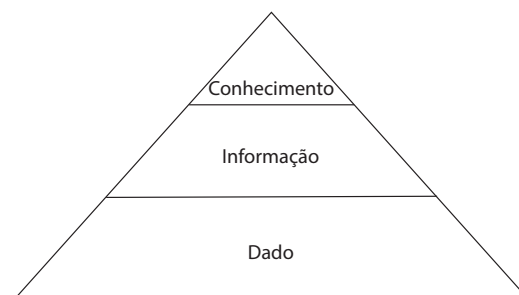
coberta de Conhecimento em Bases de Dados. Quando aplicado à IoT, as técnicas atreladas a KDD convertem os dados coletados de uma rede IoT em informações relevantes que podem ser transformadas em conhecimento, como representado na hierarquia apresentada na Figura 3.3. É importante ressaltar que todas as etapas do processo de KDD podem ter uma forte influência nos resultados obtidos na mineração. Também é importante observar que a fusão dos dados, o volume processado, transmissão e a computação descentralizada podem vir a apresentar maior impacto no desempenho do sistema e na qualidade do serviço IoT, do que KDD ou algoritmos de mineração de dados podem ter em aplicações tradicionais.

Figura 3.2: Arquitetura de IoT com KDD



Fonte: Tsai et al. (2014).

Figura 3.3: Hierarquia entre Dado, Informação e Conhecimento.



Fonte: Goldschmidt, Passos e Bezerra (2015).

De modo geral, tanto as etapas de KDD de pré-processamento, quanto as tecnologias de mineração necessitam ser remodeladas ao se trabalhar com IoT dada a produção de grandes quantias de dados. De outra forma, as tecnologias de mineração de dados atuais só poderiam ser aplicadas a sistemas IoT de pequena escala, não podendo gerar nada além poucas quantidades de dados. Para o desenvolvimento de um módulo de Mineração de Dados de alto desempenho de KDD para IoT, o autor ressalta três questões que devem ser consideradas, sendo estas:

- a) **Objetivo:** Premissas, limitações e métricas do problema precisam ser bem definidas inicialmente para determinar com maior precisão o problema a ser resolvido.
- b) **Dados:** Características dos dados, tais como tamanho, distribuição e representação precisam ser levadas em consideração, pois dados diferentes usualmente requisitam processamento e análises distintas.
- c) **Algoritmo de Mineração:** Com os objetivos e os dados claramente definidos previamente, o algoritmo de mineração dos dados pode ser determinado com maior clareza.

Em concordância com Uckelmann et al. (2011), Tsai et al. (2014), afirma que *clustering* é um método eficiente de aprimorar o desempenho do IoT na integração de identificação, detecção e atuação. Sendo esta a razão pela qual diversos algoritmos novos de cluster são desenvolvidos para as Redes de Sensores Sem Fio (WSNs) que são dispositivos mais comuns que se encontram no IoT. O trabalho ainda apresenta outras abordagens de mineração que podem ser utilizadas, para fins de mineração dos dados, tais como: classificação, regras de associação e reconhecimento de padrões. Podendo cada uma ser aplicada quando:

- a) **Clusterização:** O objetivo é classificar padrões não rotulados.
- b) **Classificação:** A finalidade é classificar padrões, onde apenas alguns encontram-se rotulados.
- c) **Regras de Associação:** A meta é encontrar eventos a partir dos padrões de entrada que ocorrem sem nenhuma ordem específica
- d) **Reconhecimento de Padrões:** O propósito é encontrar eventos a partir dos padrões de entrada que ocorrem em alguma ordem particular

O registro de ações desenvolvidas em uma rede (por exemplo, *syslog*<sup>21</sup>) apresenta valor estimado para a detecção de comportamentos inesperados ou anômalos em uma rede de grande escala. Entretanto, a identificação de falhas e suas causas são uma tarefa árdua quando ligados a uma grande quantidade de dados de *log* de um sistema em operação diária. Em Kobayashi et al. (2017) realiza-se um estudo, no qual é proposto um método de extração de falhas e suas causas a partir dos dados do *syslog* de rede.

A principal ideia do método baseia-se na inferência causal que reconstrói a consequencialidade dos eventos de rede de um conjunto através de séries de fenômenos temporais. Tal inferência permite a redução do número de ocorrências correlacionados por acaso, ou seja, produz eventos causais com maior probabilidade de ocorrência que uma abordagem tradicional baseada em correlação cruzada. Para este método, os autores sugerem o uso de um algoritmo por estes desenvolvido, chamado PC. Este algoritmo, estima a existência de grafos acíclicos dirigidos (Directed Acyclic Graph – DAG) a partir de bases de dados estatísticos baseados nas dependências condicionais.

O método foi aplicado a um conjunto de dados, o qual reunia informações de *syslog* de rede obtidos em uma rede acadêmica nacional no Japão. Assim, esta abordagem reduz significativamente o número de eventos pseudo-correlacionados em comparação com o uso de técnicas tradicionais. Além disso, através de dois estudos de caso realizados e comparação com dados de ingresso de problemas, foi demonstrada a eficácia do método abordado de operação de rede.

Para Rønning (2017), durante as últimas duas décadas ataques DDoS foram ameaças consideráveis para a infraestrutura da internet. Assim, mitigar estes ataques é uma tarefa particularmente desafiadora. Técnicas de detecção baseadas em assinaturas comuns são ineficientes em minar ataques DDoS, pois esse tipo de ataque possui a capacidade de se mascarar entre o tráfego legítimo. Desta forma em seu trabalho a autora apresenta um paradigma para combater ataques DDoS na vítima tida como alvo, usando elementos de Mineração de Dados e Aprendizado de Máquina. Estes dois métodos concentram-se na identificação de estruturas de dados ocultos no tráfego histórico são propostos, para diferenciar o tráfego legítimo do tráfego anômalo.

Em um primeiro momento recorre-se a técnicas de DM para encontrar regras de associação que sejam capazes de descrever a parte do tráfego que tem maior probabilidade de

---

<sup>21</sup> Disponível em: <https://tools.ietf.org/html/rfc5424>

reincidência. Como uma forma de dados para armazenar tais regras conduzidas por dados, foi empregada uma estrutura de árvore binária. O segundo método baseia-se em áreas anteriormente inexploradas dentro de técnicas de mitigação, onde técnicas de agrupamento são usadas para criar *clusters* geográficos. Para sintetizar as informações de agrupamento para cenários de filtragem de tráfego da vida real, utiliza-se o conceito de filtros de floração. De forma a garantir uma ótima solução e ambiente de teste, as abordagens propostas por Rønning (2017) são testadas em diferentes conjuntos de dados. Assim, os conjuntos de dados aplicados são obtidos a partir de *logs* do servidor Web na Noruega, onde os últimos oito bits foram anonimizados. Os resultados mostram que essas abordagens de mitigação melhoram a capacidade de separar anormalidades desconhecidas no conjunto de dados e a estrutura de tráfego legítima. Deste modo, a autora afirma que o esquema de filtragem DDoS proposto é capaz de suavizar 99% do tráfego de *botnet* e assim contrapor de modo significativo a magnitude destes ataques.

### 3.3 MINERAÇÃO DE DADOS PROVIDOS POR DISPOSITIVOS IOT

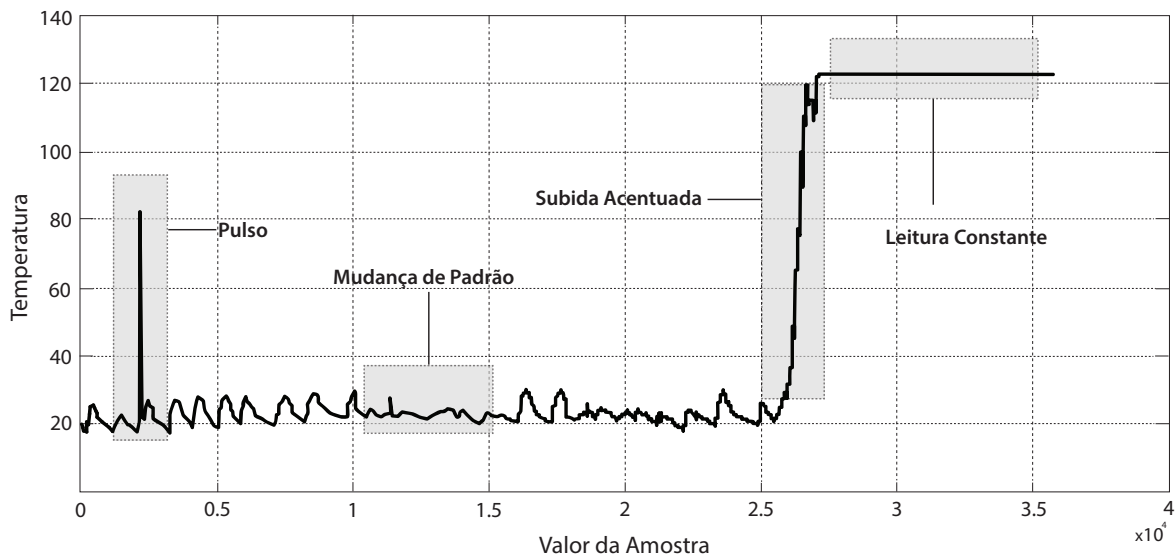
Por se tratar de uma rede vulnerável a ataques finórios em consequência à abertura de implantação, recursos limitados, heterogeneidade, mecanismos distribuídos, a implantação de metodologias convencionais de detecção de intrusão torna-se ainda mais laboriosa. De forma a propor uma solução para tal, Fu et al. (2011) sugere em seu trabalho um esquema de detecção de intrusão baseado na mineração de anomalias. O autor especifica uma arquitetura IoT com três camadas, sendo elas as camadas de percepção, rede e aplicação. Neste artigo é definido um esquema baseado em um modelo hierárquico de rede considerando restrições e peculiaridades de IoT para detecção de anormalidades ocorridas na camada de percepção. Considerando que cada objeto IoT possua um terminal que submeta dados brutos em direção a um ponto de coleta centralizado para análise posterior, denomina-se esta como camada de percepção. O esquema proposto por Fu et al. (2011) propõe a abordagem do problema em dois estágios:

- a) Utilização de algoritmos de mineração para detecção de anomalias na camada de percepção;
- b) Utilização de esquema de intrusão distribuída para distinguir entre anomalias e intrusão maliciosa no nível do segmento de rede, mas não como um todo, ou seja, não usando um esquema de detecção de intrusão centralizado.

