Instituto Federal de Educação, Ciência e Tecnologia da Paraíba

Campus João Pessoa

Programa de Pós-Graduação em Tecnologia da Informação

Nível Mestrado Profissional

MORONI NERES VIEIRA

MEDIÇÕES E AVALIAÇÕES COMPARATIVAS DE DESEMPENHO E ENERGIA DE ALGORITMOS DE MACHINE LEARNING PARA MITIGAR AMEAÇAS DE DISPONIBILIDADE EM AMBIENTE IOT.

DISSERTAÇÃO DE MESTRADO

JOÃO PESSOA

2021

Moroni Neres Vieira

MEDIÇÕES E AVALIAÇÕES COMPARATIVAS DE DESEMPENHO E ENERGIA DE ALGORITMOS DE MACHINE LEARNING PARA MITIGAR AMEAÇAS DE DISPONIBILIDADE EM AMBIENTE IOT.

Dissertação apresentada como requisito parcial para obtenção do título de Mestre em Tecnologia da Informação, pelo Programa de Pós-Graduação em Tecnologia da Informação do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – IFPB.

Orientador: Prof. Dr[a]. Luciana Pereira Oliveira

João Pessoa

2021

Dados Internacionais de Catalogação na Publicação – (CIP)
Biblioteca Nilo Peçanha do IFPB, *campus* João Pessoa.

V658m Vieira, Moroni Neres.

Medições e avaliações comparativas de desempenho e energia de algoritmos de machine learning para mitigar ameaças de disponibilidade em ambiente IOT / Moroni Neres Vieira. – 2021.
125 f. : il.
Dissertação (Mestrado – Tecnologia da Informação) – Instituto Federal de Educação da Paraíba / Programa de Pós-Graduação em Tecnologia da Informação, 2021.
Orientação: Profª D.ra Luciana Pereira Oliveira.

1. Inteligência artificial. 2. IOT- segurança. 3. Consumo energético. 4. Algoritmos. I. Título.

CDU 004.8 (043)

Lucrecia Camilo de Lima
Bibliotecária – CRB 15/132

Moroni Neres Vieira

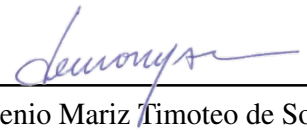
MEDIÇÕES E AVALIAÇÕES COMPARATIVAS DE DESEMPENHO E ENERGIA DE ALGORITMOS DE MACHINE LEARNING PARA MITIGAR AMEAÇAS DE DISPONIBILIDADE EM AMBIENTE IOT.

Dissertação apresentada como requisito parcial para obtenção do título de Mestre em Tecnologia da Informação, pelo Programa de Pós-Graduação em Tecnologia da Informação do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – IFPB.

Aprovado em 04 de Agosto de 2021.

BANCA EXAMINADORA:

Prof. Dr. Luciana Pereira Oliveira – IFPB
Avaliador



Prof. Dr. Denio Mariz Timoteo de Sousa – IFPB
Avaliador Interno

Prof. Dr. Alisson Vasconcelos de Brito – UFPB
Avaliador Interno

Prof. Dr. Augusto Jose Venancio Neto – UFRN
Avaliador Externo

Prof. Dr[a]. Luciana Pereira Oliveira (Orientador)

Visto e permitida a impressão
João Pessoa

Prof. Dr. Francisco Petronio Alencar de Medeiros
Coordenador PPPGTI

Este trabalho é dedicado a todos que de alguma forma contribuíram para que eu pudesse chegar aqui, especialmente que sirva de exemplo para meus amados filhos, que prossigam na vida com humildade e respeito

AGRADECIMENTOS

Agradeço primeiramente a Deus por me conceder o dom da vida, conhecimentos necessários e saúde para que eu chegasse até o fim dessa jornada, sempre iluminando e direcionando o caminho. Agradeço também a minha família amada que proporcionou todos os momentos na minha vida que me fizeram refletir enquanto pessoa e ser humano para tomar as decisões adequadas. Ademais, um agradecimento póstumo ao meu pai (*in memoriam*), com seu exemplo e sabedoria educou-me e compartilhou lições valiosas, a minha mãe que com muito esforço e dedicação pessoal sempre me proporcionou o básico para minha vida.

Agradeço a minha orientadora Prof. Drª Luciana Oliveira pelo seu empenho, dedicação, paciência, conhecimento e por sua disponibilidade ímpar, sem ela realmente não seria possível chegar até aqui e desenvolver este trabalho.

Aos colegas e também professores deste curso que, com muita tranquilidade e respeito transmitiram o conhecimento necessário para que eu chegasse até aqui.

Por fim, aos meus filhos Ana Beatriz e Miguel Felipe, nada disso poderia ser concretizado sem pensar neles, essa lembrança diária e pertinente motivou a resolução de cada problema e a escrita de cada capítulo, parágrafo e linha deste trabalho.

RESUMO

O crescimento da utilização dos dispositivos IoT baseados em Linux em um contexto mundial trouxe desafios na área de segurança da informação, e ao empregar os protocolos de comunicação (AMQP, MQTT, CoAP, HTTPS) para troca de informações entre esses dispositivos e uma vez que esses dispositivos sendo alvo de atacantes podem utilizá-los para atacar outros *smart objects*. Diante desse problema, é imprescindível que os sistemas de segurança evoluam e permitam que sistemas de detecção de intrusão realizem a detecção desses ataques. Essa detecção poderá ser dividida entre outros aparelhos no mesmo domínio de rede. Ao utilizar esses mecanismos anteriormente citados, o projeto faz o uso do Aprendizado de Máquina da Inteligência Artificial a fim de apresentar uma forma de melhorar o sistema de detecção realizando a classificação do tráfego entre legítimo e ataque por meio da utilização de algoritmos de aprendizado. Os cenários, avaliações e algoritmos para compor esse sistema de detecção de intrusão foram adquiridos por meio de duas revisões sistemáticas. Na primeira revisão, verificou-se que existem poucos estudos na área de disponibilidade. Já na segunda revisão foram encontrados trabalhos reprodutíveis com códigos fontes disponíveis, ainda mais, parâmetros e métricas de avaliação dos algoritmos de aprendizado (*Logistic Regression, k-Nearest Neighbours, Gaussian Naive Bayes, Decision Trees, Random Forests e Support Vector Machine (linear and RBF kernel)*). Além disso, a partir dessa revisão adquiriu-se os dados dos ataques aos protocolos MQTT e o modelo topológico do cenário controlado para aquisição dos dados de ataques do CoAP. Ademais, após a aquisição desses dados foram executados os algoritmos, realizando-se o aprendizado para classificar o tráfego entre legítimo e ataque com base em dados sintéticos dos fluxos unidirecionais e bidirecionais do MQTT e CoAP. Também, a fase de testes dos modelos ocorreu no mesmo ambiente em que ocorreu o aprendizado, por isso, foram utilizados hiperparâmetros para embaralhar os dados. Após esses passos, foram produzidas informações de consumo de energia dos componentes de hardware (CPU, Memória RAM, *Package* e GPU) e do desempenho dos algoritmos de ML. Por fim, com a produção dessas informações foram realizadas avaliações de desempenhos dos algoritmos com as métricas de acurácia, precisão e F1-Score. Outrossim, foi observado a média e o intervalo de confiança do consumo de energia deles sobre os hardwares. Dessa forma, foi possível observar quais algoritmos de ML foram eficazes na detecção de intrusões e a partir das médias e intervalos de confiança obtidos do dados de consumo energético quais deles foram eficazes na distribuição de regras entre sistemas de detecção na rede.

Palavras-chaves: IOT; inteligência artificial; consumo energético. Disponibilidade.

ABSTRACT

The growing use of Linux-based IoT devices in a global context has brought challenges in information security area. By employing communication protocols such as AMQP, MQTT, CoAP, and HTTPS to exchange information between these devices results in those devices being targeted by attackers in order to attack other smart objects. Faced with this problem, it is essential that security systems evolve and allow intrusion detection systems to detect these attacks. This discovery can be split among other devices in the same network domain. By using these mechanisms, the project makes use of Artificial Intelligence Machine Learning to present a way to improve the detection system by classifying traffic in either legitimate or attack through the usage of learning algorithms. The scenarios, assessments, and algorithms to compose this intrusion detection system were acquired through two systematic reviews. In the first review, it was found that there are few studies regarding to availability. In the second review, reproducible works were found with available source codes, moreover, parameters and metrics for evaluating the learning algorithms – Logistic Regression, k-Nearest Neighbors, Gaussian Naive Bayes, Decision Trees, Random Forests and Support Vector Machine (linear and RBF kernel)). In addition to that, both data on attacks to MQTT protocols and the topological model of the controlled scenario for acquisition of CoAP attack data were acquired from this review. Furthermore, after acquiring these data, the algorithms were executed learning to classify the traffic either in legitimate or in attack based on synthetic data from the unidirectional and bidirectional flows of MQTT and CoAP. Also, the testing phase of the models took place in the same environment of the learning, therefore, hyperparameters were used to scramble the data. After those steps, information on the energy consumption of the hardware components (CPU, RAM, Package and GPU) and the performance of the ML algorithms were produced. Finally, with the production of this information, performance evaluations of the algorithms were carried out with metrics of accuracy, precision, and F1-Score. Besides that, the mean and confidence interval of their energy consumption on the hardware were observed. Thus, it was possible to observe which ML algorithms were effective in detecting intrusions and, from the means and confidence intervals obtained from energy consumption data, which of them were effective in the distribution of rules between in the network detection systems.

Key-words: IOT; artificial intelligence. energy consumption. availability.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AMQP	*Advanced Message Queuing Protocol*
XMPP	*Extensible Messaging and Presence Protocol*
UPnP	*Universal Plug and Play*
IoT	*Internet Of Things*
WoT	*Web of Things*
IP	*Internet Protocol Version* 4
IPv6	*Internet Protocol Version* 6
TCP	*Transmission Control Protocol*
IDC	*International Data Corporation*
IoRT	*Internet of Robotic Things*
IA	Inteligência Artificial
CoAP	*Constrained Application Protocol*
MQTT	*Message Queuing Telemetry Transport*
DoS	*Denial of Service*
DDoS	*Distributed Denial Of Service*
DL	*Deep Learning*
SDN	*Software Defined Network*
IFRN	Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte
ML	*Machine Learning*
UDP	*User Datagram Protocol*
DTLS	*Datagram Transport Layer Security*
TLS	*Transport Layer Security*
OsCoAP	*Object Security of CoAP*
AEAD	*Authenticated Encryption with Associated Data*

IDS	*Intrusion Detection System*
6LowPAN	IPv6 *over Low-Power Wireless Personal Area Networks*
GPU	*Graphics Processing Unit*
OS	*Operacional System*
API	*Application Programming Interface*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
NN	*Neural Network*
KNN	*K–Nearest Neighbours*
SVM	*Support Vector Machines*
RBF	*Radial Basis Function*
TI	Tecnologia da Informação
IDS	*Intrusion Detection System*
NMAP	*Network Mapper*
GUI	*Graphical User Interface*
ICMP	*Internet Control Message Protocol*
M2M	*Machine-to-Machine*
RAM	*Random Access Memory*
ROM	*Read-Only Memory*
URI	*Uniform Resource Identifier*
CON	*Confirmable*
NON	*Non−confirmable*
ACK	*Acknowledgement*
OASIS	*Organization for the Advancement of Structured Information Standards*
CPU	*Central Processing Unit*
RAPL	*Running Average Power Limit*

CF	*Californium*
SC	*Scandium*
KDE	K *Desktop Environment*
RSL	Revisão Sistemática da Literatura
WSN	*Wireless Sensor Networks*
GB	*Gibabyte*
DDR	*Double-Data-Rate*
MB	*Megabyte*
HD	*Hard Disk*
SSH	*Secure Shell*
SSD	*Solid State Drives*
PV	Positivo Verdadeiro
FP	Falso Positivo
VN	Verdadeiro Negativo
FN	Falso Negativo

SUMÁRIO

1 INTRODUÇÃO

1.1 Problema

Os recentes avanços na área de tecnologia impulsionaram o uso de dispositivos inteligentes, os quais podem ser denominados também de *smart objects*. Além disso, as organizações acadêmicas e empresas de diversos segmentos desenvolvem aplicações para controles, medições e disponibilização de informações, dentre outras. Habitualmente, utilizam-se desses dispositivos para resolver problemas no cotidiano e, dessa forma, auxiliam também nas tomadas de decisões. Atualmente, com a Internet das Coisas, os objetos estão incorporando tecnologia e meios de comunicação para troca de dados entre si. Nesse contexto, eletrodomésticos, relógios inteligentes, sensores de presença, câmeras, carros e outros meios de transporte estão conectados. Os dispositivos IoT (Internet of Things) já fazem parte do dia a dia das pessoas e, por isso, novas demandas têm impulsionado o desenvolvimento de sistemas que monitoram ou que fazem a leitura de dados através desses dispositivos. Desse modo, este estudo atenderá aos futuros projetos com finalidade tecnológica no contexto de dispositivos IoT, utilizando-se de mecanismos de segurança para detectar ataques cibernéticos em uma organização acadêmica.

O IoT ou WoT (*Web of Things*) (RAHMAN; SHAH, 2016) é um grupo de objetos estáticos ou móveis interconectados, como dispositivos equipados com módulos de comunicação, sensores e atuadores conectados à internet (SENGUPTA; RUJ; BIT, 2020). Para uma melhor compreensão, o IoT é a integração dos objetos já conhecidos (carros, geladeira, microondas, máquina de lavar, fogão) com dispositivos eletrônicos capazes de trocar informações entre si.

Ainda assim, os dispositivos IoT são empregados em diversas aplicações, por exemplo, para o campo da saúde (KRISHNA; SAMPATH, 2017) que descreve um sistema baseado em IoT para monitoramento da saúde de pacientes. Além disso, outras aplicações baseadas no microcomputador raspberry, foram descritas por (KURKOVSKY; WILLIAMS, 2017) e a partir dele, utilizou-se sensores que captam dados do ambiente.

Estima-se que no ano de 2022, o número de dispositivos conectados seja de 18 bilhões (SENGUPTA; RUJ; BIT, 2020). A IDC projeta que a quantidade de dados terá uma taxa de crescimento anual composta de 28,7%, até o ano de 2025, 41,6 bilhões de dispositivos IoT irão gerar 79,4 zettabytes de dados (SECURITY, 2019).

Além do mais, levando-se em consideração os números projetados da utilização desses dispositivos IoT no cenário mundial, observa-se que é um tema bastante explorado e em desenvolvimento no contexto acadêmico. Além disso, encontram-se trabalhos no repositório institucional do IFRN[1] a disposição de forma pública alguns trabalhos sobre o tema IoT (SILVEIRA; MELO,

[1] https://https://memoria.ifrn.edu.br/

2019), (CRUZ et al., 2020), (TORRES; SILVA, 2018) e (FERREIRA et al., 2017).

Diante da utilização desses dispositivos no contexto acadêmico e empresarial, é necessário que eles funcionem em rede para troca de informações e segundo (ANDY; RAHARDJO; HANINDHITO, 2017) os cinco protocolos que atuam nessa troca de informações com maior uso são: o protocolo HTTP (*Hypertext Transfer Protocol*), o MQTT (*Message Queuing Telemetry Transport*), o CoAP (*Constrained Application Protocol*), o XMPP (*Extensible Messaging and Presence Protocol*) e o AMQP (*Advanced Message Queuing Protocol*).

Nesse sentido de utilização desses protocolos, consideram-se os seguintes critérios na escolha deles: eficiência energética (energia consumida durante o tempo de execução do dispositivo), desempenho (tempo entre o envio da mensagem e o tempo de reconhecimento), utilização dos recursos (CPU, memória RAM e memória ROM), além desses critérios, a confiabilidade é escolhido, também, para alguns cenários (MUN; DINH; KWON, 2016). Assim, o ambiente poderá exigir que o protocolo forneça funcionalidades avançadas (persistência de mensagens, mensagens arbitrárias, dentre outras). Ainda assim, a capacidade de ter mecanismos de confiabilidade e entrega de mensagens *multicast* podem ser pré-requisitos para os sistemas.

Outros critérios podem ser observados, dessa forma. Segundo (NAIK, 2017) realizou-se um estudo comparativo entre os protocolos HTTP, AMQP, CoAP e MQTT. Diante disso, observou-se que o protocolo HTTP requereu maior consumo de energia, maior largura de banda de rede e observou-se uma sobrecarga no tamanho do cabeçalho de pacotes de rede, pois este é indefinido , pois, este protocolo foi originalmente concebido para o contexto da Web, e não para aplicações no contexto da IoT. No decurso da análise, observou-se que o protocolo AMQP é leve, pois, seu cabeçalho de pacotes de rede tem apenas 8 bytes de tamanho, porém, seus requisitos para confiabilidade na troca de mensagens, provisionamento no transporte de mensagens e interoperabilidade aumentam o tamanho total da mensagem. Diante das análises comparativas do estudo, os protocolos MQTT e o CoAP apresentaram menor consumo de energia, menor sobrecarga de dados no cabeçalho e menor latência. Além disso, eles exigiram menor largura de banda do que os outros protocolos anteriormente citados. Portanto, por causa das exigências baseadas nos critérios do ambiente e de funcionalidade avançadas, o protocolo MQTT e o COAP se destacam, pois, diante dos outros protocolos são as melhores escolhas para um ambiente de dispositivos IoT em redes, ou seja, de acordo com (CARO et al., 2013).

No contexto de utilização dos protocolos, observou-se no gráfico representado pela figura 1 que vários incidentes de segurança foram analisados ao longo do ano de 2020, causados por *malwares, scans, rootkits*, dentre outros, e, esses representam os ataques cibernéticos aos protocolos IoT.

No gráfico supracitado, observa-se que a maior porcentagem de incidentes reportados ao CERT.br [2] é de Scan, que representa 58,81% de incidentes verificados. Esse incidente está

[2] https://www.cert.br/stats/incidentes/

relacionado com atividade de propagação de botnets de IoT, o qual realiza varredura nas portas TCP 23, 22, 81, 5555, 8000 e 8080 com tentativas de força bruta de credenciais (CERT.BR, 2020b). Ademais, percebe-se que as maiores ameaças estão concentradas no crescimento de ataques cibernéticos do tipo reconhecimento (SHAIKH et al., 2008), especificamente ataques de scan de portas virtuais e versões de protocolos, conforme demonstrado no gráfico representado pela figura 1.

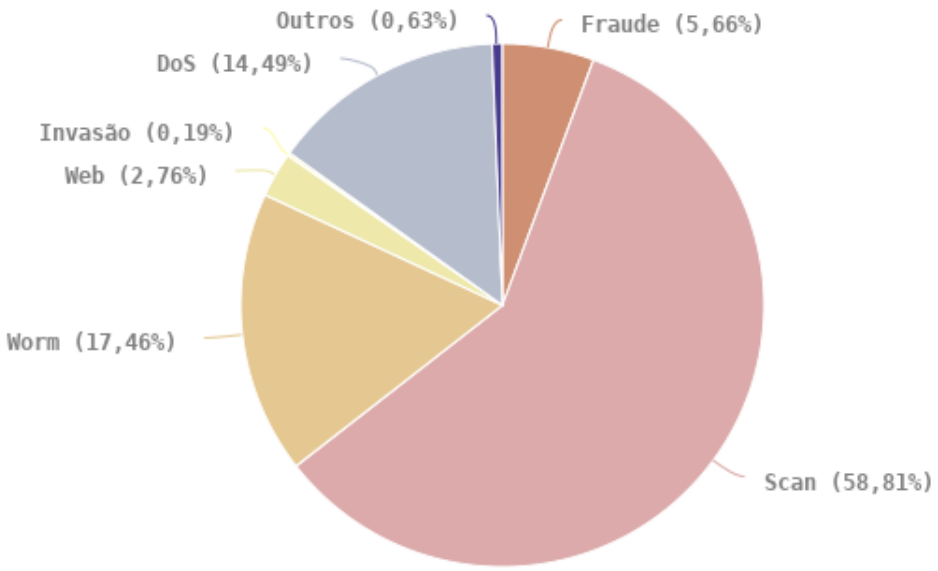


Figura 1 – Tipos de ataques reportados ao CERT.br
de janeiro a junho de 2020

fonte:(CERT.BR, 2020a)

Outrossim, (METONGNON; SADRE, 2018) apresentou-se um estudo sobre ataques aos protocolos MQTT , CoAP e UPnP (*Universal Plug and Play*) específicos de IoT. Ademais, utilizou-se a ferramenta denominada *honeypot* com o objetivo de coletar dados de ataques para serem utilizados no referido estudo. Por fim, obtiveram-se números de ataques direcionados às portas dos protocolos MQTT (1883), MQTT sobre SSL (8883), CoAP (5683) e UPnP (1900).

Diante disso, dos três protocolos apresentados no estudo somente o MQTT e o CoAP serão abordados para a análise em curso, pois, o UPnP contém uma abordagem diferente do objetivo desta pesquisa, ou seja, ele é utilizado para configuração e descoberta de serviços em um dispositivo de rede. Esses protocolos sofreram ataques cibernéticos, sendo que o protocolo UPnP alcançou a primeira posição dos ataques analisados, o MQTT alcançou a segunda posição e o CoAP alcançou a terceira posição dentre a quantidade de ataques contabilizados.

Os ataques podem causar a interrupção e indisponibilidade de informações por meio de técnicas de intrusão denominadas DoS ou DDoS direcionadas para os *smart objects*, estes podem ser utilizados para realizar ataques para outros *hosts* na rede.

Diante dessa breve contextualização acerca do problema, as seguintes questões de pesquisa foram levantadas:

- Quais as recomendações para obter um cenário mais seguro de IoT considerando o MQTT e o CoAP?
- Quais ataques podem ser detectados utilizando-se os algoritmos ML?
- Quais características em um conjunto de dados produzidos sinteticamente de tráfegos legítimo e ataques do MQTT e do CoAP podem ser considerados para treinamento nos algoritmos de ML?
- Quais algoritmos de ML são mais indicados em relação a desempenho e energia para detecção de ataques?

Para responder a esses questionamentos, é proposto este estudo que avaliará os algoritmos de ML para detecção dos ataques e consumo energético. Ainda, os resultados dessas avaliações serão discutidas na conclusão do trabalho em tela.

1.2 Justificativa

Diante do problema exposto anteriormente, os desafios concentram-se na proteção desses sistemas e dispositivos IoT. Muitos estudos estão sendo desenvolvidos utilizando-se de modelagem, simulação e experimentos, e os resultados têm destacado a utilização de algoritmos em Inteligência Artificial (IA), por exemplo, (RAZAFIMANDIMBY; LOSCRI; VEGNI, 2016) inseriu o conceito de IoRT (*Internet of Robotic Things*), onde os conjuntos de dispositivos IoT monitora eventos e consegue tomar decisões para controlar ou manipular objetos no mundo físico.

Diante do problema de ataques aos protocolos MQTT e CoAP, os desafios concentram-se na proteção desses protocolos, sistemas e dispositivos IoT. Muitos estudos estão sendo desenvolvidos utilizando-se de modelagem, simulação e experimentos, e os resultados têm destacado a utilização de algoritmos em Inteligência Artificial (IA), por exemplo, (RAZAFIMANDIMBY; LOSCRI; VEGNI, 2016) inseriu o conceito de IoRT (*Internet of Robotic Things*), onde os conjuntos de dispositivos IoT monitora eventos e consegue tomar decisões para controlar ou manipular objetos no mundo físico.

Contudo, a IA é utilizada em problemas que exigem a habilidade de aprendizado, a capacidade de compreender com a experiência, a capacidade de adquirir conhecimento e retê-lo. Ademais, desenvolve a capacidade de resposta com rapidez e sucesso diante de uma nova situação, também, utilizando-se da razão na resolução de problemas para direcionamento de uma conduta eficaz (FETZER, 1990).

Diante disso, a IA é utilizada para vários campos de pesquisas, sendo um deles a segurança da informação, utilizada para o tratamento de incidentes, proteção de sistemas e tecnologias de comunicação. Dentro do campo de IA há subcampos, por exemplo, Redes neurais, algoritmos genéticos, aprendizado profundo, processamento de linguagem natural e aprendizado de máquina (*Machine Learning* - ML), que será utilizado neste trabalho.

O *Machine Learning* é utilizado em problemas de classificação, onde, a principal tarefa é a identificação de um rótulo de classe para cada instância com o objetivo de minimizar erros no processo de classificação, as técnicas de classificação constroem modelos que facilitam a identificação de classes com precisão para um determinado conjunto de dados (ALLAHYARI et al., 2017). Assim, essas técnicas são adotadas neste trabalho, pois são amplamente utilizadas para o desenvolvimento de IDS (*Intrusion Detection System*) (BISWAS et al., 2018), (ALJAWARNEH; ALDWAIRI; YASSEIN, 2018).

Além disso, implementar um IDS utilizando-se de ML em um ambiente de pleno funcionamento da rede que exige recursos e uma compreensão dos desafios técnicos e operacionais, uma vez que a rede poderá ficar inoperante e os usuários não poderão utilizar os serviços durante um tempo indeterminado. Por causa disso, utilizar-se-á um ambiente controlado para estudos e adequações de modelos de ML. Dessa forma, é possível realizar um estudo inicial sobre os os modelos que serão eficazes na detecção de anomalias em uma rede com dispositivos IoT.

Ainda, vários IDS poderão compor o ambiente de rede. Eles verificarão os pacotes e estes poderão percorrer vários caminhos, sendo direcionados por diversos equipamentos de rede denominados roteadores. Nesse contexto, a informação pula de roteador em roteador e esse procedimento é denominado roteamento.

Tais roteadores e IDS consomem energia significativa (HEDDEGHEM et al., 2009). Por isso, pode-se considerar uma perspectiva para que as averiguações de pacotes sejam distribuídas na rede, com isso, a energia será distribuída entre os IDS cuja denominação é *Distributed* IDS (DIDS). Dessa forma, também é possível realizar a distribuição de energia e a análise dos pacotes ao longo do caminho de roteamento, pois, eles são segmentados e verificados por cada IDS na rede (MIGLIARDI; MERLO, 2011).

Ademais, a redução de consumo de energia nos IDS é influenciada pela gestão de políticas na rede, tal procedimento ocorre devido à troca de regras entre os IDS. Contudo, necessita-se de uma gestão baseada em políticas de segurança com colaboração entre protocolos de rede, assim sendo, a construção de todo esse cenário é realizado com apoio de uma rede programável (OLIVEIRA; SADOK, 2013). Diante disso, este trabalho procura contribuir no sentido de medição de energia, ou seja, realizar a medição de energia na fase de treinamento e testes dos algoritmos de ML.

Portanto, diante do contexto de segurança em redes de dispositivos IoT, do aumento da utilização desses dispositivos por causa dos projetos em ambientes acadêmicos, assim como, em

outros ambientes, da perspectiva do aumento dos ataques projetados para os próximos anos, e, além, dos desafios da disponibilidade em sistemas, faz-se necessário a utilização de algoritmos no contexto de ML para mitigar ataques aos protocolos de comunicação CoAP e MQTT.

Por fim, a proposta deste trabalho é avaliar o desempenho dos algoritmos de ML em um contexto de IDS por meio de um ambiente IoT controlado. Além disso, comparar o consumo energético dos algoritmos de ML na fase de aprendizado e testes deles.

1.3 Objetivos

1.3.1 Objetivo geral

Comparar o desempenho dos algoritmos de ML para os problemas de classificação no contexto de IDS para detecção de anomalias de tráfego de rede, provocados pelos ataques *agressive scan*, *scan* UDP e força bruta ao serviço ssh, isso, sob os protocolos MQTT e CoAP em um cenário controlado, e, baseado em uma rede de dispositivos IoT. Além disso, comparar o consumo energético desses modelos por meio da medição de energia na fase de aprendizado e testes dos algoritmos.

1.3.1.1 Objetivos específicos

- Definir os parâmetros e métricas da pesquisa, utilizando como instrumento revisões sistemáticas da literatura;
- Coletar os dados do protocolo CoAP para a pesquisa por meio da construção um ambiente virtualizado;
- Transformar e preparar os dados para sintetizar os datasets dos ataques para os protocolos MQTT e CoAP;
- Sintetizar os datasets por meio dos tratamentos dos dados para os fluxos unidirecionais e bidirecionais dos protocolos MQTT e CoAP, assim como, dos ataques;
- Tratar os dados de desempenho e do consumo de energia dos algoritmos de ML na fase de treinamento e testes;
- Avaliar os resultados de forma comparativa dos desempenhos e consumos de energia dos algoritmos.

1.4 Trabalhos relacionados

A segurança de dados em dispositivos IoT tem sido estudada na literatura com ênfase em metodologias e soluções que atendam aos requisitos da segurança e que sejam suficientes para

implementação em equipamentos com perfil de baixo consumo de recursos. Tradicionalmente, os protocolos MQTT e CoAP têm suporte aos recursos de segurança. O CoAP pode implementar DTLS (*Datagram Transport Layer Security*) e o MQTT pode fazer o uso da criptografia TLS (*Transport Layer Security*). Diante das abordagens propostas na literatura para aumentar a segurança (RANDHAWA; HAMEED; MIAN, 2019) focou na segurança da camada de pilha IoT, especificamente, no componente proxy do CoAP. Ele propôs o OSCoAP (*Object Security of CoAP*), o qual realiza a autenticação, proteção de reprodução, confidencialidade e integridade das mensagens por meio de um algoritmo de cifra de bloco AEAD (*Authenticated Encryption with Associated Data*). Além disso, o trabalho abordou essas técnicas em dispositivos que utilizam o protocolo ZigBee (802.15.4) para a comunicação. Desse modo, este trabalho faz a abordagem por meio de modelos de ML, isto é, uma forma eficiente de IDS (*Intrusion Detection System*) ao protocolo CoAP sem criptografia usando dispositivos IoT.

Em (PERETTI; LAKKUNDI; ZORZI, 2015) os pesquisadores apresentaram uma solução desenvolvida no grupo SIGNET, Departamento de Engenharia da Informação da Universidade de Pandova. Ademais, eles apresentaram os impactos do DTLS na pilha 6LoWPAN (*IPv6 over Low-Power Wireless Personal Area Networks*) nos dispositivos IoT realizando as medições no uso de memória, tempo computacional, sobrecarga de energia e a sobrecarga de pacotes introduzida pelo cabeçalho DTLS. Neste trabalho foi abordado o impacto das execuções dos modelos de ML para detecção de anomalias de rede observando o consumo virtual de energia nos componentes de hardware, memória, processador, GPU (*Graphics Processing Unit*), *Package* do processador e o tempo computacional para executá-los.

No trabalho de (RAZA et al., 2013) foi realizado um estudo sobre a implementação de mecanismos de compressão por meio do protocolo 6LoWPAN para se alcançar a eficiência energética com a redução da mensagem nos dispositivos IoT. Ademais, neste estudo utilizou-se o emulador Contiki OS (*Operacional System*) (DUNKELS; GRONVALL; VOIGT, 2004). Neste estudo não foram observados a compressão dos dados, levando em consideração que a maioria dos dispositivos IoT enviam seus dados sem compressão, outrossim, não foi utilizado emulador de IoT. Assim, os dados foram gerados a partir de computadores virtuais com configuração similar aos dispositivos IoT.

Além disso, foi observado no trabalho de (KUMAR; DEZFOULI, 2019) o estudo com o objetivo da integração de dois protocolos de comunicação na camada de aplicação para dispositivos IoT, o QUIC (INC., 2021) e o MQTT, para prover segurança por meio do protocolo UDP. Ainda assim, diante das dificuldades de documentação do MQTT para novas soluções com o QUIC. Na pesquisa desenvolveu-se APIs (*Application Programming Interface*) agentes, agente-servidor e agente-cliente. Neste trabalho não focou-se em novas implementações para o protocolo MQTT, assim foi observado o funcionamento sem sobrecargas de cabeçalho por causa de mecanismos de segurança, além disso, a comunicação dos sensores IoT com os outros *hosts* da rede foi realizada similarmente ao ambiente real de comunicação.

Outrossim, verificou-se no trabalho de (CIKLABAKKAL et al., 2019) a concepção de um IDS baseado em anomalias com algoritmos de aprendizado de máquina pra criar alertas de comportamento intrusivo no sistema quando este se desvia do comportamento normal aprendido pelos algoritmos. Este trabalho trata-se do aprendizado do comportamento normal e de ataques cibernéticos aos dispositivos IoT, realizando-se, a partir de uma análise experimental, a avaliação dos modelos de ML para um IDS.