Para realizar o desenvolvimento do algoritmo de mineração proposto com a finalidade de su-

prir as necessidades do esquema desenvolvido para detectar dados de anomalia da camada de percepção, os autores analisaram o comportamento padrão dos dispositivos de IoT em relação ao tempo, sugerindo cinco características comportamentais principais: pulso, subida acentuada, descida acentuada, leitura constante e mudança de padrão, que classificam um comportamento como anômalo. Tais características são exemplificadas na Figura 3.4.

Figura 3.4: Exemplos de Anomalias



Fonte: Fu et al. (2011).

Desta forma, possuindo padrões de tráfego regular e os tipos de anomalia detectáveis, torna-se possível detectar diferentes tipos de anomalias. Em um segundo momento, Fu et al. (2011) estabelece um esquema de detecção de intrusão distribuída projetado com base nas anomalias detectadas. Como nem todas as anomalias são desencadeadas por intrusão maliciosa, a intrusão semântica é analisada para distinguir comportamentos de intrusão de anomalias.

Os autores concluem que o método proposto pode ser perfeitamente implantado em ambientes heterogêneos, sendo flexível e extensível. A partir de uma análise teórica garantem que tal método requer uma menor quantidade de recursos, tornando-se assim mais adequado para a camada de percepção da Internet das Coisas. O experimento realizado demonstra que o método de detecção é auto-adaptativo e pode garantir uma baixa emissão de falsos positivos.

Já em Alghuried (2017), é proposta uma abordagem para detecção de anomalias em dispositivos IoT, através da combinação de dois algoritmos robustos de aprendizado de máquina. Os algoritmos trabalhados neste método são o *Inverse Weight Clustering* (IWC) e *C4.5*. O algo-



ritmo IWC apresenta uma versão aprimorada do algoritmo *k-means*, o qual é capaz de agrupar efetivamente dados baseado nas similaridades existentes nestes. Já o *C4.5*, é um algoritmo de árvore de decisão o qual pode ser utilizado para a classificação dos dados. O objetivo do modelo de detecção proposto é detectar anomalias em dados extraídos de dispositivos IoT na camada de aplicação.

A implementação do modelo proposto por Alghuried (2017) é dividida em três etapas, sendo elas: pré-processamento dos dados, agrupamento, classificação e testes. Onde na primeira, é realizada a qualificação dos dados do conjunto de dados *Intel Lab Data IoT*<sup>22</sup>, que apresenta informações relativas a coleta de dados de cinquenta e quatro sensores dispostos nos laboratórios de pesquisa Intel Berkeley em 2004, para a entrada do processo de detecção de anomalias. Clusterização e classificação são subprocessos do processo geral de prospecção destas. O agrupamento segmenta a entrada de dados em dois grupos, a partir de suas similaridades de estrutura e características, utilizando o algoritmo de clusterização ponderada inversa (IWC). O seguinte subprocesso utiliza o algoritmo *C4.5*, para classificação dos dados a partir da construção de uma árvore de decisão onde os dados são classificados entre regulares e anômalos. De acordo com o autor tal proposta de modelo foi testada e avaliada, podendo-se concluir a partir dos resultados que este modelo pode ser considerado acurado para a detecção de anomalias de dados IoT.

Em Butun et al. (2015), é apresentada a relação existente entre IoT e Computação em Nuvem, visto que em alguns casos os dados coletados de dispositivos IoT geralmente enviados a Nuvem, fato que de acordo com o autor pode expor as informações a uma variedade de ataques se os problemas de segurança e privacidade não forem devidamente tratados. Os autores analisam os desafios atuais de segurança em IoT. Os autores fornecem uma pesquisa de detecção de anomalia na IoT e resumem os resultados da seguinte forma:

a) **Abordagens Distribuídas:** Nesta abordagem, os dispositivos de modo colaborativo buscam resolver o problema da detecção de anomalias. Os esquemas de detecção de anomalia baseados em regras são amplamente utilizados nesta abordagem. A detecção de anomalia baseada em regras inclui prospecção cooperativa, busca estatística por anomalias e máquinas de vetor de suporte. As principais vantagens desta abordagem são a velocidade e escalabilidade.

b) **Abordagens Centralizadas:** Em comparação com abordagens distribuídas, estas são apre-

<sup>22</sup> Mais informações em: <http://db.csail.mit.edu/labdata/labdata.html>

sentam implementações mais simplificadas e requerem menor poder de computação por parte dos dispositivos IoT. Os algoritmos de ranking baseados em reputação e heurística são abordagens comuns usadas na detecção de nós mal-intencionados na WSN de rede de sensores sem fio.

- c) Abordagens Hierárquicas: Sendo a Internet uma rede hierárquica, o autor considera esta como sendo a abordagem de aplicação mais viável em uma grande rede. Agrupamento, detecção estatística de anomalias e modelo de Markov são técnicas comuns utilizadas na detecção de anomalias de IoT.
- d) Abordagens de detecção independentes: Esta abordagem foca em tornar cada nó ou dispositivo IoT capaz de detectar ataques através da manutenção de um histórico estatístico dinâmico de curto prazo do evento.

Tais abordagens estudadas por Butun et al. (2015) concentram-se na segurança de dispositivos IoT do ponto de vista da rede. Deste modo, podem ser utilizados para detectar ataques baseados em rede direcionados a IoT, como nós falsos, endereços de *Media Access Control* (MAC) ilusórios, falso roteamento, DoS e DDoS.

Em seu trabalho, Preuss et al. (2017) propõe uma arquitetura de identificação de anomalias em redes IoT utilizando registros de *logs*. Para isto, o autor faz uso de ferramentas tais como Apache Hadoop, Apache Flume e Apache Spark. Considerando-se a heterogeneidade das redes de Internet das Coisas, o Apache Flume é utilizado de forma a realizar a coleta de fontes diversas. Dessa forma, em um cenário onde existam aplicações heterogêneas conectadas a um mesmo *gateway*, é possível criar e configurar agentes individuais para cada aplicação e seus respectivos dispositivos. De acordo com os preceitos de Zhu et al. (2010), o autor define *gateways* IoT, como sendo elementos responsáveis por tornar possível a interação entre elementos de redes e tecnologias heterogêneas. Assim, uma vez que os *gateway* concentram o fluxo de dados da rede em um ponto único, o posicionamento de agentes Flume nestes locais, garante acesso aos *logs* de eventos de todos os dispositivos interligados neste *gateway*. Após a obtenção dos dados, estes são encaminhados para ao Hadoop HDFS e ao Spark, onde serão armazenados e processados respectivamente. Posterior a conclusão de tais estágios, tem-se a visualização dos dados, onde são apresentados alertas e demais tipos de informações, pertinentes ao administrador da rede responsável.

Preuss et al. (2017) ressalta porém, que para cada aplicação monitorada pela arquite-

tura proposta, deve-se existir um conjunto de treinamento que contenha dados normalizados de comportamento tido como normal para aquela aplicação. Desta forma, baseado em tal conjunto o monitoramento fica responsável pela comparação dos dados provenientes de um fluxo e classifica-los como anômalo ou regular. A análise executada, segue as características definidas por Fu et al. (2011). Assim, a partir das características da arquitetura proposta, enfatizando-se a capacidade de coleta e análise de grandes volumes de dados heterogêneos, o autor acredita que a proposta seja capaz de realizar a análise de possíveis falhas em tempo de execução ou tão próximo quanto seja possível. Desta forma, redes IoT com sistemas que requerem alta tolerância a falhas, podem reduzir o tempo necessário para a identificação de falhas, bem como na prevenção das mesmas.

### 3.4 CONSIDERAÇÕES

O estudo e desenvolvimento de propostas que visem abranger diferentes abordagens para detecção de anomalias em tráfegos de rede vem sendo amplamente trabalhado na área de segurança. A eclosão de pesquisas na área se deve ao fato do crescimento do volume de dados trafegados e do valor das informações ali contidas. É possível encontrar de modo amplo na literatura, diversos métodos e algoritmos propostos que podem ser utilizados para o desenvolvimento Sistemas de Detecção de Intrusão. Dentre os trabalhos correlatos aqui apresentados, tem-se o uso de diferentes técnicas e ferramentas para o processamento e análise de conteúdos que remetam a segurança da informação contida em registros de rede.

O uso de tecnologias de *Big Data*, tem se mostrado eficaz quanto ao suporte à execução de técnicas de Mineração de Dados. Desta forma, estudos voltados a esta área crescem diretamente ligados a segurança dos dados manipulados, buscando assim analisar anomalias que podem vir a serem geradas na rede em que estes se situam. Após uma análise do *overview* apresentado neste Capítulo sobre trabalhos desenvolvidos com relação à detecção de anomalia de modo geral e detecção de anomalias em IoT, é possível observar o uso de métodos estatísticos de Mineração de Dados e Aprendizado de Máquina na detecção de anomalias em ambos os tipos de rede (redes comuns e redes IoT). Utilizam-se ainda em alguns casos uma combinação de técnicas de agrupamento e classificação de modo a encontrar anomalias. São utilizadas técnicas para agrupar os dados dos rótulos como na aprendizagem supervisionada ou em dados não marcados. O agrupamento é seguido por uma estratégia de classificação para identificar dados anômalos e regulares. Tais atividades ocorrem durante a fase de treinamento após o teste

baseado em determinadas regras previamente estipuladas. Para validação, o modelo de detecção de anomalias é testado usando dados diferentes daquele usado na fase de treinamento, e os resultados são avaliados.