Pode-se citar o trabalho de (HINDY et al., 2020b), este é um estudo de caso na área de detecção de anomalias no protocolo MQTT e está sendo utilizado como base para este trabalho. Além disso, realiza a avaliação experimental por meio de um ambiente virtualizado, para avaliação dos experimentos, para tanto, utiliza-se as métricas: acurácia, precisão, revocação e F1-*score*. Ademais, ampliou-se neste estudo a realização de avaliações com o protocolo CoAP, utilizando-se de parâmetros e métricas similares e foi realizado a medição de energia na fase de aprendizado e testes dos algoritmos.

O trabalho de (OLIVEIRA et al., 2020) identificou potenciais ameaças no contexto de segurança IoT, são elas: problemas de confidencialidade (extração de informações dos nós IoT), Integridade (evitar as modificações ou alterações das informações e configuração dos sistemas pelo acesso de usuários não autorizados) e Disponibilidade (Evita a indisponibilidade de um sistema ou dispositivo por usuários autorizados), a energia foi classificada em dois tipos de ameaças, virtual (baseada em Software-Defined Power Meters), físico (utilização de ferramenta para realizar a identificação do consumo de energia) e híbrido (a medição da energia usa a abordagem virtual e física). Desse modo, este trabalho aborda a detecção de potenciais ameaças no contexto IoT e o consumo de energia dos algoritmos de ML.

A falta de dados baseados em IoT para avaliar pesquisas de segurança é observada nos trabalhos de (RAZA; WALLGREN; VOIGT, 2013), (CERVANTES et al., 2015) e (AMARAL et al., 2014). Esses trabalhos tratam sobre proteção de dispositivos IoT e ataques às redes desses smart objects. Sobretudo, utilizaram em seu ambiente de experimentos o simulador Contiki e TinyOS e não criaram nenhum dataset para avaliação das pesquisas (ZARPELÃO et al., 2017). Neste trabalho produziu-se um conjunto de dados sintético para avaliação dos desempenhos dos algoritmos de ML.

A seguir encontra-se a tabela 1 que mostra um resumo dos trabalhos relacionados.

Tabela 1 – Resumo dos Trabalhos Relacionados

Trabalho	MQTT	CoAP	DTLS	AEAD	Seg Energia	Seg QUIC	IDS com ML
(RANDHAWA; HAMEED; MIAN, 2019)		X		X			
(PERETTI; LAKKUNDI; ZORZI, 2015)		X	X				
(RAZA et al., 2013)					X		
(KUMAR; DEZFOULI, 2019)	X					X	
(CIKLABAKKAL et al., 2019)							X
(HINDY et al., 2020b)	X						X
(OLIVEIRA et al., 2020)					X		

1.5 Estrutura do documento

Este trabalho está organizado da seguinte forma, o capítulo 1 é a introdução, mostra também os trabalhos relacionados e quais objetivos foram propostos. O capítulo 2, envolve os conceitos das tecnologias utilizados nesta pesquisa. O capítulo 3, refere-se a metodologia da pesquisa exploratória e aplicada, inicialmente descreve o cenário e a aplicação dos códigos no ambiente controlado , também, descreve as avaliações das ferramentas e recursos testados na pesquisa. Ademais, o capítulo 4 discorre sobre os resultados da comparação entre os desempenhos dos algoritmos de ML e o consumo energético deles sobre os hardwares. Por fim, o capítulo 5 realiza uma análise final sobre todo o trabalho, envolve todas as seções e as conclusões observadas a partir da avaliação do consumo de energia e avaliação de desempenho dos algoritmos de ML, além disso, realiza uma discussão para futuros trabalhos.

2 FUNDAMENTAÇÃO

2.1 Inteligência artificial

A Inteligência artificial (IA) é um conceito abordado em 1956 no escritório de Dartmouth, considerando que os artefatos podem ser capazes de aprender qualquer maneira de realizar uma tarefa específica que seria realizada por humanos. Apesar do aparente tema de ficção, inicialmente, o conceito recebeu atenção da comunidade científica e, nas últimas décadas, com o aumento de recursos computacionais, como processamento, memória volátil e armazenamento, foi possível desenvolver novas abordagens e aplicativos usando computadores.

O conceito de IA é um conceito que é estudado há décadas, seu objetivo é fazer com o que o computador possa desenvolver um aprendizado sobre algo sem a interferência humana, dessa forma a máquina pode realizar tarefas simples ou complexas sem a necessidade de escrever códigos para todas as possibilidades que podem existir diante de um problema. Como também, pode ser vista como um conjunto que contém aprendizado de máquina (ML) e aprendizado profundo (DL) (IZEBOUDJEN; LARBES; FARAH, 2014).

2.2 Aprendizado de máquina

O aprendizado de máquina é um subconjunto de IA, além disso, utiliza-se de dados disponíveis em seu ambiente para aprender com a experiência, e usa-os para melhorar o seu desempenho (LATAH; TOKER, 2018). Dessa forma, classifica-se as abordagens de ML em: aprendizado supervisionado, aprendizado não supervisionado, aprendizado semi-supervisionado e aprendizado por reforço. nas próximas subseções será discutido cada um dessas abordagens. A tabela 2, mostra uma comparação entre as vantagens e desvantagens dessas abordagens.

2.2.1 Aprendizado supervisionado

O modelo supervisionado do ML é definido através de base de conhecimento rotulado, ou seja, etiquetas já com a classe pré-definida e incluída nos dados. Por exemplo, dados extraídos de um tráfego de rede em uma rede com protocolos definidos em sua classificação, por exemplo, identificação e classificação dos protocolos CoAP e HTTP (*Hypertext Transfer Protocol*) em um tráfego de rede. Algoritmos desse tipo precisam dessa informação no conjunto de dados para conseguir prever novas etiquetas/classes em novos dados. Este tipo de algoritmo é utilizado na identificação de protocolos de redes em uma rede baseada em fluxo, outra utilização é no comportamento do mercado financeiro, diagnósticos baseado em imagens médicas, bem como, reconhecimento de imagens em uma câmera de rua (ISI-TICS, 2018). Na aprendizagem

Tabela 2 – Comparação entre as abordagens ML

(LATAH; TOKER, 2018)

Abordagem ML	Vantagens	Desvantagens
Aprendizagem Supervisionado	Aprende com dados rotulados. Generaliza bem com base em um conjunto de dados suficiente.	Requer um conjunto de dados que representa o sistema. Os dados são rotulados manualmente por especialistas humanos, o que não é apropriado para muitas aplicações do mundo real
Aprendizagem Não-supervisionado	Localiza padrões ocultos sem depender de dados rotulados. Desempenho melhor para dados invisíveis em comparação com a abordagem supervisionada.	Pode não fornecer uma visão útil dos padrões ocultos e do que eles realmente significam.
Aprendizagem Semi-supervisionado	Aprende com dados rotulados e não rotulados. Certas suposições sobre a distribuição de dados subjacente devem ser atendidas.	Pode levar a um desempenho pior quando escolhemos suposições erradas.
Aprendizagem por reforço	Adaptação dinâmica e aprimoramento gradual. Um agente interage com um ambiente incerto, no qual o objetivo é maximizar a recompensa do agente. Também pode ser usado para problemas difíceis que não possuem formulação analítica.	Há uma troca entre exploração e exploração. Além disso, precisamos especificar uma função de recompensa, política parametrizada, estratégia e política inicial.

supervisionada, há representação entre os dados de entrada e os dados de saída. Como também, divide-se em duas fases, a fase de treinamento e a fase de teste (RAIKAR et al., 2020).

2.2.2 Aprendizado não supervisionado

O aprendizado não supervisionado não faz o uso de etiquetas. O modelo tenta descobrir os padrões através de algoritmos de recomendação, agrupamento e segmentação. Dessa forma, desenvolve a predição, faz o uso de atributos, notas e características em geral dos dados. Esses tipos de algoritmos tem algumas utilizações no campo da visualização de dados, recomendação de itens, segmentação de clientes em um sistema ou até mesmo envolvido problemas complexos, os quais, precisam de exclusão de atributos em problemas complexos (ISI-TICS, 2018).

2.2.3 Aprendizado semi-supervisionado

O aprendizado semi supervisionado utiliza os exemplos rotulados para se obter informações sobre o problema e aproveitá-las para guiar o processo de aprendizado, a partir de exemplos não rotulados (SANCHES, 2003). A ideia é rotular os dados, com uma certa margem de segurança, alguns dos exemplos não rotulados, os quais são utilizados durante a fase de treinamento do classificador.

2.2.4 Aprendizado por reforço

É um modelo adotado em sistemas em tempo real. A desvantagem desse modelo é que não existe um conjunto de dados históricos para treinamento, por consequência, não é possível criar um modelo de aprendizado. O aprendizado é feito com a chegada de novos dados. É um modelo adotado por exemplo para decisões que precisam ser tomadas por robôs que necessitam trafegar em ambientes ainda não explorados, o robô vai aprender por força bruta ao executar cada passo, o próximo agente da NN (*Neural Network*) recebe o resultado do agente anterior para tomar a próxima decisão (ISI-TICS, 2018).

A aprendizagem por reforço permite que um sistema ou agente aprenda por experiências anteriores, observando os resultados dessas interações. A interação permite que o sistema se comporte como um humano ou animal aprendendo. Conceitualmente, o aprendizado por reforço escolhe ações que maximizem ações bem sucedidas no futuro, os resultados adquirido por um agente influenciam na condição de um agente futuro. Na DL (*Deep Learning*), o agente do aprendizado por reforço ignora o aprendizado completamente e aprende participando diretamente da atividade que foi especificada (ABIODUN et al., 2018).

2.2.5 Algoritmos de Classificação

Os algoritmos de classificação encontram-se no grupo de aprendizado supervisionado (LUXBURG; SCHÖLKOPF, 2011), tipicamente, estes são denominados conjuntos de treinamento. Desse modo, O problema da classificação refere-se a uma entrada X_i e uma classe qualitativa desconhecida Y_i ε {1,2,...,L}, dessa maneira, assume-se geralmente que L < ∞. Logo, um algoritmo de classificação produzirá um classificador capaz de generalizar os dados, com o objetivo de classificá-los, posteriormente, objetos com rótulo desconhecido.

2.2.5.1 Regressão Logística (Logistic Regression)

A regressão logística foi proposta no final da década de 1960 e início dos anos de 1970, como uma proposta alternativa, e tornou-se disponível nos pacotes estatísticos no início dos anos 1980. Além disso, é um modelo que usa a função *logit*, o logaritmo natural de uma razão probabilística (PENG; LEE; INGERSOLL, 2002), O algoritmo de regressão logística é usado para lidar com problemas de classificação. Os seus resultados são binomiais, por exemplo, pode prever se um e-mail que acabou de chegar na caixa de entrada é spam ou não. No entanto, é utilizado também para problemas multiclasses, ou seja, analisa diferentes elementos ou conjunto de variáveis para determinar qual a classe do objeto em questão, por meio de estimativas probabilísticas, usando uma função logística.

2.2.5.2 K-Vizinhos mais próximos (KNN)

O algoritmo KNN (*K–Nearest Neighbours*) é um modelo de classificação (ALTMAN, 1992), que utiliza o algoritmo LlazyL. Este pula a etapa de treinamento, realizando a relação

entre a variável K e o seu vizinho mais próximo. Assim, considera-se como entrada o valor da variável. Logo, a partir dessa informação o algoritmo também considera os votos obtidos de outros vizinhos, para identificar se o próximo vizinho faz ou não parte do grupo ou classe. Ademais, é considerado um algoritmo a ser utilizado em pequenos volumes de dados, pois tem uma fase de treinamento considerada rápida, porém os testes de validação são lentos. Dessa forma, ele utiliza dois valores de entrada, sendo eles, o valor para variável k e os votos dos vizinhos.

2.2.5.3 Gaussian Naive Bayes

O algoritmo *Gaussian Naive Bayes* considera todas as probabilidades relevantes conhecidas. Em sua teoria de decisão, *Bayes* escolhe os melhores rótulos de classe, baseando-se neles e nas perdas identificadas incorretamente. Ademais, pode-se determinar a classificação do tráfego, ou seja, se é proveniente de um ataque, ou tráfego normal. Por exemplo, se houver dois rótulos de classes presentes possíveis em $\gamma = \{C_1, C_2\}$, onde C_1, representa o primeiro rótulo (tráfego normal) e C_2, representa o rótulo denominado ataque (ZHANG et al., 2018). Assim, o teorema de *Bayes* é representado por meio da seguinte fórmula:

$$P(c|x) = \frac{P(c)P(x)}{P(x)} \tag{1}$$

No teorema representado pela fórmula 1 é apresentado um tipo de probabilidade *a priori*, P(x | c), que é classificado como uma probabilidade condicional de um registro de conexão x em relação ao rótulo de classe c, e P(x) representa um modelo usado para normalização. Portanto, é uma das formas de predição utilizando classificação para a área de segurança de informação.

2.2.5.4 Árvore de decisão - Decision Tree

O algoritmo de árvore de decisão é uma das técnicas conhecidas e com grande utilização na área de aprendizado de máquina. Ademais, a árvore é feita de nós de decisão e nós folha, estes, por sua vez, representam o resultado da decisão de uma classe. Assim, o resultado corresponde a um teste sobre um único atributo dos dados de entrada e também é composto por número de ramos, o qual lida com um resultado do teste.

A construção de uma árvore de decisão é baseada em um processo de divisão e conquista (ISHIDA; TAKAKURA; OKABE, 2011). Além disso, também, é formada por um conjunto T de dados de treinamento que consiste em k classes $(C_1, C_2, ..., C_k)$, isto é, T consiste em classes de um mesmo grupo, denominado de folha. Além do mais, por exemplo, se T é constituído em dados de várias casos, denominado de classes mistas, é realizado um teste baseado em alguns atributos a_i, a partir disso, será transformado em subconjuntos $(T_1, T_2, ...T_n)$, onde n é o número de resultados do teste sobre o atributo a_i. Então, esse processo repete-se recursivamente sobre

cada T_j, onde $1 \leq j \leq n$. Caso T consista em casos de mesma classe, considerado pelo algoritmo como uma folha. Assim sendo, se T não conter nenhum caso, será o principal nó de seu nó pai. Na figura 2 observa-se um exemplo do modelo da árvore de decisão.

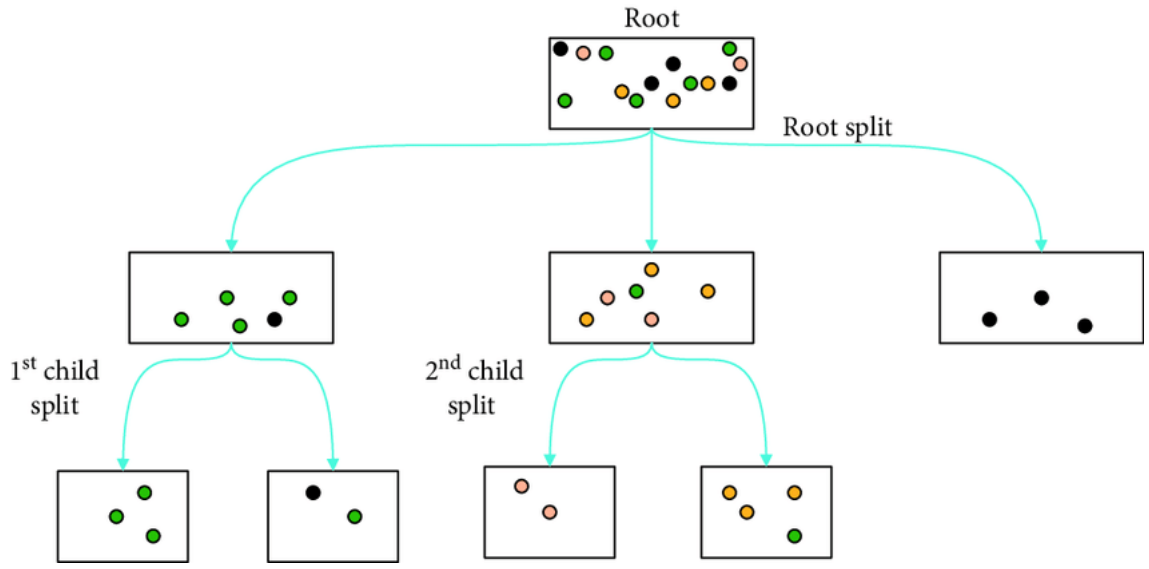


Figura 2 – Exemplo de uma Árvore de decisão

fonte:(HUSSAIN et al., 2020)

2.2.5.5 Florestas Aleatórias - Random Forests

Segundo (FARNAAZ; JABBAR, 2016), o modelo de florestas aleatórias é um tipo de classificador de conjunto elaborado para otimizar a precisão, apresenta baixo erro de categorização se comparado com outros algoritmos de classificação. Além disso, o modelo é formado por um conjunto de classificadores composto por árvores, sendo assim, cada uma delas é cultivada de acordo com um vetor aleatório, e, consequentemente, as árvores ficam idênticas. Ademais, a diversidade dessa floresta é formada por um conjunto de atributos do conjunto de dados, ou pode ser obtido pela modificação de alguns parâmetros da árvore de decisão de forma arbitrária (PRIMARTHA; TAMA, 2017). Enfim, existem dois parâmetros, que permitem ajustes, no modelo de florestas aleatórias: o número de variáveis a serem escolhidas em cada nó e o número de árvores que constroem a floresta (PRIMARTHA; TAMA, 2017). A figura 3, apresenta um exemplo do modelo de florestas aleatórias.