A projeção de esquemas de detecção de anomalia para redes IoT possui características comuns a abordagens de detecção em outros dispositivos. Todavia, para estes é necessário considerar os desafios únicos apresentados. Esquemas de detecção intrusiva que requerem alto poder de computação ou consumo de memória significativo não são adequados à IoT. Além disso, os dispositivos IoT são dissemelhantes e geram dados heterogêneos enquadrando-se no padrão de *Big Data*. Com tal característica, as abordagens de verificação de anomalias em IoT não devem limitar-se a busca de ataques de rede, examinando também irregularidades em dados ou mesmo em cargas úteis transportadas no tráfego de rede.

Em complemento as propostas apresentadas nos estudos aqui mencionados, o presente trabalho tem como objetivo o desenvolvimento de uma arquitetura que faça uso de uma infraestrutura de *Big Data* (Apache Spark) capaz de realizar processamento de dados em larga escala de modo eficiente, com o auxílio de bibliotecas (MLlib<sup>23</sup>) aptas a proporcionar suporte a técnicas de Descoberta de Conhecimento em Bases de Dados. Para experimentos e validação da proposta realizada é utilizado um conjunto de dados que dispõe de registros de tráfego de rede em sensores e atuadores em operação em um sistema de tratamento de água. Tais informações representam um ambiente composto por dispositivos de Internet das Coisas, sendo passível de realização de um estudo de caso para aplicação em tais esferas.

---

<sup>23</sup> Biblioteca do Spark utilizada para aprendizado de máquina.

## 4 PROPOSTA

Este Capítulo apresenta uma proposta de sistema para detecção intrusão baseado em anomalias a partir da utilização de uma infraestrutura de *Big Data* para realização do processamento. Descreve-se nesta Seção o desenvolvimento do trabalho e da proposta de utilização de técnicas de Mineração de Dados para elaboração de um sistema de detecção de intrusão capaz de realizar o processamento de dados gerados por dispositivos de Internet das Coisas. Este Capítulo encontra-se organizado da seguinte maneira: a Seção 4.1.1 apresenta a arquitetura proposta para o desenvolvimento do presente trabalho; em seguida a Seção 4.1.2 apresenta a descrição do uso de *Big Data* como base à Mineração de Dados; na Seção 4.1.3 descreve-se a utilização de algoritmos em Python como técnicas voltadas a Mineração de Dados; e por fim a Seção 4.2 destina-se a descrição dos algoritmos de mineração utilizados.

### 4.1 ARQUITETURA

A seguir, são descritos os módulos da arquitetura proposta para viabilizar o uso de *Big Data* como suporte a Mineração de Dados na detecção de anomalias em tráfegos de rede.

#### 4.1.1 Arquitetura

A estruturação da proposta apresenta-se dividida em dois módulos principais: Coleta e Análise e Processamento dos Dados, como ilustrado na Figura 4.1. A infraestrutura aqui seguida busca reunir e implementar as principais características de uso de *Big Data*, utilizando assim ferramentas de Mineração de Dados para detecção de anomalias em uma rede de dispositivos IoT. A Figura 4.2 apresenta a forma na qual estrutura-se a arquitetura utilizada, fazendo uso de uma solução *Big Data*. Com o uso de tal infraestrutura, tem-se então o objetivo mensurar o proveito de uso na detecção de intrusão e sua viabilidade de aplicação em ambientes de larga escala, analisando então o uso de uma ferramenta de *Big Data* neste contexto.

Como mostrado na Figura 4.2, os dados são coletados de dispositivos de Internet das Coisas e redirecionados a um *gateway*, onde reúnem-se as informações coletadas. Neste trabalho, entretanto, simula-se a coleta de dados e a geração de fluxos destes para o Spark através de um conjunto de instruções programadas em *Shell Script*<sup>24</sup>. Neste último, realiza-se então o

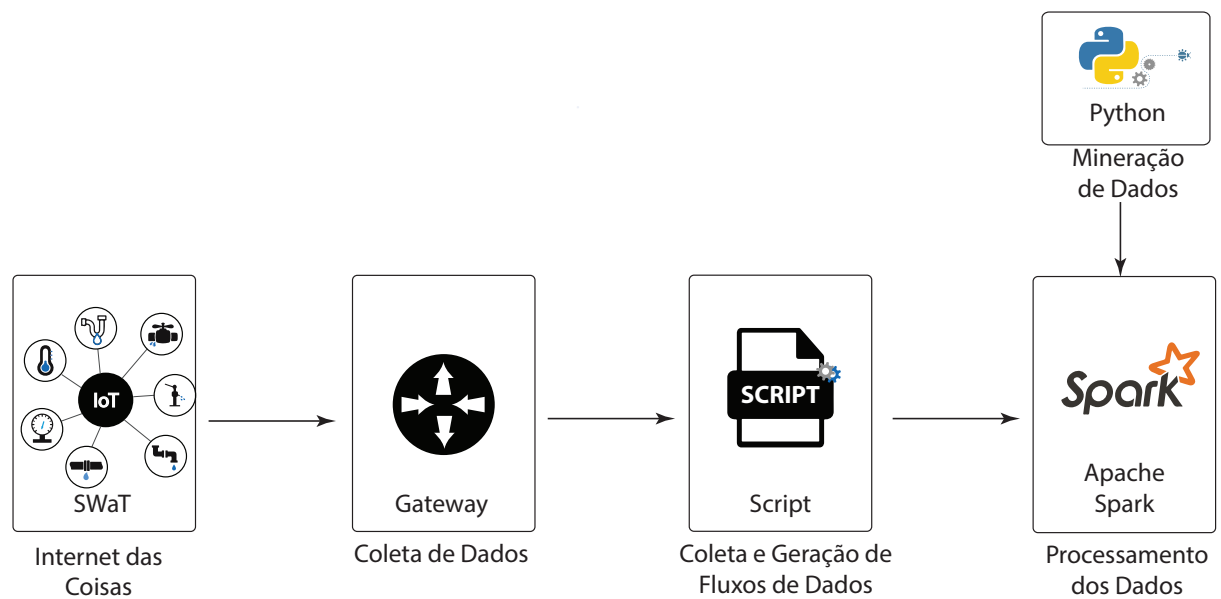
<sup>24</sup> Mais informações em: <https://searchdatacenter.techtarget.com/definition/shell-script>

Figura 4.1: Estrutura



Fonte: Acervo Pessoal.

Figura 4.2: Arquitetura Utilizada



Fonte: Acervo Pessoal.

processamento das informações recebidas. Algoritmos desenvolvidos em Python tem seu uso acoplado ao Apache Spark, assim durante o processamento das informações aplicam-se algoritmos de Mineração de Dados desenvolvidos nesta linguagem a fim de averiguar a existência de anomalias na rede.

#### 4.1.2 Utilização de Infraestrutura Big Data como Suporte à Mineração de Dados

Como demonstrado no Capítulo 3, a utilização de algoritmos de Mineração de Dados como meio de detecção de anomalias em redes se obteve grande enfoque quando trata-se da segurança. Tal receio torna-se ainda mais intenso ao tratar de redes IoT, devido ao seu exorbitante crescimento e poucas padronizações tornam-se mais suscetíveis a ataques. Desta forma, propõe-se o uso de uma infraestrutura de *Big Data* de modo que esta provenha o *background* fundamental para realização do processamento necessário para o monitoramento e preservação de uma rede.

Focado no processamento de dados estruturados e não estruturados, bem como nas correlações e descobertas que desse processamento podem advir, soluções *Big Data* são vantajosas pois não limitam-se apenas a um segmento, podendo ser aplicado nas mais distintas áreas. O uso dessas tecnologias propiciam acesso a dados importantes, tornando-se um repositório exclusivo e ilimitado de elementos. No entanto, é necessário observar características tais como a confiabilidade da fonte de informações, superficialidade do conteúdo disponível e insuficiência de dados, pois estas podem prover análises enganosas, inferindo em um entendimento equivocado. Ressalta-se também a complexidade no gerenciamento de informações ao se trabalhar com tais volumes de dados. Apesar de tal dificuldade, deve-se ainda ter em vista a vasta quantidade de informações a serem obtidas ao manipulá-los, logo, soluções de *Big Data* versam sobre dados brutos até transformá-los em discernimentos valiosos para as tomadas de decisão.

A utilização de ferramentas de *Big Data*, para realização do processamento e análise dos dados obtidos através do registro de informações de tráfego de rede propicia vantagens quando confrontado com o volume de informações a serem averiguadas. Neste contexto, técnicas de Mineração de Dados são utilizadas de modo a extrair dados relevantes, seja para o administrador de rede ou para o usuário final. Neste âmbito, a conversão de tais dados em conhecimento pode vir a destacar questões importantes relativas a segurança do sistema observado. Todavia, pesquisas como Peer Reserach Big Data Analytics (2013) apontam que com o uso de poder computacional tradicional, ou seja, de um computador usual, o desempenho obtido na análise de grandes bases de dados não é satisfatório. A adequação de uma infraestrutura de *Big Data* para suportar a Mineração de Dados fornece o aporte exigido para um processamento apropriado de informações em um tempo capaz de agir de forma reativa a ações indesejadas, ou mesmo indevidas.

A interação entre dispositivos IoT ocorre de forma pervasiva, ou seja, a tendência de expansão aumenta a cada dia. Este aumento ocorre devido a diversidade de aplicação destes e suas variedades. Por consequência, o número de informações trocadas e o registro de ações cresce, sendo necessário que a capacidade de processamento acompanhe tal demanda. Dada tamanha visibilidade, aumenta a possibilidade de ataque às informações armazenadas. Na existência de tais tentativas de usurpação de propriedades contidas na rede, por vezes geram-se alterações ao tráfego esperado desta. De forma a detectar tais investidas de acesso não autorizado, o uso de abordagens de descoberta de conhecimento em bases de dados, tais como a mineração, tornam-se concebíveis como formas de detecção. Acometimentos porém necessitam de ações responsivas imediatas, visando assim tratar e garantir a segurança de informações. Entretanto, com o acentuado fluxo de informações que é gerado por dispositivos de Internet das Coisas a realização de processamento por meio de computações tradicionais torna-se um desafio.

Com o emprego de ferramentas e aplicações devidas busca-se proporcionar o uso de uma infraestrutura basilar capaz de dotar a execução de DM como meio de prospecção às anomalias geradas. Por conseguinte diminuir o tempo de responsividade, visando aumentar a segurança de informações, resultando em maior confiabilidade do sistema. Desta forma, ao vincular a aplicação de algoritmos desenvolvidos em Python a recursos *Big Data* tal qual como demonstrado na Figura 4.2, o objetivo é prover uma base para melhor desenvoltura do processo de detecção de anomalias por esse analisado.

### 4.1.3 Mineração de Dados em Python

De acordo com Rossum (1995), criador da linguagem, Python provê o desenvolvimento de programação em alto nível, multi paradigma e interpretada . Esta é uma linguagem de *script* e de tipagem dinâmica, sendo desta forma de fácil aprendizado e manutenção. Foi projetada com a filosofia de enfatizar a importância do esforço do programador sobre o esforço computacional. Deste modo, valoriza-se a legibilidade do código não deixando de considerar velocidade ou expressividade. Python combina uma sintaxe concisa e clara com os recursos poderosos de bibliotecas padrão e por módulos e *frameworks* desenvolvidos por terceiros. Devido às suas características, Python costuma ser utilizado para processamento de textos, dados científicos, criação de Interfaces Comuns de Gateway (CGI) para páginas dinâmicas para a Web dentre diversas outras aplicações.

A linguagem possui embasamento para o uso diversas bibliotecas externas, capazes de



dar suporte a Ciência de Dados. Este conceito busca unificar áreas como Estatística, Análise de Dados e seus métodos relacionados, de modo a compreender e analisar fenômenos reais com o uso de dados. Assim, empregam-se técnicas e teorias extraídas de diversos campos dentro de áreas como Matemática, Estatística, Ciência da Informação e Ciência da Computação, em particular dos subdomínios de Aprendizado de Máquina, classificação, análise de *cluster*, Mineração de Dados, Bancos de Dados e visualização.

Em Nielsen (2015), enumeram-se razões para o uso de Python no ramo da Ciência de Dados, tais como:

- a) Simplicidade: Programadores consideram Python como uma linguagem clara e simples com uma alta legibilidade, mesmo para os não-programadores. A simplicidade existe tanto na própria sintaxe quanto no incentivo para escrever código claro e simples preva-  
lente entre os programadores Python.
- b) Independência de Plataforma: Python é executável nas três principais plataformas de computação desktop Mac, Linux e Windows, bem como em várias outras.
- c) Programa Interativo: Prompt interativo com o loop de leitura, avaliação e impressão (Read–Eval–Print Loop – REPL), que facilita a programação exploratória conveniente para muitas tarefas de Mineração de Dados, enquanto ainda pode-se desenvolver programas completos em um ciclo de edição-depuração.
- d) Linguagem de Uso Geral: Propósito geral que pode ser usado para uma variedade de tarefas além da Mineração de Dados, como por exemplo aplicativos de usuários, administração de sistema, jogos e etc.
- e) Qualidade: Baixa taxa de erros no seu desenvolvimento quando comparada as demais.

Assim, para o desenvolvimento da proposta deste trabalho propõe-se o uso de técnicas de Mineração de Dados em conjunto com ferramentas de *Big Data*. Para fins de estudos de desempenho em relação ao tempo do uso de infraestrutura *Big Data*, será realizada uma comparação com o WEKA, visto que este utiliza técnicas de Aprendizado de Máquina sem requerer uma base mais elaborada.

## 4.2 ALGORITMOS

A seguir, são descritos os algoritmos que tem sua execução proposta como teste na predição e detecção de anomalias no conjunto de dados estudado neste trabalho. Para as execuções realizadas no Apache Spark, a implementação utilizada baseou-se no exemplo disponibilizado na MLlib no site oficial do mesmo. Já no WEKA utilizou-se dos algoritmos oferecidos na versão 3.8.1 da ferramenta.

### 4.2.1 Naïve Bayes

Classificadores Bayesianos são classificadores estatísticos que rotulam um objeto numa determinada classe baseando-se na probabilidade deste enquadrar-se a esta condição, desta forma podem ser considerados uma abordagem interpretativa e analítica para o raciocínio probabilista que descrevem a independência condicional entre subconjuntos de variáveis. Na Aprendizagem de Máquina, classificadores Bayesianos ingênuos são um conjunto classificadores simples probabilísticos com base na aplicação do teorema de Bayes com pressupostos de independência entre os recursos. De acordo com Witten et al. (2016), *Naïve Bayes* é um exemplo raro de um algoritmo que não precisa de adaptação para lidar com fluxos de dados, desde que não haja mudanças substanciais no fluxo. O treinamento é incremental, ou seja, envolve unicamente a atualização de um conjunto fixo de parâmetros numéricos. Deste modo, o uso da memória é pequeno pois nenhuma estrutura é adicionada ao modelo.

Este algoritmo é dividido em duas etapas principais, sendo estas as etapas de aprendizado e classificação. No estágio de aprendizado do algoritmo, são calculadas as probabilidades iniciais de classificação para cada classe existente, ou seja, as chances de um atributo ser rotulado em cada classe. Após, o modelo é construído calculando as probabilidades de pertencimento a classe de acordo com a base de treinamento já especificada previamente. Neste processo, ocorre também uma suavização das probabilidades de forma a evitar cálculos errôneos. Já na etapa de classificação realiza-se o cálculo das probabilidades de pertencimento a cada classe, aplicando-se o modelo estimado previamente. Após o cálculo de tais exequibilidades, a entrada de dados é então classificada dentro das possibilidades existentes.

### 4.2.2 Random Forest Tree

O conceito de árvores de decisão é voltado principalmente para sistemas baseados em regras. Dado o conjunto de dados de treinamento com alvos e recursos, algoritmos da árvore de decisão apresentaram então um conjunto de regras. As mesmas regras de conjunto podem ser usadas para executar a previsão no conjunto de dados de teste.

Em Breiman (2001) define-se este algoritmo como uma combinação de preditores de árvores, de modo que cada árvore depende dos valores de um vetor aleatório amostrado independentemente e com a mesma distribuição para todas as árvores na floresta. Assim, *Random Forest Tree* é um algoritmo supervisionado para classificação, regressão e outras tarefas, que opera construindo florestas de árvores de decisão no tempo de treinamento e produzindo a classe (classificação) ou previsão média (regressão) das árvores individuais. Este, no lugar do uso do ganho de informação ou o índice gini (medida de desigualdade) para o cálculo do nó raiz, realiza o processo de encontrá-lo dividindo de modo aleatório os nós de recurso. De modo geral, quanto mais árvores na floresta, mais robusta ela se torna. Da mesma forma que no classificador de florestas aleatórias, quanto maior o número de árvores na floresta, maior a precisão dos resultados.

Este algoritmo pode ser utilizado tanto para a classificação como para a tarefas de regressão, sendo capaz ainda de trabalhar com atributos faltantes. O algoritmo *Random Forest* é um tipo de *ensemble learning*, ou seja, método que gera muitos classificadores e combina o seus resultados. Neste, são geradas diversas árvores de decisão, cada qual com suas particularidades e combina-se então o resultado da classificação de todas estas. Esta combinação de modelos, torna este um poderoso algoritmo de *Decision Tree*.

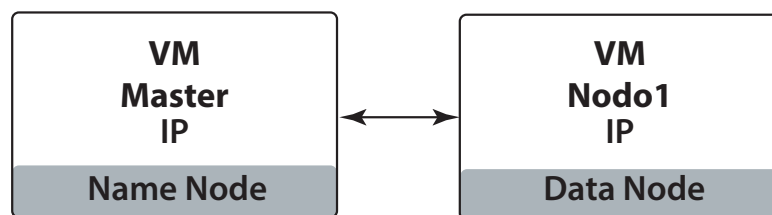
## 5 EXPERIMENTOS

Neste Capítulo são descritos os experimentos realizados a fim de validar a utilização de técnicas de mineração para detecção de anomalias em um ambiente *Big Data*. Desta forma, encontra-se organizado do seguinte modo: a Seção 5.1 apresenta uma descrição do cenário utilizado para fins de realização dos testes da implementação proposta; a Seção 5.2 concentra-se no detalhamento da base de dados utilizada como ambiente estudo; e por fim a Seção 5.3 apresenta uma breve discussão a cerca dos experimentos realizados.