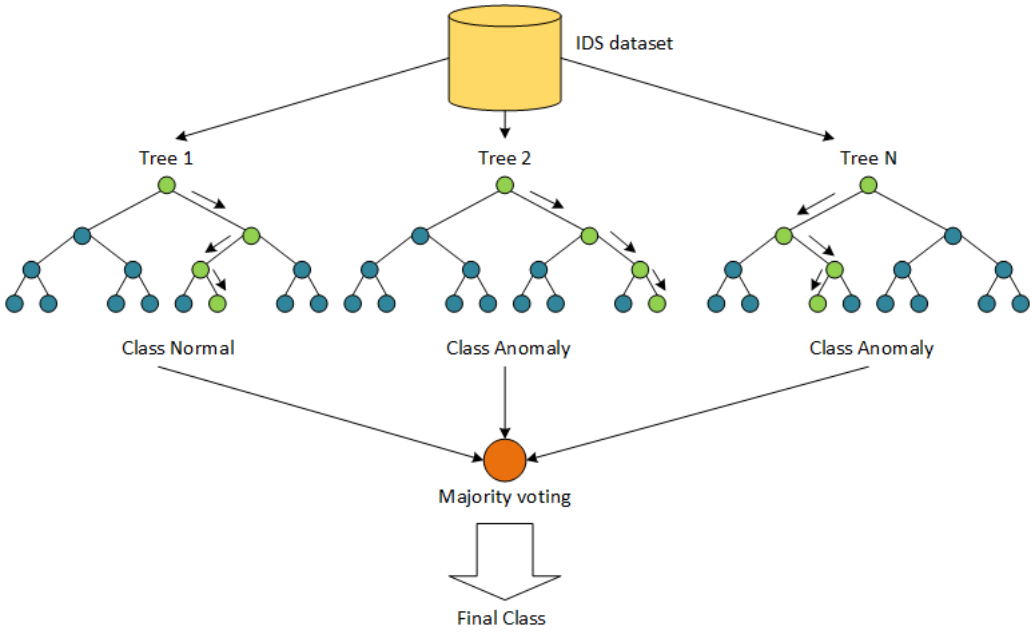


Figura 3 – Exemplo do modelo de árvores aleatórias

fonte:(PRIMARTHA; TAMA, 2017)

2.2.5.6 Máquina de Vetor de Suporte - Support Vector Machine (linear and RBF kernel)

A máquina de vetor de suporte (SVM) é um algoritmo de classificação que analisa dados para estabelecimento de padrões. Inicialmente, foi desenvolvido como um classificador linear, pois, aprende os resultados com poucos exemplos de treinamento, com isso, produz uma classificação precisa e reduz o índice de resultados classificados como falsos positivos. Dessa forma, para obter esses resultados ele sempre encontra uma ótima solução global por meio de separação linear em um hiperplano fracionando as duas classes (KOKILA; SELVI; GOVINDARAJAN, 2014), a figura 4 mostra o SVM com os agrupamentos de dados e as linhas de margem separam as classes.

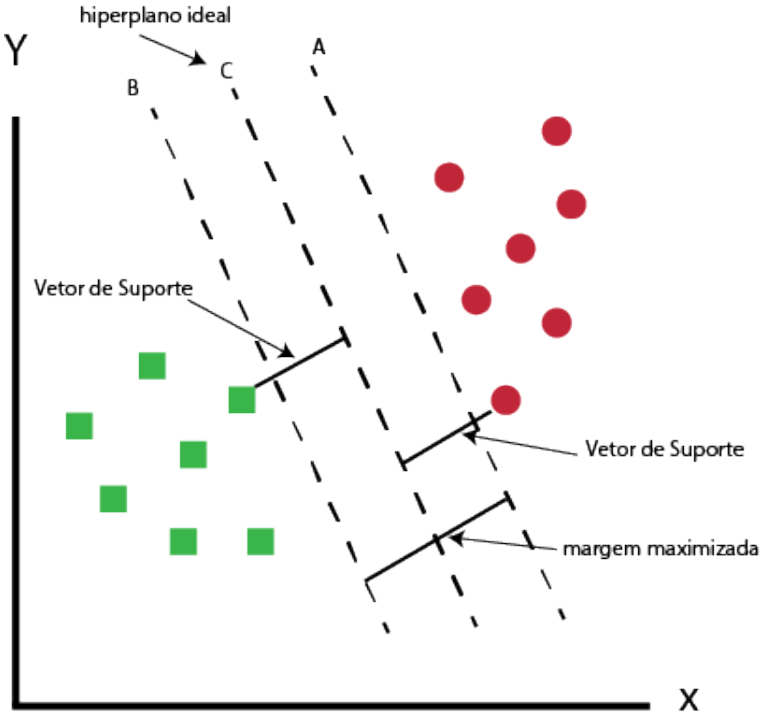


Figura 4 – Exemplo do modelo SVM Linear

fonte:(CAVALCANTI, 2021)

Na figura 4, observa-se que o algoritmo encontra a linha do hiperplano ideal que separa as duas regiões. As linhas são separadores marginais que estão à frente de suas respectivas regiões e os vetores de suporte são determinados pelas distâncias deles dos pontos próximos da linha de separação. Portanto, a ação de aprendizagem do SVM Linear consiste na determinação dos pontos do vetor de suporte e a distância de margem que separa as regiões, após o processo de aprendizagem os outros pontos sem suporte não são usados para previsão.

O SVM foi baseado em uma divisão linear, porém nem todos os dados podem ser divididos linearmente (YANG; LI; YANG, 2015). Dessa forma, a utilização do RBF (*Radial Basis Function*) é usado para mapeamento não linear da entrada de recursos de dimensão superior, também têm menos dificuldades ao tratar dados numéricos, porque os valores ficam entre zero e um, entretanto os valores do *kernel* polinomial podem ir de zero ao infinito (HEBA et al., 2010). Dessa forma, observando-se a quantidade de hiperparâmetros do *kernel* polinomial a mais do que o *kernel* RBF, este é usado como a função de *kernel* padrão (HEBA et al., 2010). A figura 5 mostra um resultado a partir da aplicação do algoritmo SVM com *Kernel* RBF.

A figura 5 mostra um aglomerado dividido em *clusters* representados em cores. Assim, os pontos inseridos nas cores representam as instâncias das classes e as cores dos fundos dos *clusters* os identificam. (JUNIOR, 2010).

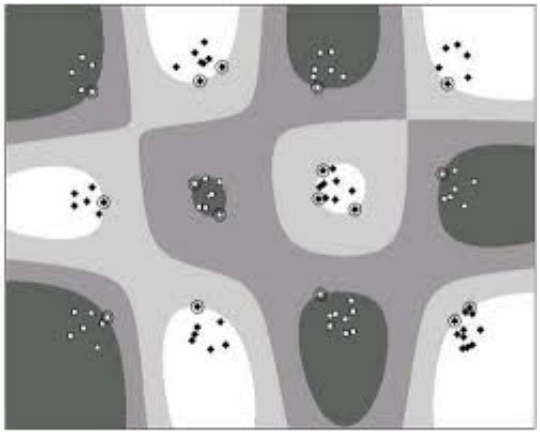


Figura 5 – Exemplo do modelo SVM Kernel RBF

fonte:(JUNIOR, 2010)

2.3 Validação Cruzada

A validação cruzada é um método útil para os modelos de aprendizado de máquina supervisionado com o objetivo de evitar o *overfiting*, ou seja, um modelo que apenas repete os rótulos das amostras que foram visualizados pelo algoritmo recentemente (CROSS-VALIDATION, 2020). Nesse modelo, os dados são separados em dois tipos: conjunto de aprendizado e conjunto de testes. A seguir está a figura 6 que representa o fluxograma desse modelo:

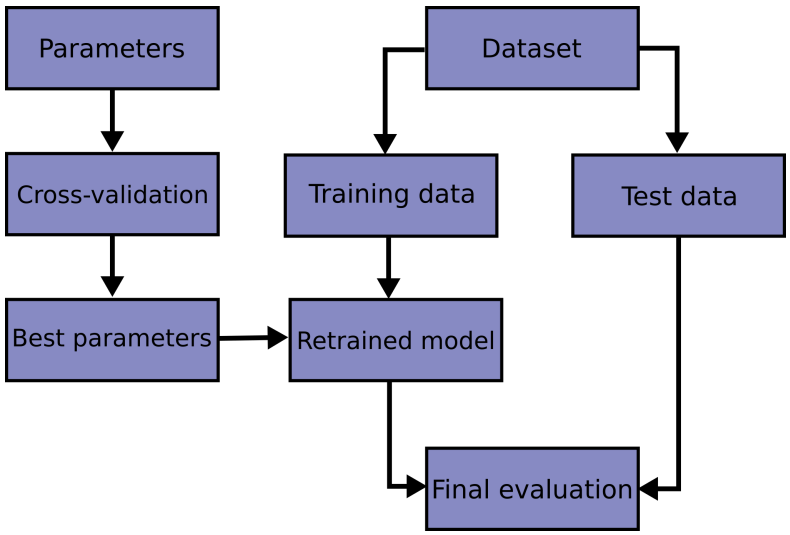


Figura 6 – Fluxograma do modelo de Validação Cruzada

Fonte: (CROSS-VALIDATION, 2020)

Na figura 6, observa-se diferentes hiperparâmetros que são utilizados para melhorar os modelos de aprendizagem de máquina supervisionada. Eles podem representar um problema se não houver uma correta parametrização. No entanto, para resolver esse problema é utilizada uma técnica denominada *k-fold cross-validation*, a qual divide-se o conjunto em k conjuntos menores, aleatórios e de tamanho aproximadamente iguais. Esses conjuntos são denominados dobras

(*folds*), deste modo, um desses conjuntos da partilha (k-1) é escolhido no conjunto durante esse processo para que posteriormente seja utilizado para testar o modelo (WONG; YANG, 2017). A figura 7 a seguir demonstrar o processo *k-fold cross-validation*.

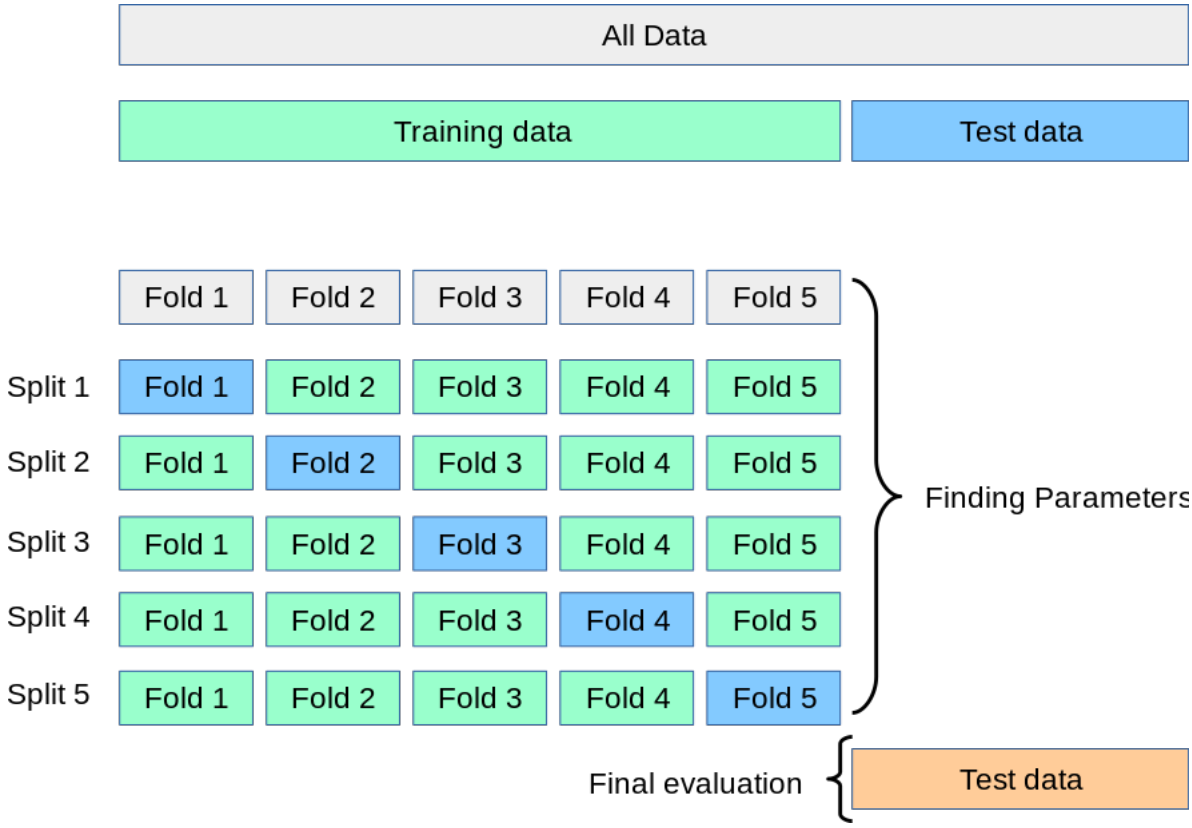


Figura 7 – *K-fold Cross-Validation*

Fonte: (CROSS-VALIDATION, 2020)

2.4 Internet das Coisas

A popularidade dos dispositivos da Internet das Coisas (IoT) está aumentando cada vez mais para diferentes domínios de uso. Em uma escala globalmente abrangente, os dispositivos são usados para resolver vários problemas. Alguns exemplos de campos de uso da IoT são, a utilização para o campos da logística e assistência médica (BASSI et al., 2013), na logística, o equipamento IoT atua para controle de estoque, monitoramento de carga e monitoramento do veículo de transporte. No campo da saúde, os dispositivos IoT estão sendo usado para monitorar os sinais vitais do paciente, como temperatura, porcentagem de saturação de oxigênio e taxa de pulso (KRISHNA; SAMPATH, 2017).