### 5.1 AMBIENTE BASE

Para o desenvolvimento do presente trabalho, foi utilizado um computador com o sistema operacional Windows 10 Pro de 64 bits, com processador Intel Core i7-4500U de 1.80GHz, com base em x64. Para o desenvolvimento do cenário de utilização ao qual a arquitetura destina-se, foram utilizadas duas máquinas virtuais emuladas a partir do *software* VirtualBox<sup>25</sup>. Cada uma destas foi estruturada com 6 Gigabytes (GB) de memória RAM, 20 GB de armazenamento e utilização do sistema operacional GNU/Linux Ubuntu 16.04.2. Cada máquina destas, conta ainda com o uso de dois processadores. O cenário base é composto por duas máquinas virtuais interligadas por uma rede IP. Na Figura 5.1 apresenta-se a forma na qual o ambiente foi elaborado.

Figura 5.1: Máquinas Virtuais



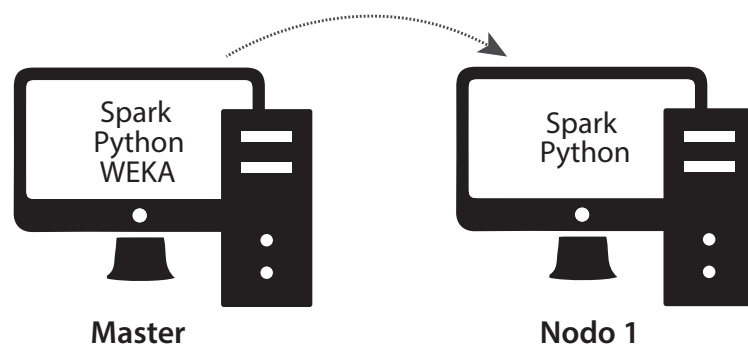
Fonte: Acervo Pessoal.

Dentro deste ambiente, foram instaladas as ferramentas necessárias para composição da arquitetura descrita no Capítulo anterior, ou seja, Apache Spark, WEKA e algoritmos de Mineração de Dados desenvolvidos em Python. Na Figura 5.2 apresenta-se a distribuição das

<sup>25</sup> Disponível em: <https://www.virtualbox.org/>

ferramentas dentro do ambiente de testes. Ao avaliar o Spark, a máquina que opera como *Master* é responsável pelo gerenciamento e submissão das tarefas de DM a serem executadas nos nodos auxiliares. O Nodo 1 (*Data Node*) atua então como um *worker* o qual é responsável pelo processamento e execução das incumbências recebidas. Nesta máquina executam-se os métodos de Mineração de Dados sobre os dados recebidos para computação. Para análise do WEKA, utilizou-se apenas da estrutura da máquina *Master*.

Figura 5.2: Cenário Base



Fonte: Acervo Pessoal.

## 5.2 BASE DE DADOS

Como contexto de IoT, adotou-se dados de um sistema industrial, cenário onde as vulnerabilidades da arquitetura IoT são mais exploradas. A detecção de intrusão para *Cyber Physical System* (CPS) passa por processos de avaliações precisas e de avaliação comparativa. Este fato se deve a falta de conjuntos de dados nesta área de pesquisa disponíveis para fins de estudo. Grande parte dos dados de detecção de intrusão atualmente disponíveis concentram-se apenas no tráfego de rede, sendo inadequado para a detecção de intrusão CPS, recorrendo-se assim à simulação por vezes inadequadas de dados. *Cyber Physical Systems* comumente são formados por redes industriais que trocam dados entre sensores, atuadores e controladores lógicos programáveis em tempo real. Tais redes costumam utilizar sistemas de controle e supervisão (Supervisory Control and Data Acquisition – SCADA) que envolvem *hardware* e *software*. Como os ataques cibernéticos podem fazer com que o comportamento do CPS se altere drasticamente. É necessário assim gerar um conjunto de dados que reflita a complexidade do sistema através evolução das intrusões. À vista disso Goh et al. (2016) provê acesso ao *Secure Water Treatment*

(SWaT) Dataset para fins de pesquisa.

O conjunto de dados SWaT foi gerado sistematicamente a partir do teste em uma estação de tratamento de água. Na Figura 5.3 apresenta-se uma visão geral dos processos testados e como o sistema encontra-se estruturado. Nesta, é possível perceber que a estação é composta de seis partes que constituem todo o processo de tratamento de água. Os dados coletados consistem em registros de onze dias de operação contínua. Sendo destes sete dias de dados de tráfego regular e quatro em cenários de ataque. Os autores afirmam que todos os dados do tráfego de rede, dos sensores e atuadores do sistema foram coletados durante os dias considerados.

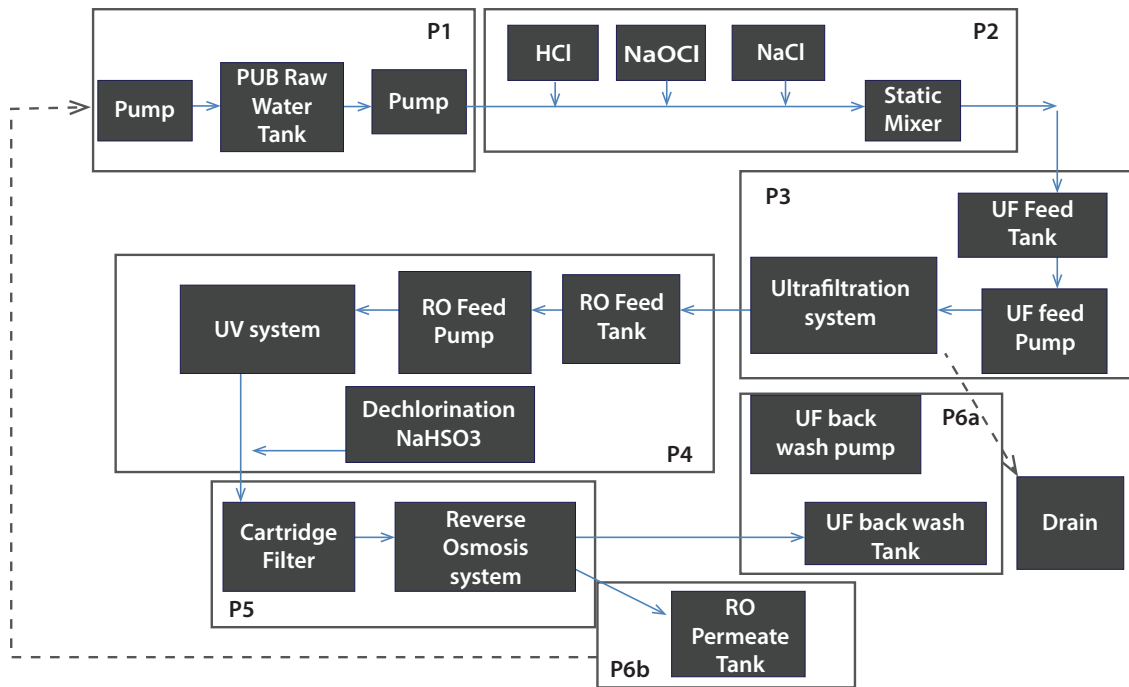
O objetivo deste conjunto de dados é auxiliar pesquisadores na concepção de soluções, testes, avaliação e fins de comparação para a pesquisa da CPS. Este conjunto apresenta-se segmentado em dois módulos, onde no primeiro descrevem-se as propriedades físicas capturadas de cada dispositivo em modo operação no período de coleta, e no segundo informações do tráfego de rede coletado usando equipamentos comercialmente disponíveis. Este *dataset* apresenta as seguintes características:

- a) Tráfego de Rede e Propriedades CPS: O conjunto de dados contém todo o tráfego de rede capturado durante esse período. Inclui ainda a relação completa dos valores obtidos de todos os cinquenta e um diferentes sensores e atuadores disponíveis no SWaT.
- b) Rotulado: Todos os dados adquiridos durante este processo são rotulados de acordo com comportamentos normais e anormais
- c) Cenários de Ataque: Os ataques gerados para este conjunto de dados foram derivados através dos modelos de ataque desenvolvidos pelos autores. Tais modelos consideram o espaço de intenção de um CPS como uma categoria de ataque. Durante o período de observação, em quatro dias foram lançados 36 ataques.

No total, 946.722 (novecentas e quarenta e seis mil setecentos e vinte e dois) amostras, compreendendo trinta atributos, foram coletadas ao longo de onze dias a partir de cinquenta e um dispositivos. Os atributos incluem:

- a) Leituras do sensor: Nível do tanque de água, medidor de fluxo, propriedades da água, pressão.
- b) Leituras do atuador: Bomba de água, válvulas motorizadas, De-clorador ultravioleta.

Figura 5.3: SWaT: Visão Geral dos Processos Testados



Fonte: Adaptado de Goh et al. (2016)

De forma a aplicar algoritmos de DM, foram separados em arquivos distintos os dados captados de cada dispositivo. Assim, estes documentos reúnem informações dos valores das amostras obtidas bem como a rotulação indicada pelos autores. Este último atributo serve como requisito de treinamento para os algoritmos a serem aplicados de modo a capacitá-los a realizar uma detecção apropriada.

Ao longo do período de estudo da base de dados foram realizados diferentes ataques, de forma a causar anomalias nos registros de rede. As investidas efetuadas foram direcionadas a diferentes tipos de sensores e atuadores do sistema. Estes acometimentos incluem ataques de polarização, de repetição, ataques de ponto único e de múltiplos pontos. Assim, estas investidas direcionadas aos componentes digitais do sistema, implicam em situações tais como o transbordamento dos tanques, estouro dos canos, entre outras. Dadas as falhas nas solicitações realizadas, decorre-se então um mau funcionamento da estação.

### 5.3 RESULTADOS E DISCUSSÃO

Tendo como base o cenário descrito na Seção 5.1, realizou-se a execução dos algoritmos *Naïve Bayes* e *Random Forest Tree* discutidos na Seção 4.2 sobre os dados de *log* fornecidos através do *Secure Water Treatment Dataset*. Os experimentos foram realizados utilizando os seguintes atributos encontrados no *dataset*: *orig*, *type*, *i/f\_name*, *i/f\_dir*, *src*, *dst*, *proto*, *appi\_name*, *proxy\_src\_ip*, *CIP\_Function\_Code*, *CIP\_Function\_Description*, *CIP\_Transaction\_ID*, *SCADA\_Tag*, *CIP\_Value*, *service*, *s\_port*. Tal como descrito em Goh et al. (2016), estes atributos correspondem respectivamente a descrição apresentada na Tabela 5.1. Estes dados foram então combinados através de informações de *timestamp* com dados previamente rotulados do mesmo *dataset*, gerando assim o atributo utilizado como alvo, a classificação entre ‘Ataque’ e ‘Normal’.

Tabela 5.1: Dados do Tráfego de Rede

<b>Categoria</b>	<b>Descrição</b>
Origin	IP do servidor
Type	Tipo de log
Interface Name	Tipo de interface de rede
Interface Direction	Direção de dados
Source IP	Endereço IP da fonte
Destination IP	Endereço IP do destino
Protocol	Protocolo de Rede
Proxy Source IP	Endereço de proxy da fonte
Application Name	Nome da aplicação
Modbus Function Code	Código da Função
Modbus Function Description	Descrição da Função Modbus
Modbus Transaction	ID da transação
SCADA Tag	ID do sensor ou do atuador
Modbus Value	Valor Modbus
Service/Destination Port	Número da porta do IP de Destino
Source Port	Número da porta do IP de origem

Fonte: Adaptado de Goh et al. (2016).

Através do uso de tais atributos, torna-se possível então detectar padrões de anomalias que envolvam alterações nestas propriedades. Estas alterações podem ser desde mudanças na fonte de recepção de dados, mudança do destino das mensagens, variações repentinas no protocolo utilizado, modificações em funções realizadas, fato que poderia vir a gerar alterações no código da descrição da função respectiva. Tais anomalias podem vir a ser então indícios de



ataques na rede.

A execução dos experimentos foi realizada em duas etapas, sendo estas:

a) *Naïve Bayes*

- Execução do algoritmo *Naïve Bayes* com a ferramenta WEKA
- Execução do algoritmo *Naïve Bayes* em um ambiente Spark apenas com a máquina *master*
- Execução do algoritmo *Naïve Bayes* em um ambiente Spark com a máquina *master* e outra atuando como *worker*

b) *Random Forest Tree*

- Execução do algoritmo *Random Forest Tree* com a ferramenta WEKA
- Execução do algoritmo *Random Forest Tree* em um ambiente Spark
- Execução do algoritmo *Random Forest Tree* em um ambiente Spark com a máquina *master* e outra atuando como *worker*

Para cada instância da execução foram utilizadas as 946.722 (novecentas e quarenta e seis mil setecentos e vinte e dois) amostras, sendo 25% dedicado ao treinamento dos algoritmos em questão, e 75% para aplicação. De modo a particionar o *dataset* em diferentes fluxos, foi elaborado um *script* com o propósito de agrupar em arquivos distintos as informações de cada dispositivo baseado no atributo *SCADA Tag*. A partir deste roteiro de instruções foram então gerados cinco fluxos distintos. Na Tabela 5.2 são apresentadas as informações dos respectivos tamanhos de cada fluxo gerado. Mesmo com um volume moderado de dados quando comparados a demais propriedades *Big Data*, este conjunto propicia conteúdos satisfatórios.

Tabela 5.2: Fluxos de Dados

<b>Fluxo</b>	<b>Tamanho</b>
HMI_AIT202	2, 159 GB
HMI_FIT201	3, 244 GB
HMI_LIT101	2, 529 GB
HMI_LIT301	2, 059 GB
HMI_LIT401	2, 123 GB

Fonte: Acervo Pessoal.

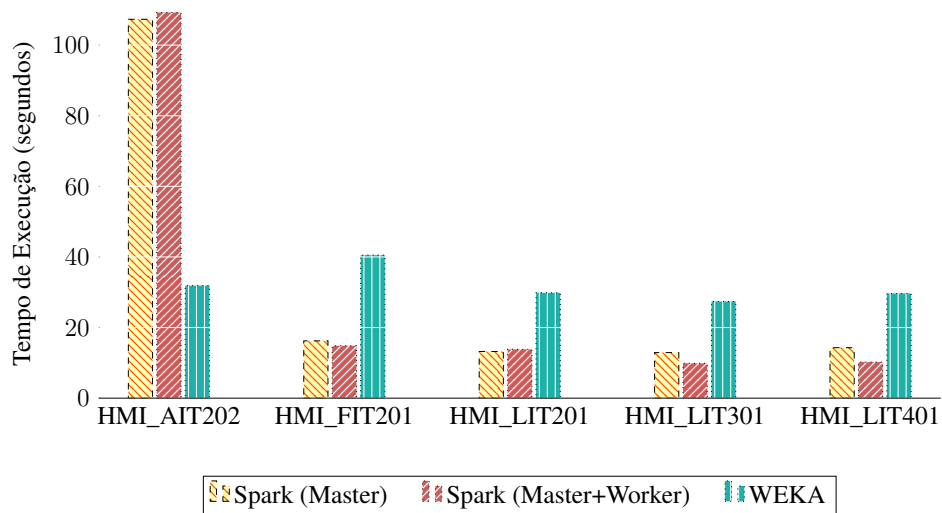
Os resultados obtidos a partir da execução dos fluxos de dados nas três situações as quais foram submetidos são descritos nas Figuras 5.4 e 5.5. Estes gráficos foram gerados a partir dos dados contidos nas Tabelas 5.3 e 5.4 respectivamente, denotados em segundos.

Tabela 5.3: Tempos de Execução do Algoritmo Naïve Bayes (segundos)

Fluxo	Spark (Master)	Spark (Master + Worker)	WEKA
HMI_AIT202	109,32	109,24	31,87
HMI_FIT201	16,23	14,83	40,51
HMI_LIT101	13,22	13,82	29,84
HMI_LIT301	12,90	9,87	27,29
HMI_LIT401	14,28	10,21	29,65
Total	163,95	157,97	159,26

Fonte: Acervo Pessoal.

Figura 5.4: Tempos de Execução do Algoritmo *Naïve Bayes*

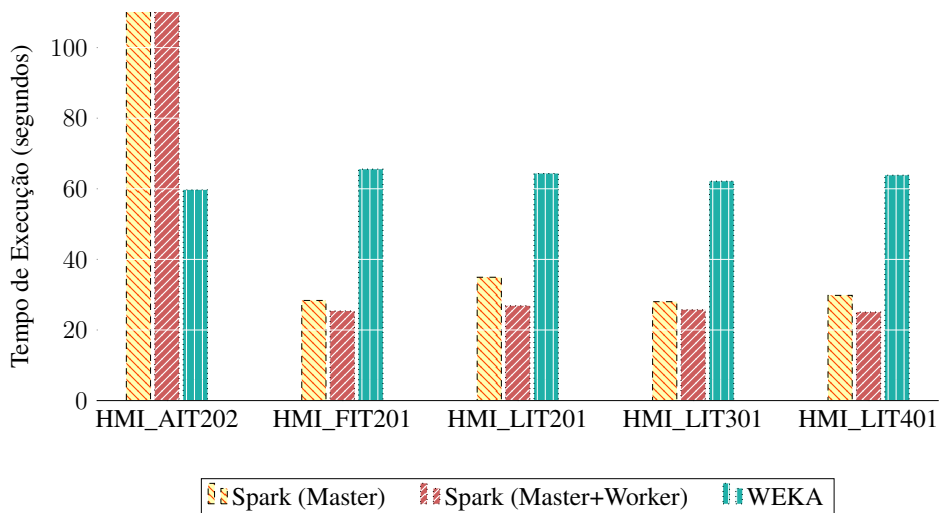


Fonte: Acervo Pessoal.

Tabela 5.4: Tempos de Execução do Algoritmo Random Forest Tree (segundos)

Fluxo	Spark (Master)	Spark (Master + Worker)	WEKA
HMI_AIT202	129,326	128,42	59,71
HMI_FIT201	28,37	25,37	65,55
HMI_LIT101	34,94	26,84	64,32
HMI_LIT301	28,02	25,71	62,14
HMI_LIT401	29,81	25,06	63,83
Total	250,79	231,40	315,55

Fonte: Acervo Pessoal.

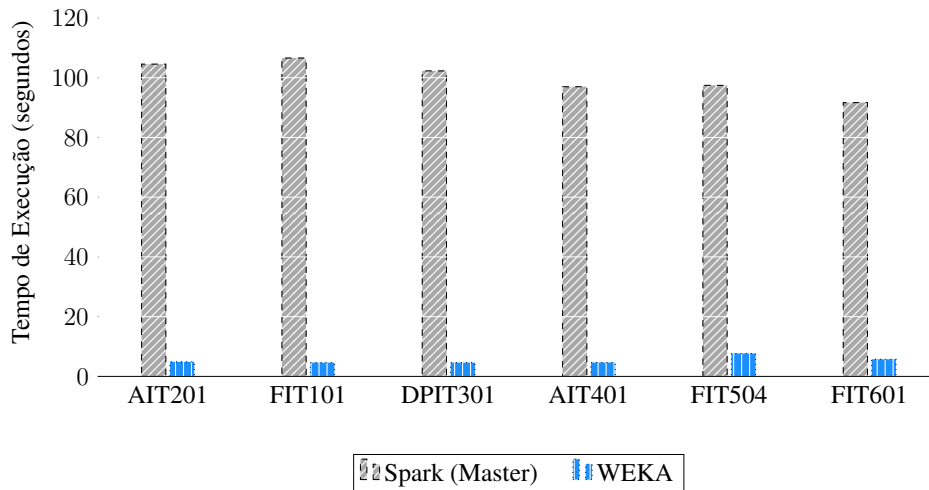
Figura 5.5: Tempos de Execução do Algoritmo *Random Forest Tree*

Fonte: Acervo Pessoal.