As aplicações IoT atualmente são utilizadas nas pesquisas militares e civis, incluindo serviços de detecção de intrusão, vigilância monitorada, monitoramento de processos industriais e controle. O conceito de IoT nasceu da ideia de interconexão entre objetos, o que era possível somente através de equipamentos e informática bem maiores como computadores, notebooks, servidores e grandes equipamentos de rede. Esses equipamentos podem ser acessíveis dentro de uma estrutura global, modular e totalmente conectada. É típico que os equipamentos em uma rede IoT são alimentados por bateria

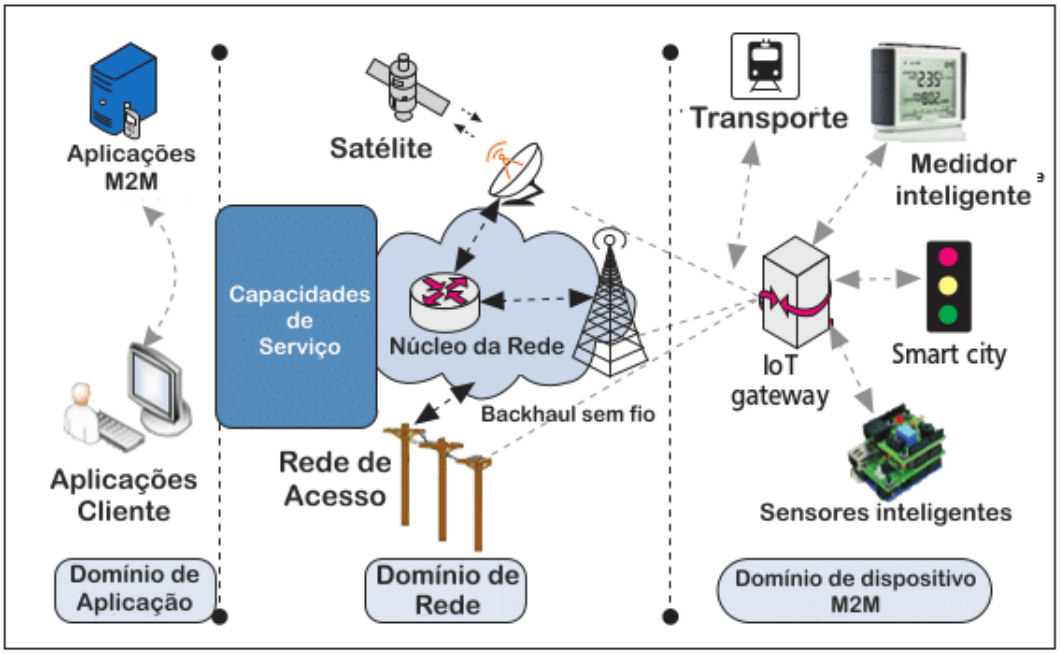


Figura 8 – Arquitetura IoT

Fonte: (SILVA, 2018)

As ameaças nas redes têm acompanhado o crescimento e desenvolvimento de dispositivos inteligentes que se interligam entre eles, atualmente existem muitas vulnerabilidades nos dispositivos de rede e consequentemente cresce o número de ameaças, somando-se a facilidade de instalação dos equipamentos IoT e a despreocupação dos profissionais de TI (Tecnologia da Informação) na gerência desses dispositivos, o cenário se estabelece como preocupante. As

ameaças são classificadas como críticas, o que pode impossibilitar o funcionamento correto ou mesmo inviabilizar o funcionamento de sistemas computacionais baseados em IoT. O estudo dessas ameaças especificamente em redes de dispositivos IoT possibilita um entendimento de como as tecnologias tem evoluído para tratar incidentes de segurança, quais eventos anômalos ocorrem com mais frequência em ambientes que informações são trocadas entre equipamentos.

Uma rede de dispositivos IoT geralmente agregam uma grande quantidade de equipamentos heterogêneos, por exemplo, câmeras, eletrodomésticos inteligentes, sensores de saúde, dentre outros. Diante disso, o trabalho de (HALDER; GHOSAL; CONTI, 2019) classifica o padrão de coleta de dados em orientado a eventos e orientado a tempo. Quando os sensores geram informações pelo tempo, cada nó é responsável pela geração de informações a uma taxa predeterminada, posteriormente envia esses dados ao nó coletor. Nos nós que são orientados a eventos, as informações são enviadas ao nó coletor na ocorrência de algum evento. Para as rede de dispositivos IoT a coleta de dados é o mais adequado para detecção de intrusão.

Para detectar uma intrusão é considerado o posicionamento do IDS dentro da rede, o IDS tem papel fundamental na detecção de intrusão e na velocidade de resposta de incidentes na rede (CONTI, 2015). Para detecção de invasor na rede é necessário uma cobertura bem ampla de dispositivos em uma rede. Alguns sistemas de segurança são compostos de vários mecanismos, entretanto um sistema desses não econômico e nem prático, aumenta a quantidade de configurações a serem realizadas, o aumento do custo é um fator principal. Portanto, um sistema de segurança estratégico, econômico e com custo reduzido é o desejável para aplicações de detecção de intrusão em uma rede IoT.

2.4.1 Protocolo MQTT

MQ *Telemetry Protocol* (MQTT) é um protocolo de mensagens usando mecanismo de publicação/assinatura, desenvolvido sob o padrão OASIS (*Organization for the Advancement of Structured Information Standards*). Inclusive, é um protocolo amplamente utilizado atualmente por causa de seus recursos limitados, pois é leve e utiliza pouca largura de banda, além de ser aberto e simples de ser implementado (ANDY; RAHARDJO; HANINDHITO, 2017).

Além disso, sua arquitetura assemelha-se ao modelo cliente e servidor. Ademais, contém um nó principal concentrador de informações denominada *broker*, responsável pela coleta das informações dos sensores IoT. Ou seja, é essencial em todo o processo de comunicação entre clientes externos ao ambiente MQTT e os *smart objects* (PRADA et al., 2016). Ademais, todos os nós envolvidos na comunicação podem assumir o papel de assinante ou publicador. Em suma, todos os nós sensores do MQTT podem publicar periodicamente seus dados de medições para um endereço de tópico. Dessa forma, cada dispositivo que se registra como assinante recebe uma mensagem do *broker* sempre que o tópico é atualizado. Na figura 9 a seguir observa-se a arquitetura do protocolo MQTT.

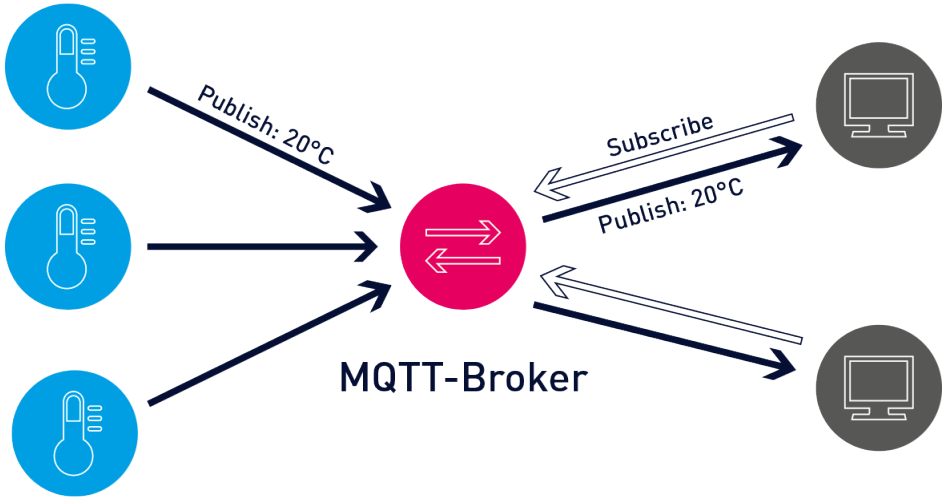


Figura 9 – Arquitetura MQTT

Fonte: (AG, 2021)

Observa-se na figura 9 os clientes enviando solicitações por meio do método *subscribe* (assinatura) para o MQTT *Broker*. Além disso, os sensores publicam a informação de temperatura (20º C) utilizando-se do MQTT *Broker*, assim, este, por sua vez, envia as informações aos clientes que estão escritos no tópico.

Ainda, na arquitetura MQTT, há a troca de mensagens entre os elementos da arquitetura para envio e recebimento de mensagens confiáveis. O MQTT é implementado na camada de aplicação do modelo TCP/IP sobre o protocolo da camada de transporte TCP. Essas mensagens do MQTT estão descritas na tabela 3 (BANKS ANDREW; GUPTA, 2015) a seguir.

Tabela 3 – Tabela de Mensagens do MQTT

Valor	Nome	Direção do Fluxo da Mensagem	Descrição
0	Reservado	Cliente para Servidor	Requisição do cliente para conectar ao servidor
1	CONNECT	Cliente para Servidor	Requisição do cliente para conectar ao servidor
2	CONNACK	Servidor para Cliente	Reconhecimento da conexão
3	PUBLISH	Cliente para Servidor ou Servidor para cliente	Publicar mensagem
4	PUBACK	Cliente para Servidor ou Servidor para cliente	Reconhecimento da publicação
5	PUBREC	Publicação recebida	Publicação recebida (entrega garantida parte 1)
6	PUBREL	Cliente para Servidor ou Servidor para cliente	Publicação lançada (entrega garantida parte 2)
7	PUBCOMP	Cliente para Servidor ou Servidor para cliente	Publicação completa (entrega garantida parte 3)
8	SUBSCRIBE	Cliente para Servidor	Pedido de inscrição do cliente
9	SUBACK	Servidor para cliente	Reconhecimento de inscrição do servidor
10	UNSUBSCRIBE	Cliente para servidor	Cancelar subscrição
11	UNSUBACK	Servidor para cliente	Reconhecimento do cancelamento de subscrição
12	PINGREQ	Requisição	Requisição PING
13	PINGRESP	Servidor para cliente	Resposta PING
14	DISCONNECT	Cliente para servidor	Cliente está desconectado
15	Reservado	Proibido	Reservado

2.4.1.1 Fluxo Unidirecional do MQTT

O fluxo unidirecional do MQTT é caracterizado pelo envio das mensagens: CONNECT, PUBLISH, SUBSCRIBE, UNSUBSCRIBE e PINGREQ. Essas mensagens são enviadas pelo cliente para realizar as requisições para estabelecer conexão, realizar a publicação da mensagem no *Broker*, pedido de inscrição no *Broker*, desfazer a inscrição no *Broker* e garantir a manutenção da conexão entre servidor e cliente. Desse modo, não haverá as mensagens de confirmações. Outrossim, esse fluxo é caracterizado também pelo envio da mensagem de qualquer um dos elementos da arquitetura MQTT designado como cliente ou servidor.

2.4.1.2 Fluxo Bidirecional do MQTT

O fluxo bidirecional do MQTT é caracterizado pelo envio e resposta das mensagens MQTT, por exemplo, um cliente envia uma mensagem CONNECT para o servidor e o servidor responde com uma mensagem CONNACK de reconhecimento da conexão.

2.4.2 CoAP (*Constrained Application Protocol*)

O protocolo CoAP é utilizado por equipamentos da arquitetura M2M (*Machine-to-Machine*). Esses equipamentos fazem uso de baixo consumo de energia e baixa potência. Os nós, de rede, que fazem o uso do protocolo são compostos por microcontroladores e suportam poucas instruções de uma só vez,e, também, tem pouca memória RAM (*Random Access Memory*) e ROM (*Read-Only Memory*). Além disso, tem altas taxas de erros de transmissão. Ademais, esses equipamentos podem ser utilizados em grupo (rede local) ou separados individualmente.

Outrossim, o protocolo CoAP é utilizado sob o modelo de arquitetura cliente e servidor, ou seja, é um tipo de abordagem que tem como característica o envio de solicitações de forma assíncrona e recebimento das respostas dos nós servidores (sensores) de forma também assíncrona. Além disso, é possível integrar o CoAP ao protocolo HTTP, no entanto, há outros recursos para realizar operações de trocas de mensagens em rede, por exemplo, o suporte ao *multicast*, e também suporta consulta por URIs (*Uniform Resource Identifier*), semelhante ao protocolo HTTP e suporta as mídias da Internet.

Ademais, a garantia de entrega de mensagens do CoAP e a sua confiabilidade é feita pela troca de mensagens com alguns métodos de solicitação e resposta. Além disso, o protocolo é classificado relacionando-se com a camada de aplicação segundo o modelo de arquitetura TCP/IP, mas pode-se realizar uma abstração em duas camadas para compreender o funcionamento do protocolo, como visualizado na figura 10, onde pode-se observar essa abstração.

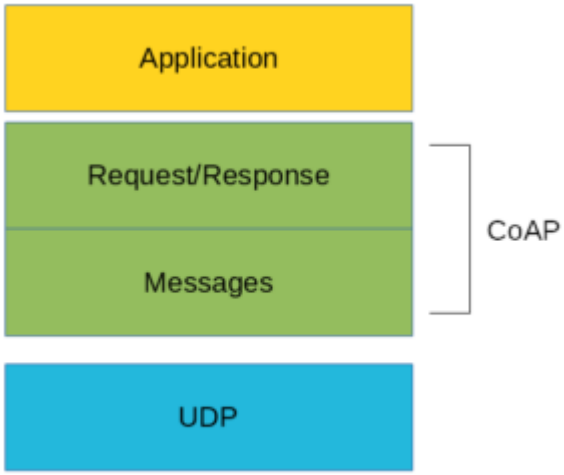


Figura 10 – Camadas Protocolo CoAP

Fonte: (AZZOLA, 2019)

Na figura 10 é observado que o protocolo CoAP está estruturado sob duas camadas: transporte com a utilização do protocolo UDP e a camada de aplicação abstraindo as camadas de mensagens, denominadas requisição e resposta. Portanto, pode-se dizer que o protocolo é composto, basicamente, de duas camadas. Ademais, ele suporta quatro tipos de mensagens: *CON*

(Confirmable), *NON (Non−confirmable)*, *ACK (Acknowledgment)* e *Reset*.

As mensagens do tipo CON são utilizadas para solicitações e precisam ser confirmadas com uma mensagem do tipo ACK. Já as mensagens do tipo NON não necessitam de uma confirmação de entrega, pois é utilizada em requisições repetidas para um sensor. A mensagem do tipo ACK confirma que uma outra chegou ao destino. Além disso, a mensagem reset informa ao nó de origem que a mensagem foi recebida, porém não pode ser processada adequadamente pela falta de algum contexto, utiliza-se esse tipo de mensagem para testar a comunicação com um ping CoAP (SHELBY; HARTKE; BORMANN, 2014).

2.4.2.1 Fluxo Unidirecional do CoAP

O fluxo unidirecional do CoAP é caracterizado por mensagens do tipo CON não confirmadas pelo servidor. Além dessa, há as mensagens do tipo NON e RST, as quais não precisam de confirmação de entrega no destino. Além disso, esse fluxo é caracterizado também pelo envio da mensagem de qualquer um dos elementos da arquitetura CoAP como cliente ou servidor.

2.4.2.2 Fluxo Bidirecional do CoAP

O fluxo bidirecional no CoAP é caracterizado por duas mensagens. A mensagem do tipo CON é enviada pelo cliente, ou seja, é um tipo de mensagem que pede a confirmação de entrega. Já a outra mensagem é a ACK, que é enviada pelo servidor realizando a confirmação da mensagem recebida

2.4.2.3 Multicast CoAP

O recurso do *multicast*, como serviço de difusão, entrega pacotes para um ou mais grupos de hosts em um domínio de rede (KUROSE; ROSS, 2012). Algumas aplicações utilizam-se dele para enviar informações a um pequeno grupo de nós que compartilham seus dados, entre essas aplicações estão a transferência de dados em grandes volumes, aplicações de fluxo de mídia, as aplicações de dados compartilhados, a atualização de *cache* da Web e jogos interativos. Ao verificar as aplicações observam-se dois problemas. O primeiro refere-se a entrega da informação entre vários nós da rede e o outro refere-se à coexistência de vários endereços IP no mesmo dispositivo de rede. Dessa forma, alguns protocolos surgiram para permitir que os hosts determinados para participar de um grupo *multicast* possam se comunicar, a partir daí, a figura 11 demonstra esse recurso.

Ainda mais, uma classe de endereços IP foi destinada exclusivamente para a implementação dessa comunicação entre os participantes dos grupos *multicast*, esse endereço foi denominado pela classe D, o qual começa em 224.0.0.0 e vai até 239.255.255.255. Assim sendo, os endereços IP atribuídos para *multicast* podem ser compartilhados com os mesmos endereços IP de identificação do *host* na rede na mesma *interface*.

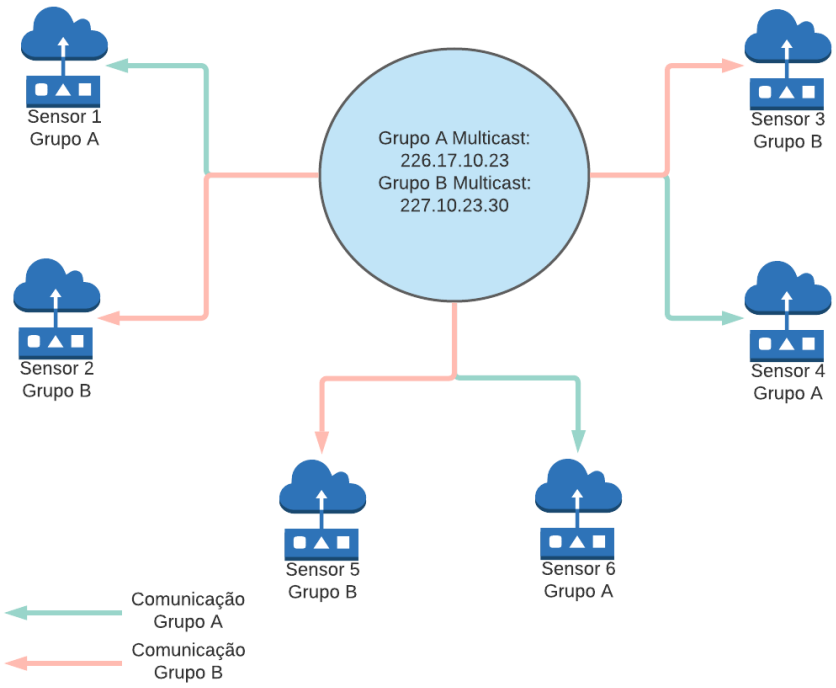


Figura 11 – Multicast CoAP

Fonte: Elaborado pelo autor (2021)

No diagrama representado pela figura 11, observa-se dois grupos e alguns sensores IoT. Nos grupos A e B a identificação é feita por meio do endereço IP de classe D, designado para esse fim. Ao primeiro grupo foi atribuído o endereço 226.17.10.23, e para o outro grupo foi atribuído o endereço 227.10.23.30. Dessa forma, quando um participante do grupo A desejar se comunicar com os hosts que pertencem ao seu grupo, enviará a informação para o endereço de destino 226.17.10.23, assim, todos que pertencem ao respectivo agrupamento receberão o pacote contendo os dados.

No CoAP esse recurso é utilizado para realizar descoberta de serviços, ou seja, é habilitado nos dispositivos finais da comunicação, para que os nós sejam capazes de escutar as solicitações (SHELBY; HARTKE; BORMANN, 2014) *multicast*. Assim sendo, um dispositivo IoT habilitado com CoAP pode estar preparado para ouvir solicitações por meio de endereços *multicast*, também, pode ser configurado com mais de um endereço *multicast*, pois poderá fazer parte de mais de um grupo, assim, a porta do serviço tem que estar habilitada. No CoAP a porta padrão é 5683 sob o protocolo UDP.

2.4.2.4 Proxy CoAP

O *proxy* CoAP é um host na extremidade do cliente CoAP que realiza as solicitações dos clientes aos servidores, para realizar essas requisições assumirá o papel de um cliente. Além disso, é suportado pelos dispositivos finais para atender às solicitações de um ou mais clientes

(SHELBY; HARTKE; BORMANN, 2014). Ao usar o *proxy*, a URI (*Unique Resource Identifier*) é informada na solicitação, enquanto o cliente utiliza o endereço de destino o IP do servidor *proxy* para realizar a solicitação, isso está demonstrado na figura 12, que exemplifica essa comunicação entre cliente e sensor CoAP por meio de *proxy*.

Ainda, ele é útil em redes internas, limita o tráfego de rede, melhora o desempenho dela e atende aos requisitos de segurança. Por exemplo, suporta requisições aos servidores CoAP por meio do protocolo COAPS (*Constrained Application Protocol Security*). Além disso, é suportado pelos dispositivos finais para atender às solicitações de um ou mais clientes (SHELBY; HARTKE; BORMANN, 2014).

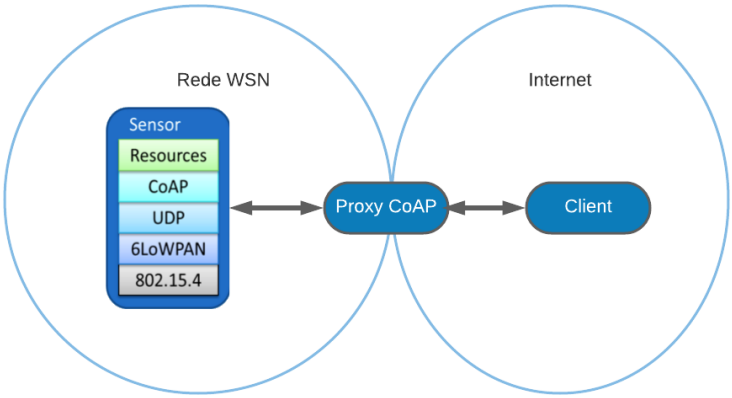


Figura 12 – Proxy CoAP

Fonte: Adaptado de (DAMODARAN, 2015)

2.4.2.5 Ferramenta CoAP SHELL

O COAP Shell[1] é uma ferramenta de linha de comando interativa, ela fornece recursos para solicitar informações de servidores habilitados com o protocolo CoAP, também suporta interações com o CoAP sobre o protocolo UDP e CoAPS, o qual habilita segurança sob o protocolo DTLS. Além disso, é desenvolvido sobre os projetos Spring Shell, *Californium* (Cf) e *Scandium* (Sc), também é um aplicativo do tipo jar autoexecutável e para funcionar é necessário que o ambiente Java, versão 8 ou superior. Ademais, a figura 13 mostra a saída do comando *connect* dentro da ferramenta.

[1] https://github.com/tzolov/coap-shell

Figura 13 – Ferramenta CoAP Shell

Fonte: (TZOLOV, 2018)

2.4.2.6 Quick CoAP Multicast Discovery

Esta ferramenta foi desenvolvida sob o framework Qt. Ademais, a função dela é a descoberta de dispositivos habilitados com recurso de difusão seletiva (*multicast*) e que estão escutando na porta 5683 do protocolo UDP. Outrossim, necessita da biblioteca Qt instalada, para a interface KDE (*K Desktop Environment*) no sistema operacional Linux, ou seja, utiliza uma interface gráfica com a aparência dos programas do KDE, a figura 14 demonstra a aplicação fazendo a busca ao *multicast* por meio de um endereço URI.

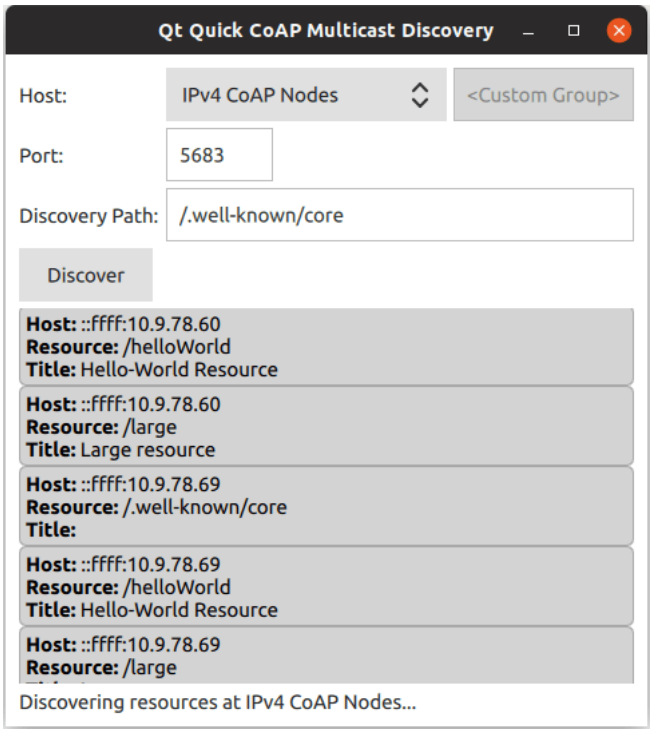


Figura 14 – Ferramenta Quick Multicast Discovery

Fonte: (QTCOM, 2017)

2.4.2.7 Biblioteca AIOCoAP

O pacote aiocoap é escrito em uma biblioteca da linguagem Python para interagir com o protocolo CoAP. Além disso, é disponibilizado uma documentação detalhada da API, suas funções, parâmetros e métodos. Dessa forma, é possível realizar a implementação das principais funcionalidades de um cliente, servidor e *proxy* habilitados para o protocolo CoAP (WASILAK MACIEJ;AMSüSS, 2014).

2.5 Segurança IoT

A segurança dos dispositivos é uma preocupação contemporânea e seu baixo espaço de armazenamento não permite atualizações de software, por isso, em muitos casos o dispositivo encontra-se desatualizado. Além disso, a configuração padrão dos dispositivos não é uma preocupação dos usuários, pois muitos dispositivos encontram-se com os perfis de acesso com padrão de fábrica, podendo ser facilmente acessados remotamente.

As redes de dispositivos IoT contém vários *Smart Objects* interligados que podem ou não trocar informações entre si. Nessa classificação, encontram-se os *Smart Health, Smart Laptop, Smart Cars, Smart Individual* e *Smart Home* (GURUNATH et al., 2018). Assim, esses dispositivos utilizam protocolos de comunicação para trocar informações com as aplicações entre eles. Desse modo, muitos desses protocolos não estão configurados com uma camada de segurança. Portanto, apresentam riscos em toda a rede pois um atacante pode apoderar-se do dispositivo para controlar toda a rede.

(PESCATORE; SHPANTZER, 2014) identificou preocupações dos riscos de segurança dos dispositivos IoT, foram identificados que há o mesmo nível de problemas de segurança, se comparado a outras tecnologias. Ainda sim, observou-se que o desenvolvimento de políticas, configuração, gerenciamento de vulnerabilidades, coleta de dados e visualização de desempenho são os novos desafios de segurança para os dispositivos IoT.

Os conceitos de segurança com o paradigma IoT são relacionados e a seguir são apresentadas as bases conceituais da segurança relacionando-as com o paradigma IoT:

- **Confidencialidade**: Garante que somente a origem e destino manipulam a informação. No ambiente IoT, o problema do princípio da confidencialidade nas redes de dispositivos IoT é a autorização dada aos dispositivos para manipular informações (MIORANDI et al., 2012). Além disso, deve-se definir um controle de acesso e autenticação para consultas de informações aos sensores dos dispositivos para garantir um acesso seguro e evitar que informações sejam manipuladas por terceiros. Portanto, a existência de atacantes nas redes de dispositivos com acesso e autenticados representa uma grande falha para esses dispositivos.

- **Integridade**: Assegura que o conteúdo da mensagem não seja modificado entre a origem e destino da mensagem (KUROSE; ROSS, 2012). Ademais, nas redes de dispositivos IoT esse problema tornou-se central, pois o número de dispositivos e usuários é incontável e a necessidade de prover segurança nesse aspecto é um desafio. Assim sendo, um usuário pode realizar alterações nas informações sem que os dispositivos detectem-nos. Além disso, a identidade dos dispositivos e a determinação de sua origem são problemas complexos. Assim, a proteção de dados e a integridade nas redes IoT necessitam de soluções confiáveis para garantir a entrega dos dados sem modificações entre os hosts.

- **Autenticação**: É a confirmação da identidade do remetente e destinatário (KUROSE; ROSS, 2012), pois identifica a legitimidade do usuário ou dispositivo participante da comunicação. Além disso, com o aumento escalável de dispositivos IoT na Internet, a autenticação tornou-se grande ameaça (JOSHITTA; AROCKIAM, 2016), pois, os atacantes tentam se legitimar como usuário ou dispositivo na rede.

- **Não repúdio**: É a garantia de que os dispositivos ou usuários não neguem a informação originada a partir deles (KUMAR; PATEL, 2014). Assim sendo, no contexto de um ambiente IoT, um nó não pode negar a informação que foi enviada ao outro nó ou ao usuário

- **Disponibilidade**: É um princípio da segurança da informação que envolve a capacidade de recuperação e confiabilidade em ambiente IoT (LASZLO, 2007). Ademais, segundo (CARNEIRO; FEITOSA, 2014) o processo de *fingerprint* pode detectar os possíveis protocolos e dispositivos passíveis de ataques em uma rede de dados com dispositivos IoT. Ou seja, os atacantes utilizam essa informação e enviam um grande número de requisições, com isso, os equipamentos deixam de responder a outras solicitações e podem exaurir os seus próprios recursos. Com isso, deixam de atender às solicitações de outros dispositivos ou usuários na rede.

2.6 Ferramenta NMAP

A Ferramenta NMAP (*Network Mapper*) é um software gratuito escrito sob código aberto com o objetivo de exploração de rede e auditoria de segurança em ambientes de redes. Além disso, ele é ferramenta útil para tarefas como inventário de rede, gerenciamento de atualização dos serviços e monitoramento de host ou tempo de atividade de serviço, usado por empresas e organizaçãoes. Além disso, o NMAP usa como técnica o envio de pacotes brutos sob o IP para determinar quais hosts são detectados como disponíveis na rede e quais os serviços - nome do software e versão - estão sendo oferecidos. Ademais, o programa identifica os sistemas operacionais e suas versões, que estão executando. Também, o tipo de filtros de pacotes ou firewalls que estão em uso e dezenas de outras características. Por conseguinte, foi projetado para

fazer a varredura rápida em grandes redes e funciona ,também, bem em hosts únicos (LYON, 2008). Ademais, na figura 15 é observado a execução da ferramenta.

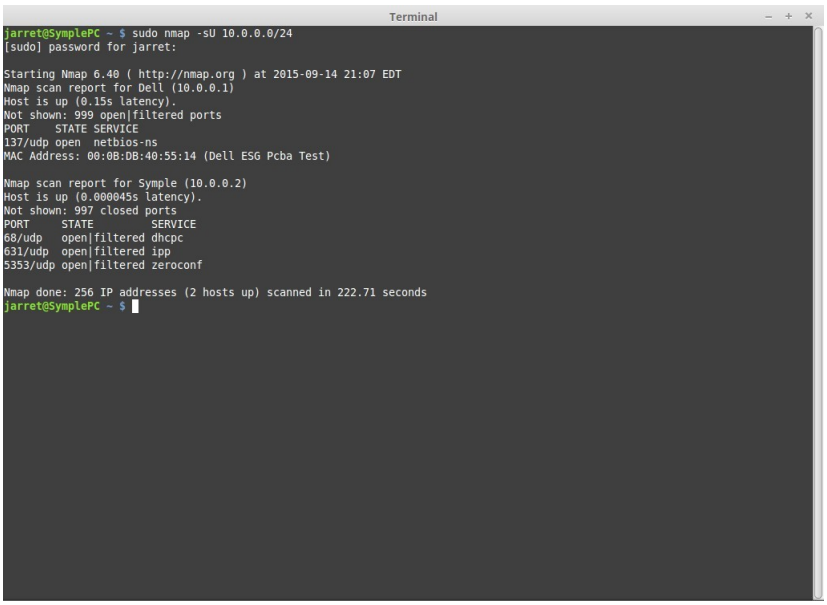


Figura 15 – Execução da ferramenta NMAP

2.6.1 Ferramentas de Segurança

2.6.1.1 Ferramenta Sparta

O software SPARTA é um programa escrito na linguagem de programação Python com GUI (*Graphical User Interface*) o qual simplifica o teste de segurança da infraestrutura de rede, auxilia o administrador de redes no processo de varredura e enumeração. Ademais, a GUI possibilita economizar tempo, pois, o acesso ao menu, opções e parâmetros são facilmente acessados e os resultados são exibidos na própria GUI. Dessa forma, o tempo de análise é menor se comparado com outras ferramentas de ataques e intrusões (QUINA; STAVLIOTIS, 2015).

2.6.1.2 Ferramenta Tshark

O software tshark é um analisador de protocolos de redes. A partir dele, pode-se capturar dados da rede por meio da leitura de informações das funções de rede no sistema operacional. Ademais, esses arquivos são gravados no formato PCAPNG nativamente, que pode ser lido pelo software wireshark e outros.

Além disso, ele pode ser executado sem filtros assemelhando-se aos outros softwares, por exemplo, o tcpdump. O funcionamento dele necessita da instalação da biblioteca PCAP. Além disso, o tshark é um software utilizado na linha de comando com as funcionalidades de um analisador gráfico. Dessa forma, ele é comumente utilizado na análise e captura de tráfego de redes em terminal sem a necessidade de bibliotecas gráficas (COMBS, 1998).

2.6.1.3 Ferramenta Tranalyzer

O software Tranalyzer é construtor de fluxo de redes e analisador de pacotes da pilha de protocolos TCP/IP (*Transmission Control Protocol* / *Internet Protocol*). Ademais, é leve, simples, tem bom desempenho e é escalável, pois, é escrito em código aberto, também é implementado sob a linguagem de programação C e baseado na biblioteca LIBPCAP. Essas funcionalidades são estendidas da ferramenta Cisco NetFlow e são disponibilizadas para os profissionais e pesquisadores por meio dessa ferramenta. Outrossim, o Tranalyzer fornece recursos para realizar a análise e gerar parâmetros-chave. Além disso, pode-se gerar dados estatísticos em tempo real ou de arquivos PCAP capturados. Os relatórios de extração do software são gerados em formato texto. Esses relatórios dependem da instalação de plugins. Dessa forma, é possível personalizar os relatórios de acordo com as necessidades (BURSCHKA STEFAN; DUPASQUIER, 2020).