Tanto na Figura 5.4, quanto em 5.5 é possível observar que a execução do primeiro fluxo no Spark, tende a ser a mais lenta. Este acontecimento se deve ao fato de que na primeira execução, o Spark realiza o carregamento do contexto e todas funcionalidades básicas necessárias as suas atividades. A fim de exemplificar tal ocorrência realizou-se ainda a execução de um experimento aplicando os mesmos algoritmos, porém desta vez sobre o conjunto previamente rotulado, o qual descreve as propriedades físicas do banco de provas em modo operação no período de coleta. Para fins de teste, este *dataset* foi particionado em cinquenta e um arquivos distintos, cada qual correspondendo a um dispositivo diferente, suas leituras capturadas e a classificação recebida pelos autores de acordo com os registros de ataques. Na etapa de execução do Spark desta experimentação, foi realizado o envio de um novo conjunto de dados apenas após a finalização do anterior. Nas Figuras 5.6 e 5.7 apresentam-se os resultados na execução

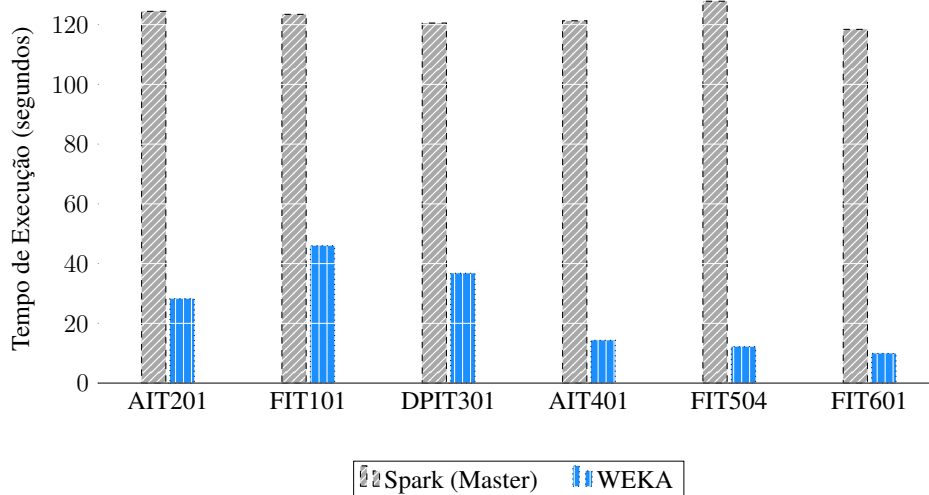
de cada algoritmo. Para fins de demonstração tais imagens apresentam os resultados obtidos em um dispositivo de cada parte do sistema de tratamento de água.

Figura 5.6: Tempos de Execução do Algoritmo *Naive Bayes II*



Fonte: Acervo Pessoal.

Figura 5.7: Tempos de Execução do Algoritmo *Random Forest Tree II*



Fonte: Acervo pessoal.

Como é possível perceber, com a execução realizada de modo não contínuo o tempo obtido com o uso do Spark torna-se relativamente alto quando comparado ao do WEKA, que se mantém pelo menos 91% abaixo no pior caso (maior tempo) do melhor caso (menor tempo) da primeira ferramenta. Isto se da pelo fato de que ao executar o WEKA mesmo de modo

sequencial, para sua execução carrega-se apenas as funcionalidades do algoritmo em questão e não todo contexto como é o caso do Spark. Desta forma em um ambiente IoT onde são dados continuamente gerados, esta aplicação torna-se viável.

Portanto, a realização dos experimentos demonstrou que a utilização do Spark como meio de processamento dos dados aglomerados no *Secure Water Treatment Dataset*, combinados com a aplicação de algoritmos de Mineração de Dados em Python apresenta-se como uma proposta viável para elaboração de um Sistema de Detecção baseado em Anomalias. Um melhor aproveitamento desta infraestrutura é obtido ao realizar a execução simultânea dos fluxos. Tal simultaneidade assemelha-se a geração contínua de dados característica de dispositivos IoT.

Com a realização de tais experimentos, obteve-se uma acurácia superior a 85% em todas as execuções. Este resultado acentua a aplicabilidade desta arquitetura na prospecção de anormalidades registradas nas ações realizadas em uma rede de dispositivos. Entretanto estes resultados se mostram ainda passíveis de melhorias, visando aproximar-se de 100% de acurácia. Assim, um aprimoramento dos algoritmos aplicados ou mesmo a adoção de diferentes métodos surge como opção a ser investigada a fim de acrescer qualidade aos resultados obtidos previamente pelos experimentos.

Todavia, com relação ao desempenho de tempo das ferramentas estudadas no curso deste trabalho, tem-se que a concorrência entre máquinas virtuais podem levar a resultados diferentes do que seria em um ambiente de produção, com servidores ou um *cluster*. Entre possíveis causas destas diferenças está a eventual necessidade de utilização de memórias alocadas em área de *swap* devido a pequena disponibilidade da mesma na máquina de teste, retirando uma das principais características do Spark (*i.e.*, utilização plena de memória principal).

A execução em servidores tende a ter um tempo de processamento consideravelmente menor, dadas características como maior disponibilidade de Memória RAM para uso, maior número de núcleos na Unidade Central de Processamento, permitindo assim maior realização de tarefas de modo paralelo, uma menor concorrência por recursos entre máquina, dispensando a virtualização de recursos computacionais como aqueles que ocorrem em máquinas virtuais. Desta forma, estas possíveis intervenções podem divergir de uma aplicação concreta da proposta aqui realizada, podendo resultar em variações nos tempos de execução colhidos na utilização de máquinas virtuais. Porém, mesmo com estas ressalvas, os experimentos realizados indicam bons resultados.

## 6 CONCLUSÃO

Este trabalho de conclusão propôs uma análise de desempenho com relação ao tempo do uso de uma infraestrutura de *Big Data*, utilizando o Apache Spark. Para este fim, foram comparados os tempos de duas formas distintas de execução em dois algoritmos, sendo uma nesta infraestrutura e outra na ferramenta de Aprendizado de Máquina, WEKA. Os algoritmos trabalhados foram o *Naïve Bayes* e *Random Forest Tree*. Os experimentos realizados tinham como objetivo mostrar o comportamento das ferramentas estudadas quando utilizadas para processamento de volumes de dados diários visando prover um mecanismo de segurança capaz de monitorar os registros coletados dos tráfegos de uma rede envolvendo dispositivos de Internet das Coisas.

De modo a averiguar o comportamento nas execuções dos algoritmos *Naïve Bayes* e *Random Forest Tree*, utilizando a ferramenta Spark contendo apenas o nodo *master*, e ainda a mesma com um *worker* atuando como auxiliar, e por fim a plataforma do WEKA, foram realizados experimentos utilizando um conjunto de dados contendo registros de tráfego de rede em um cenário com sensores e atuadores. O conjunto de dados utilizados foi o *Secure Water Treatment Dataset*, onde neste tem-se um total 946.722 (novecentas e quarenta e seis mil setecentos e vinte e dois) amostras compreendendo trinta atributos que foram coletados ao longo de onze dias a partir de cinquenta e um dispositivos. Este conjunto contém dois módulos, um no qual descrevem-se as propriedades físicas capturadas de cada dispositivo em modo operação no período de coleta, e outro com informações do tráfego de rede coletado usando equipamentos comercialmente disponíveis.

Analisando os resultados obtidos através desta comparação, foi constatado que a atribuição proposta as ferramentas utilizadas, de modo geral, apresentam bons resultados uma vez que a acurácia obtida ultrapassa os 85%. Os tempos de execução do Spark, quando os algoritmos são executados de modo contínuo sobre os dados, demonstram sua capacidade de processamento em grandes volumes de dados e ainda o potencial de uso para um Sistema de Detecção de Intrusão baseado em anomalias, de modo a prevenir e agir em resposta a investidas contra o sistema a que se monitora. Porém não descarta-se a utilização do WEKA como modo de validação dos resultados extraídos, bem como para o processamento de parcelas de dados. Para Tao (2017) quando os dados podem ser transformados com sucesso em inteligência, utilizando-se dados maiores para uma melhor inteligência, é possível obter maior conhecimento sobre a segurança,

assumindo assim uma posição pró-ativa e não reativa.

Este trabalho concentrou esforços na análise de desempenho em relação a tempo de aplicação de algoritmos de Mineração de Dados como meio de prospecção de anomalias em registros de rede aplicados a dispositivos IoT. O uso de uma infraestrutura completa de *Big Data* não foi explorado com profundidade neste trabalho, porém existem ainda outros métodos de desenvolvimento e de técnicas passíveis de utilização para processamento em larga escala utilizando mecanismos de Descobertas de Conhecimento em Bases de Dados. Assim, uma proposta a ser investigada é avaliar o que pode ser feito para otimizar a coleta, geração de fluxos de dados e suas análises. Algumas possibilidades de estudo incluem:

- a) Aumentar o número de nodos auxiliares a fim de analisar com mais afinco o comportamento do Apache Spark.
- b) Analisar a prospecção de anomalias em um ambiente real simulado, empregando o uso de ferramentas para a coleta e armazenamento do dados.
- c) Analisar o comportamento das ferramentas aqui utilizadas com diferentes algoritmos, buscando reduzir o tempo de processamento e aumentar a acurácia dos resultados.
- d) Investir em métodos de captura e análise de registros arbitrários classificados como regulares a fim de certificar o uso das técnicas de Mineração de Dados aplicadas.
- e) Analisar a execução em um ambiente distribuído fisicamente de forma a estressar a arquitetura, averiguando também o *delay* de tráfego que é desprezado em uma simulação local.