2.7 Ataques aos dispositivos IoT

2.7.1 Agressive Scan

O Agressive Scan é um ataque de reconhecimento executado a partir do software NMAP. É realizado para detectar o sistema operacional do host de destino, realiza a detecção de serviços

de rede e suas versões. Outrossim, inclui o comando traceroute, o qual realiza uma estatística de respostas de saltos na rede a partir do protocolo ICMP (*Internet Control Message Protocol*). Além disso, realiza uma varredura de scripts.

2.7.2 Scan UDP

O ataque Scan UDP é utilizado com o objetivo de realizar escaneamento das portas virtuais no protocolo UDP a partir do software NMAP. Utiliza um método de varredura diferente de outros ataques se comparado a outros métodos que enviam pacotes brutos. Ademais, é utilizado por usuários Unix contra alvos com endereço IPv6, pois, esse tipo de ataque não envia o pacote de pedido de estabelecimento de conexão, denominado SYN.

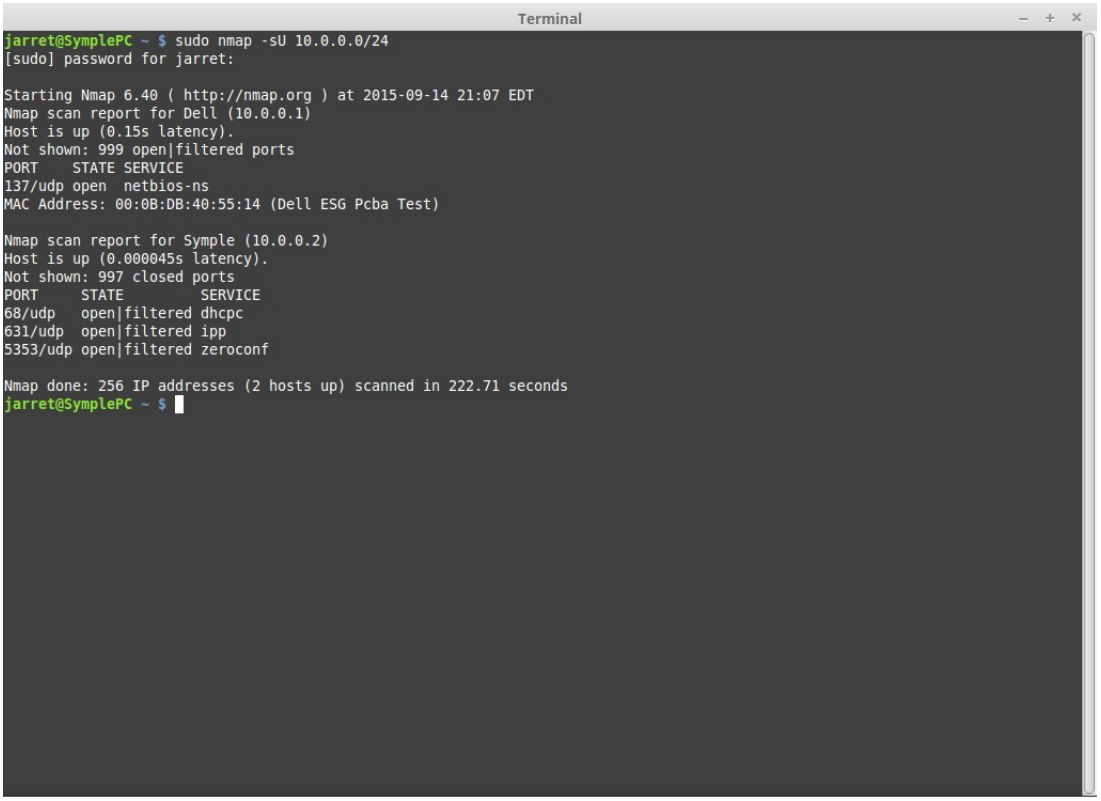


Figura 16 – NMAP Scan UDP

2.8 PowerAPI

O PowerAPI é composto por um conjunto de ferramentas para construir medidores de energia definidos por software [2], e, também, denomina-se medidor de energia (OLIVEIRA et al., 2020). Além disso, são configuráveis por meio de bibliotecas de software que são capazes de estimar o consumo de energia da aplicação ou parte dela em tempo real. Ademais, a arquitetura de medição de energia é constituída por dois componentes, um sensor e uma fórmula, esta é utilizada para produzir uma estimativa do consumo de energia dos componentes de hardware.

Essa estimativa de consumo de energia é utilizada em vários domínios: computação de alto desempenho (EASTEP et al., 2017), Sistemas baseados em virtualização (COLMANT et al., 2015), sistemas distribuídos (COLMANT et al., 2017), dentre outros. Ademais, a figura 17 demonstra a arquitetura do *Power Meters*. Nela observa-se componentes da medição de energia.

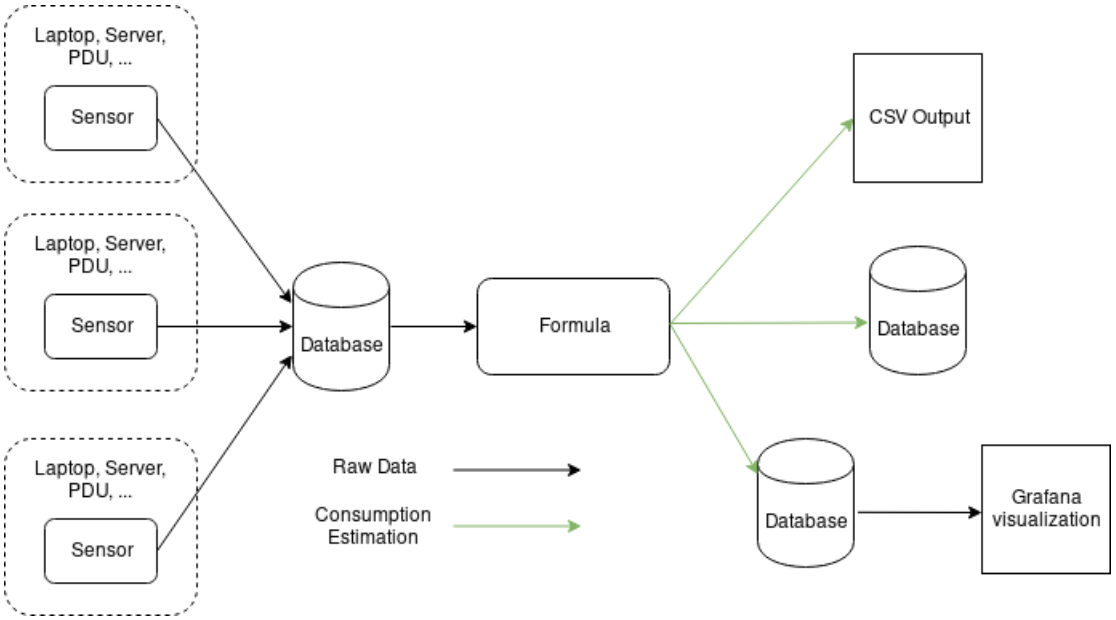


Figura 17 – Arquitetura Power Meters

Fonte: (GROUP, 2020)

Na figura 17 observa-se que os sensores coletam os dados brutos correlacionados com o consumo de energia do software, por sua vez, a fórmula é um modelo que usa os dados coletados para produzir estimativas de consumo. Essas informações são armazenadas em banco de dados utilizados para conectar esses dois componentes e enviar os dados para a fórmula [3].

Ademais, a biblioteca PyJoules (INRIA, 2019), na linguagem de programação Python, estima o consumo de energia da CPU (*Central Processing Unit*), memória e GPU (*Graphics processing unit*) integrada, usando a tecnologia da Intel RAPL (*Running Average Power Limit*). Além disso, também, usa a tecnologia *Nvidia Management Library* para realizar a medição de

[2] https://powerapi-ng.github.io/
[3] https://powerapi-ng.github.io/

energia de dispositivos nvidia (INRIA, 2019). Na figura 18 a seguir observa-se os domínios de leitura dessa biblioteca.

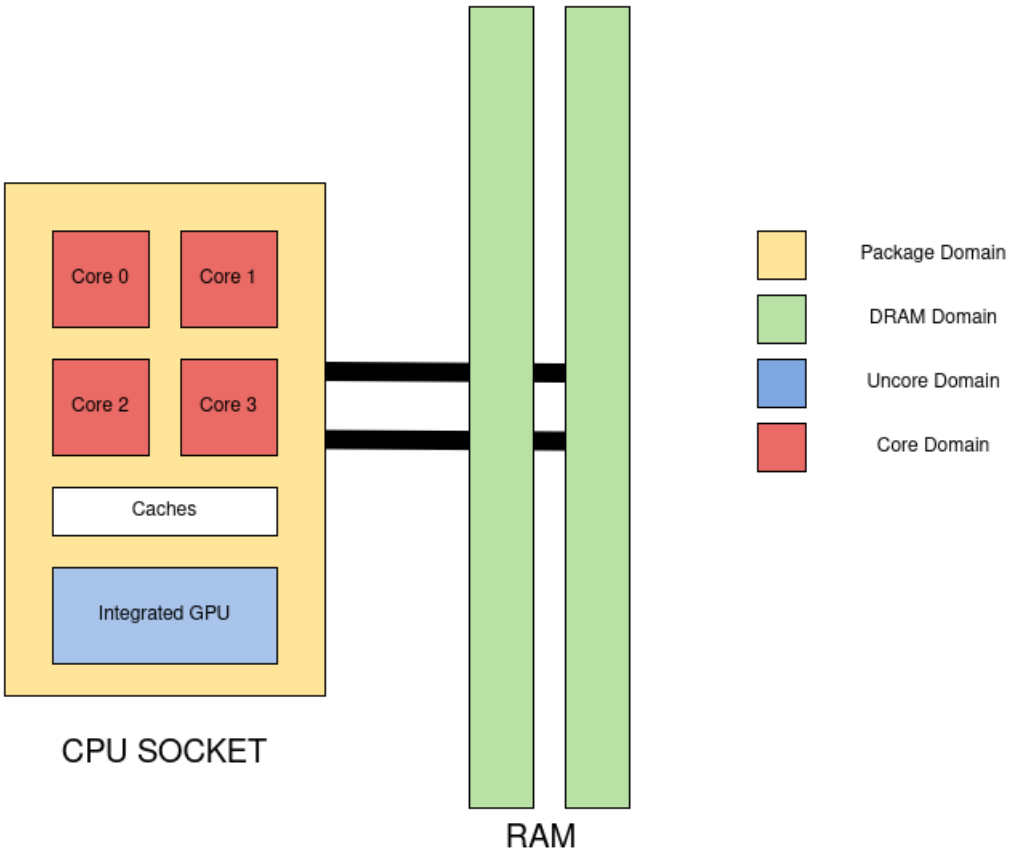


Figura 18 – Domínios de leitura da biblioteca PyJoules

Fonte: (INRIA, 2019)

A figura 18 mostra os domínios de componentes de leitura da medição de energia. Dessa forma, o *package* refere-se ao consumo de energia da parede da CPU. Outrossim, o *core* corresponde à soma da estimativa de todo o conjunto dos núcleos da CPU. Ademais, o *uncore* corresponde ao consumo da GPU integrada. Ainda, a RAM (*random access memory*) corresponde à estimativa de consumo da memória.

3 METODOLOGIA

3.1 Materiais e Métodos

Este trabalho utilizou os métodos de pesquisa exploratória e aplicada. Aquele busca por trabalhos que aumentem a quantidade e possibilitam a extração de informações. Este é considerado outra classificação na pesquisa, pois aplica-se à prática do conhecimento científico, com métodos, protocolos e experimentos em ambiente real. A metodologia deste trabalho divide-se em 4 etapas apresentadas pelo diagrama metodológico representado pela figura 19, a seguir.

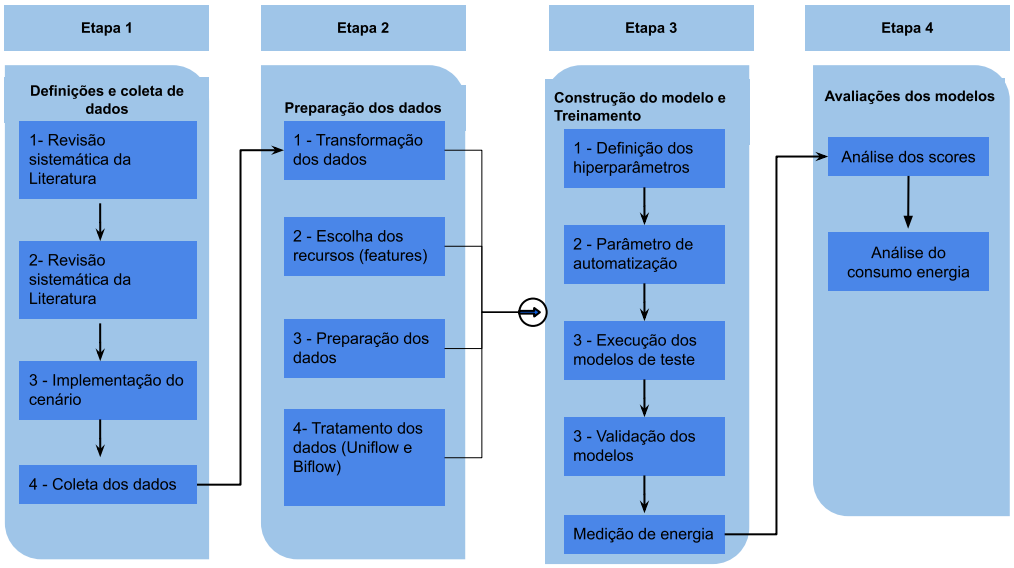


Figura 19 – Diagrama da Metodologia do Trabalho

Fonte: Elaborado pelo autor (2021)

Na figura 19 observa-se que a primeira etapa do projeto é composta por duas RSLs que auxiliaram na definição e delimitação da pesquisa que resultaram em artigos que estão nos anexos.

Na primeira RSL conduziu-se uma pesquisa exploratória de trabalho relacionados a IoT, WSN (*Wireless Sensor Networks*) *sensors*, segurança e energia que resultou na publicação em (OLIVEIRA et al., 2020). Ou seja, analisando esse artigo, observa-se que foram encontrados mais de 800 artigos que avaliam o consumo de energia das soluções de segurança WSN e IoT por meio de simulações, modelagem ou experimentos práticos por medição. Desses, apenas 72 trabalhos descreveram os experimentos por meio de análises práticas experimentais que são

difíceis de serem refeitos, pois necessita de um endereçamento do protocolo de condução desses experimentos em ambiente real, ou seja, o tempo para reprodução, solicitação do protocolo através de contato com o autor e busca nos meios eletrônicos são fatores que dificultam nesse processo tornando-o inviável à reprodução dessas pesquisas. Esta primeira RSL contribuiu com a publicação de um artigo em e na identificação de lacunas de pesquisas que relacionem IoT, segurança, energia, disponibilidade e medições com dispositivos reais.

A partir disso, realizou-se uma segunda pesquisa exploratória (anexo D) que buscou uma correlação com a área de segurança em redes de IoT utilizando SDN e abordagens IA. Esta segunda pesquisa teve a participação de um estudante do Curso de Sistemas para Internet do IFRN que resultou em um Trabalho de Conclusão de Curso do aluno Daniel Walmir dos Santos Alves, do IFRN - campus Currais Novos, orientado pelo autor deste trabalho (Moroni Neres Vieira) e pela Profª Drª Luciana Pereira Oliveira.

Após essa segunda RSL, iniciou-se a pesquisa aplicada que será detalhada das próximas seções. Isto é, os processos para a coleta de dados, a transformação, os tratamentos dos dados e a elaboração dos conjuntos de dados (*datasets*) dos protocolos MQTT e CoAP para que sejam apresentados aos algoritmos de ML e, a partir destes sejam realizadas as aprendizagens dos modelos e realização dos testes de aprendizagem. Ademais, os métodos para execução dos algoritmos de ML e os critérios de avaliação. Outrossim, serão definidos os atributos selecionados, bem como o processo para definir o procedimento de classificação. Seguindo este trabalho, é proposto um processo de padronização na captura de dados dos protocolos e transformação em informações estatísticas de tráfego de comunicação dos protocolos IoT. Além disso, realizou-se avaliações de desempenho dos modelos de ML.

Dessa maneira, por meio das avaliações são realizadas as comparações do trabalho em tela e o trabalho de (HINDY et al., 2020b), ainda assim, no processo de validação utilizou-se as seguintes métricas: acurácia, precisão, revocação e f1-*score*. Além disso, é realizada a avaliação de consumo de energia dos componentes de *hardware* enquanto os modelos de ML são executados.