Como trabalhos futuros, sugere-se fazer uso de ferramentas como Apache Flume<sup>26</sup>, como meio de coletar dados de rede e transportar fluxos a serem processados. Este processamento pode ainda vir a ser realizado com uso do Spark Streaming<sup>27</sup>, neste traz-se uma API integrada a linguagem do Spark para o processamento de fluxos, permitindo a escrita de transmissões de modo análogo a atividades em lote. Esta ferramenta suporta ainda implementações em Java, Scala e Python. Propondo assim, uma arquitetura completa de estratégias capazes de processar de modo ainda mais eficiente e efetivo investidas maliciosas em um tráfego de rede monitorado. Ou mesmo com a utilização do Apache Kafka<sup>28</sup>, ferramenta capaz de criar fluxos

<sup>26</sup> Disponível em: <https://flume.apache.org>

<sup>27</sup> Disponível em: <https://spark.apache.org/streaming/>

<sup>28</sup> Disponível em: <https://kafka.apache.org>

de dados para transmissão em tempo real obtendo dados confiáveis entre sistemas ou aplicações. Este último, apresenta-se também como uma alternativa na construção de *pipelines* de dados em tempo real e aplicativos de *streaming*, sendo horizontalmente escalável, tolerante a falhas e eficiente.



## REFERÊNCIAS

- ALGHURIED, A. **A Model for Anomalies Detection in Internet of Things (IoT) Using Inverse Weight Clustering and Decision Tree**. 2017. Dissertação (Mestrado em Ciência da Computação) — Dublin Institute of Technology, Irlanda.
- AMARAL, F. **Introdução à Ciência de Dados: mineração de dados e big data**. [S.l.]: Alta Books, 2016.
- BREIMAN, L. Random forests. **Machine learning**, [S.l.], v.45, n.1, p.5–32, 2001.
- BROWN, B. et al. Big data: the next frontier for innovation, competition, and productivity. **McKinsey Global Institute**, [S.l.], 2011.
- KNUDSEN, L. R.; ROBSHAW, M. J. B. **Brute Force Attacks**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p.95–108.
- BUTUN, I. et al. Anomaly detection and privacy preservation in cloud-centric Internet of Things. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATION WORKSHOP (ICCW), 2015. **Anais...** [S.l.: s.n.], 2015. p.2610–2615.
- CHAKRABARTI, S. **Mining the Web: discovering knowledge from hypertext data**. [S.l.]: Elsevier Science, 2002. (The Morgan Kaufmann Series in Data Management Systems).
- DARPA Dataset. Acesso em: 17 de outubro 2017, Disponível em: <https://ll.mit.edu/ideval/data/>.
- DELL EMC. Acesso em: 28 de agosto de 2017, Disponível em: <https://www.emc.com/leadership/digital-universe/index.htm>.
- DI PIETRO, R.; MANCINI, L. **Intrusion Detection Systems**. [S.l.]: Springer US, 2008. (Advances in Information Security).
- DOODY, P.; SHIELDS, A. Mining network relationships in the internet of things. In: SELF-AWARE INTERNET OF THINGS, 2012. **Proceedings...** [S.l.: s.n.], 2012. p.7–12.
- EIBE, F. et al. The WEKA workbench. **Online appendix for “data mining: practical machine learning tools and techniques.”: Fourth Morgan Kaufmann**, [S.l.], 2016.

FAYYAD, U. M.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From Data Mining to Knowledge Discovery: an overview. In: FAYYAD, U. M. et al. (Ed.). **Advances in Knowledge Discovery and Data Mining**. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1996. p.1–34.

FU, R. et al. An intrusion detection scheme based on anomaly mining in internet of things. In: IET INTERNATIONAL CONFERENCE ON WIRELESS, MOBILE MULTIMEDIA NETWORKS (ICWMMN 2011), 4. **Anais...** [S.l.: s.n.], 2011. p.315–320.

GOH, J. et al. A Dataset to Support Research in the Design of Secure Water Treatment Systems. In: Proceedings of the 11th International Conference on Critical Information Infrastructures Security. **Anais...** Springer, 2016. To Appear.

GRACE, L. K. J. et al. Analysis of Web Logs and Web User in Web Mining. **CoRR**, [S.l.], v.abs/1101.5668, 2011.

Hogzilla IDS. Acesso em: 16 de outubro 2017, Disponível em: <http://ids-hogzilla.org>.

KOBAYASHI, S. et al. Mining causes of network events in log data with causal inference. In: IFIP/IEEE SYMPOSIUM ON INTEGRATED NETWORK AND SERVICE MANAGEMENT (IM), 2017. **Anais...** [S.l.: s.n.], 2017. p.45–53.

LEE, W.; STOLFO, S. J. Data Mining Approaches for Intrusion Detection. In: CONFERENCE ON USENIX SECURITY SYMPOSIUM - VOLUME 7, 7., Berkeley, CA, USA. **Proceedings...** USENIX Association, 1998. p.6–6. (SSYM'98).

MEHMOOD, A. et al. Protection of big data privacy. **IEEE access**, [S.l.], v.4, p.1821–1834, 2016.

NIELSEN, F. Å. Data Mining with Python (Working draft). , [S.l.], 2015.

PEER Research Big Data Analytics. Acesso em: 18 de outubro de 2017, Disponível em: <http://www.intel.com/content/www/us/en/big-data/data-insights-peer-research-report.html>.

PREUSS, J. O. et al. Uma Proposta de Arquitetura para Identificação de Anomalias em Redes IoT utilizando Registros de Logs. In: ESCOLA REGIONAL DE REDES DE COMPUTADORES, 15., Santa Maria. **Anais...** Sociedade Brasileira de Computação, 2017. p.193–196. (Anais da Escola Regional de Redes de Computadores, v.1).

RATHORE, M. M. et al. Hadoop Based Real-Time Intrusion Detection for High-Speed Networks. In: IEEE GLOBAL COMMUNICATIONS CONFERENCE (GLOBECOM), 2016. **Anais...** [S.l.: s.n.], 2016. p.1–6.

RONALDO GOLDSCHMIDT, E. P. e. E. B. **Data Mining**: conceitos, técnicas, algoritmos, orientações e aplicações. [S.l.]: Elsevier Brasil, 2015.

RØNNING, M. V. K. **Mitigating DDoS attacks using data mining and density-based geographical clustering**. 2017. Dissertação (Mestrado em Ciência da Computação) — University of Oslo.

ROSSUM, G. **Python Reference Manual**. Amsterdam, The Netherlands, The Netherlands: [s.n.], 1995.

SAID, O.; MASUD, M. Towards internet of things: survey and future vision. **International Journal of Computer Networks**, [S.l.], v.5, n.1, p.1–17, 2013.

SEELAMMAL, C. et al. Computational intelligence in intrusion detection system for snort log using hadoop. In: INTERNATIONAL CONFERENCE ON CONTROL, INSTRUMENTATION, COMMUNICATION AND COMPUTATIONAL TECHNOLOGIES (ICCICCT), 2016. **Anais...** [S.l.: s.n.], 2016. p.642–647.

SHARMA, R. et al. Towards MapReduce based classification approaches for Intrusion Detection. In: INTERNATIONAL CONFERENCE - CLOUD SYSTEM AND BIG DATA ENGINEERING (CONFLUENCE), 2016. **Anais...** [S.l.: s.n.], 2016. p.361–367.

SOARES, H. et al. Information Security Aspects of Public Software. In: FIFTH INTERNATIONAL CONFERENCE ON MANAGEMENT OF EMERGENT DIGITAL ECOSYSTEMS, New York, NY, USA. **Proceedings...** ACM, 2013. p.336–339. (MEDES '13).

SUTHAHARAN, S. Big Data Classification: problems and challenges in network intrusion prediction with machine learning. **SIGMETRICS Perform. Eval. Rev.**, New York, NY, USA, v.41, n.4, p.70–73, Apr. 2014.

TAHERKORDI, A. et al. From IoT Big Data to IoT Big Services. In: SYMPOSIUM ON APPLIED COMPUTING, New York, NY, USA. **Proceedings...** ACM, 2017. p.485–491. (SAC '17).

- TANENBAUM, A. S. **Redes de computadoras**. [S.l.]: Pearson Educación, 2003.
- TAO, B. Security Big Data Analytics. **Cybersecurity France-Japan: Scientific collaborations between France and Japan**, [S.l.], 2017.
- TAURION, C. **Big data**. [S.l.]: Brasport, 2013.
- TSAI, C. W. et al. Data Mining for Internet of Things: a survey. **IEEE Communications Surveys Tutorials**, [S.l.], v.16, n.1, p.77–97, First 2014.
- UCKELMANN, D. et al. An Architectural Approach Towards the Future Internet of Things. In: ARCHITECTING THE INTERNET OF THINGS, Berlin, Heidelberg. **Anais...** Springer Berlin Heidelberg, 2011. p.1–24.
- WEISER, M. The computer for the 21st century. **Mobile Computing and Communications Review**, [S.l.], v.3, n.3, p.3–11, 1999.
- WITTEN, I. H. et al. **Data Mining**: practical machine learning tools and techniques. [S.l.]: Morgan Kaufmann, 2016.
- ZHU, Q. et al. IOT Gateway: bridging wireless sensor networks into internet of things. In: IEEE/IFIP INTERNATIONAL CONFERENCE ON EMBEDDED AND UBIQUITOUS COMPUTING, 2010. **Anais...** [S.l.: s.n.], 2010. p.347–352.