3.2 Ambiente de Simulação

O ambiente de simulação foi criado para coleta de dados do protocolo CoAP com configurações similares aos dos dispositivos IoT. O cenário é composto por 12 sensores, dois roteadores, uma máquina representada pelo atacante e um cliente CoAP, o qual realiza solicitações aos sensores para obter informações.

Os sensores IoT são representados pelas máquinas virtualizadas no cenário, por meio do programa VirtualBox (ORACLE, 2015). Há neles as seguintes configurações: 1 processador, 500 MB (megabytes) de memória RAM e HD (*Hard Disk*) de 8 GB (Gibagytes). Essas máquinas virtuais estão em uma máquina hospedeira que contém a seguinte configuração: processador

Intel da 7ª geração, 16 GB de memória DDR4 e sistema operacional Linux Debian.

Além dos *hardwares* utilizados nesse cenário de simulação, os softwares utilizados para realizar os ataques são: NMAP e *Sparta*. Além disso, a captura do tráfego é realizada por meio do programa Tshark (WIRESHARK, 2005).

As máquinas e os softwares foram utilizados no diagrama representado pela figura 20. Além disso, observa-se nesse modelo as máquinas que compõem o cenário e suas interligações representando a topologia da rede com todos os componentes. Esse modelo de cenário foi descrito por (HINDY et al., 2020b) e foi utilizado para realizar a captura de tráfego para o protocolo MQTT. Ainda sim, neste trabalho adotou-se o mesmo cenário adaptando-o para a coleta dos dados de tráfego de rede do protocolo CoAP.

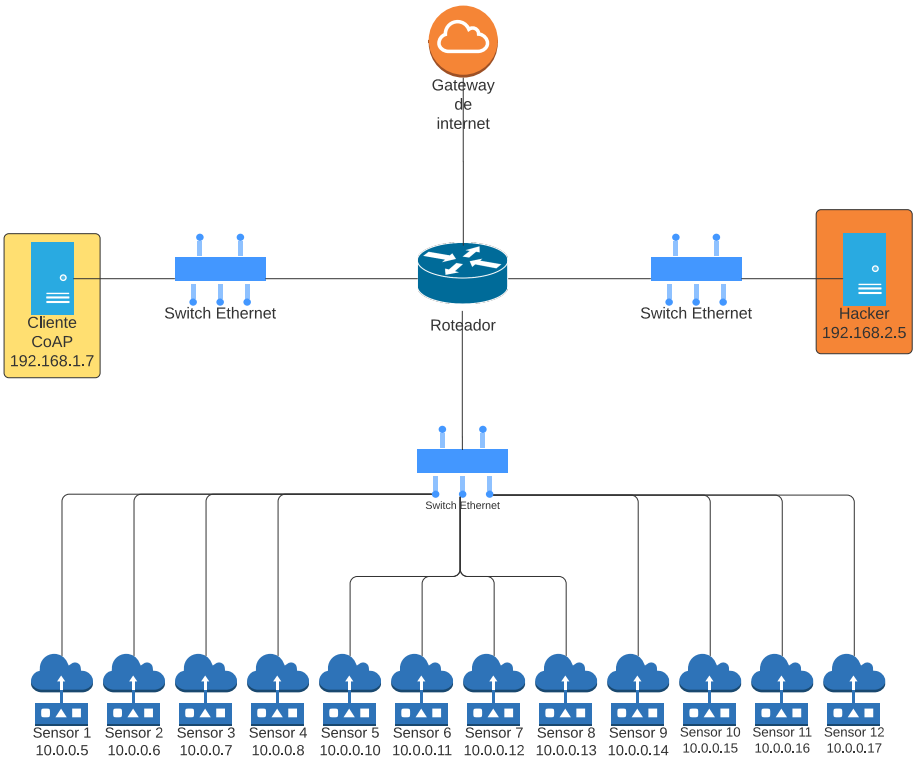


Figura 20 – Modelo Topológico do cenário da simulação CoAP

Fonte: Elaborado pelo autor (2021)

No cenário representado pela figura 20 visualiza-se dois roteadores, o *gateway* de Internet, o qual é representado no simulador VirtualBox pela interface NAT dele e realiza a conexão com a Internet dos *hosts* da rede por meio do segundo roteador, que encontra-se no centro do diagrama, os *Switches* Ethernet realizam as interconexões entre os elementos da rede e é representado no simulador por meio da conexão da Rede Interna, essa configuração está disponível em cada máquina virtual, a máquina hacker com o sistema Kali Linux instalado e o Cliente CoAP que realiza as requisições para os sensores que encontra-se em outra rede.

Por fim, os sensores que estão na parte superior do diagrama, são responsáveis por atender às requisições do cliente e estão habilitados com o protocolo CoAP, esses também são considerados servidores na arquitetura cliente e servidor.

3.3 Algoritmos Avaliados

A realização deste trabalho baseia-se na coleta dos dados, transformação, tratamento deles e execução dos algoritmos de ML. Além disso, na fase de execução dos modelos dos algoritmos de ML são usados hiperparâmetros para otimizá-los e parâmetros de automatização. Também, nesse processo de automatização foi adaptado o código de (HINDY et al., 2020b) que está disponível em (HINDY et al., 2020a), o qual foi escrito na linguagem de programação Python, ademais, ele está no repositório do projeto [1], é utilizado para automatizar a quantidade de execuções, os parâmetros, quais modelos de ML serão executados e seus hiperparâmetros. A seguir estão listados os modelos de ML utilizados:

- Regressão Logística
- k-vizinho mais próximo
- *Gaussian Naive Bayes*
- Árvore de decisão
- Florestas Aleatórias
- Máquina de Vetor de Suporte (linear and RBF kernel)

Estes algoritmos serão utilizados para classificar o tráfego da rede de dispositivos IoT entre ataques e legítimos, os procedimentos para adquirir os dados de tráfego serão descritos na seção de coleta de dados.

3.4 Ataques avaliados

Os ataques são realizados por meio do *host* hacker aos sensores IoT como visualizado no cenário representado pela figura 20. Ainda, os programas NMAP e *Sparta* são utilizados para realizar as investidas de reconhecimento e de acesso, respectivamente. Ademais, os ataques explorados neste trabalho são os seguintes:

- *Agressive Scan*
- *Scan* no *User Datagram Protocol* (UDP)

[1] https://www.https://github.com/moronivieira/IDS-CoAP

- *Sparta Brute Force* para o serviço SSH (*Secure Shell*)

O ataque *Agressive Scan* é utilizado neste trabalho para detectar o sistema operacional dos sensores, detecção de serviços de rede e suas versões. Além disso, realiza uma varredura de scripts automatizada, direcionada aos sensores.

Outrossim, o ataque Scan UDP é utilizado neste trabalho para realizar uma varredura de portas virtuais que utilizam o protocolo UDP, pois o referido protocolo é utilizado por padrão para comunicação entre os sensores e os clientes CoAP.

Por fim, o programa *Sparta* é utilizado nos experimentos para automatizar e simplificar os testes de ataques de força bruta ao serviço SSH dos *hosts* da rede, e, também, auxilia na fase de varredura e enumeração em ataques na rede IoT utilizada nesta pesquisa. Portanto, ele é utilizado para analisar resultados a partir de um ataque ao serviço SSH dos sensores e sua saída está representada pela figura 21.

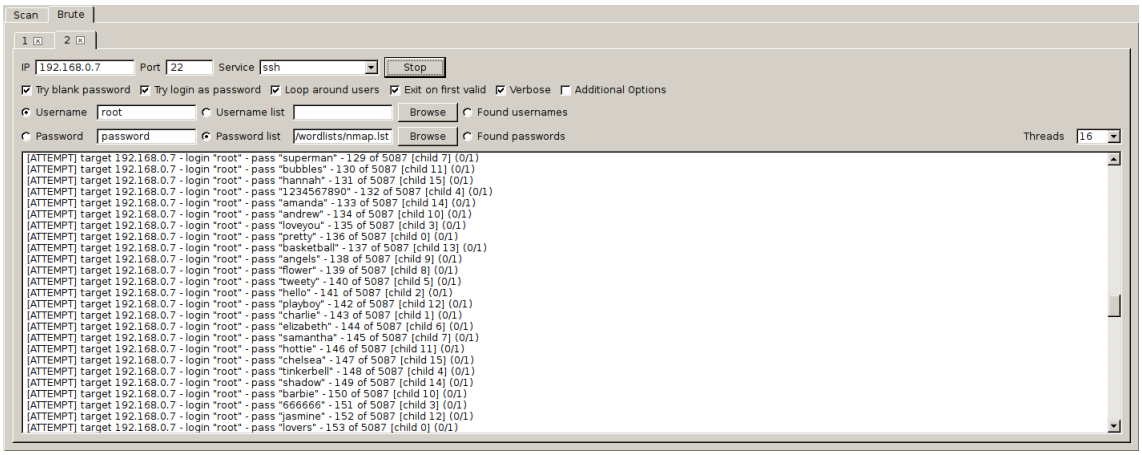


Figura 21 – Ataque de Força Bruta do Sparta

Fonte: (SERVICES, 2021)

Os ataques descritos foram adaptados a partir do trabalho de (HINDY et al., 2020b) para o protocolo CoAP, o qual é utilizado neste trabalho. Ademais, os sensores CoAP mostraram-se pouco suscetíveis ao aumento de requisições. Consequentemente, eles deixam de responder às solicitações quando há o aumento da quantidade de requisições e mensagens de erros de conexão de rede são informados pelo sensor. Dessa forma, foi possível realizar um conjunto de investidas do cliente aos sensores.

Diante disso, foram sistematizados ataques de forma sequencial aos sensores, ou seja, um tipo de ataque por vez, foi parametrizado também, a execução de requisições do cliente de 1 por segundo, e, de forma paralela foi realizada às solicitações de tráfego legítimo do cliente aos sensores IoT e de ataque do hacker aos sensores. Assim, caracterizaram-se operações cotidianas no dia a dia, em um ambiente real com dispositivos IoT, onde o atacante pode estar oculto na rede interna ou na Internet e realizando o ataque de forma simultânea com as operações comuns.

3.5 Coleta dos dados

A coleta de dados representando o tráfego legítimo, ou seja, as requisições e respostas do cliente CoAP aos sensores foram realizadas de forma autônoma por meio de um script escrito na linguagem de programação Bash Script, conforme anexo B. O script realizou requisições utilizando-se do método GET ao protocolo CoAP. Diante disso, foram feitos 4 tipos de requisições de mensagens aos sensores, sendo elas: "time, whoami, other/separate e other/block". Ao executar o script foi utilizado um arquivo auxiliar que contém os endereços IP dos sensores, e, uma variável com valor randômico a fim de prover a aleatoriedade da consulta aos sensores.

Por outro lado, na continuidade, após a construção do script, este é executado isoladamente para o tráfego normal e de forma concomitante quando se está executando os ataques por meio da máquina hacker. Além disso, na execução do script e ataque a partir da máquina hacker, a coleta dos dados dos tráfegos foi realizada através do programa Tshark, gerando arquivos no formato PCAP do tráfego normal, do ataque *agressive scan, scan UDP* e os ataques da ferramenta *Sparta* direcionadas ao serviço ssh dos sensores.

Ademais, os dados brutos para o MQTT, que foram utilizados na pesquisa estão disponíveis em [2]. Portanto, foi possível coletar dados com tamanho e características similares ao trabalho de (HINDY et al., 2020b).

Ainda, a coleta de dados energéticos foi realizada durante a fase de aprendizagem e testes dos algoritmos de ML para realizar a estimativa de energia consumida por cada um dos modelos. Dessa forma, é possível projetar uma divisão de análises dos pacotes na rede objetivando a diminuição de consumo de energia nos IDS, essa captura de dados é realizada por meio do PowerAPI. Portanto, a precisão do PowerAPI é mensurada em um medidor de potência, a margem de erro dele é calculada para variar entre 0,5% a 3% (NOUREDDINE; ROUVOY; SEINTURIER, 2013).

3.6 Transformação dos Dados

As informações do tráfego foram gravadas em arquivos do tipo PCAP e foram lidas pelo *software* Tranalyzer (BURSCHKA STEFAN; DUPASQUIER, 2020), o qual transformou os dados brutos da captura de rede em informações dos fluxos de redes entre os *hosts* da rede. A próxima etapa, consequentemente, foram utilizadas as ferramentas Tawk (BURSCHKA STEFAN; DUPASQUIER, 2020) e Awkf (BURSCHKA STEFAN; DUPASQUIER, 2020) por meio do terminal Linux para selecionar os dados e transformá-los em arquivos do tipo CSV. Além disso, para cada tipo de tráfego ou ataque foram criados dois arquivos para representar os fluxos bidirecionais e unidirecionais. Por fim, após a escolha das características de composição dos dados foram selecionadas as colunas dos arquivos para filtrar e transformá-los em arquivos CSV para tratamento deles posteriormente.

[2] https://ieee-dataport.org/open-access/mqtt-iot-ids2020-mqtt-internet-things-intrusion-detection-dataset

3.7 Escolha das Características (*features*)

As características (*features*) são utilizadas pelos modelos de aprendizado em ML para compor um conjunto de informações, as quais são lidas pelo algoritmo de aprendizagem, representando nomes de colunas no *dataset*. Diante disso, é realizada a leitura das informações na fase de aprendizagem e, posteriormente, para determinar o comportamento do tráfego de rede na fase de testes e aprendizagem.

Os recursos escolhidos neste trabalho são avaliados em dois níveis de características, denominadas, fluxo unidirecional e bidirecional, este representa a comunicação entre a origem e destino com resposta à origem do destino, aquele representa o tráfego de um *host* de origem a um *host* de destino sem haver resposta na comunicação. A seguir, encontra-se a tabela 4 com a descrição de cada recurso, o tipo de dado, a descrição e se é fluxo unidirecional ou bidirecional.

Tabela 4 – Tabela de Características

Característica	**Tipo de dado**	**Descrição**	**Fluxo Unidirecional**	**Fluxo Bidirecional**
ip_src	Texto	Endereço IP de origem	x	x
ip_dest	Texto	Endereço IP de destino	x	x
prt_src	Inteiro	Porta de Origem	x	x
prt_dst	Inteiro	Porta de Destino	x	x
proto	Inteiro	Protocolo da Camada de Transporte (TCP/UDP)	x	x
num_pkts	Inteiro	Quantidade de Pacotes no Fluxo	x	x
mean_iat	Decimal	Tempo Médio entre as Chegadas de Fluxo	x	x
std_iat	Decimal	Desvio Padrão de Tempo entre as Chegadas de Fluxo	x	x
min_iat	Decimal	Tempo mínimo entre as Chegadas de Fluxo	x	x
max_iat	Decimal	Tempo máximo entre as Chegadas de Fluxo	x	x
num_bytes	Inteiro	Número de bytes	x	x
num_psh_flags	Inteiro	Número de Flags PSH	x	x
num_rst_flags	Inteiro	Número de Flags RST	x	x
num_urg_flags	Inteiro	Número de Flags URG	x	x
mean_pkt_len	Decimal	Tamanho Médio dos Pacotes	x	x
std_pkt_len	Decimal	Desvio Padrão do Tamanho dos Pacotes	x	x
min_pkt_len	Decimal	Tamanho Mínino dos Pacotes	x	x
max_pkt_len	Decimal	Tamanho Máximo dos Pacotes	x	x
is_attack	Binário	Se for 1, indica ataque, senão representa outro tipo	x	x

Os conjuntos de dados dos fluxos bidirecionais apresentam duas colunas denominadas pelo prefixo fwd e bwd, este representa a resposta do destino ou sentido inverso da informação,

aquele representa o envio da informação. Já a linha is_attack representa o registro de ataque nas linhas desse conjunto, desse modo, se o número for 1, indica o ataque, mas, se o número for 0, indica o tráfego legítimo. Além disso, cada linha do conjunto de dados representa uma instância, ou seja, elas são usadas pelos algoritmos de aprendizado e representa o conjunto de recursos.

3.8 Preparação dos Dados

Os conjuntos de dados, denominados *datasets*, são necessários para realizar o treinamento e testes dos algoritmos de ML e foram obtidos de duas maneiras, a primeira para o protocolo MQTT, este está disponível em base pública no endereço [3], o outro *dataset* foi criado a partir da descrição dos dados de (HINDY et al., 2020b) da captura de tráfego realizada no ambiente de simulação e a tranformação dos dados. Portanto, foi construído um *dataset* próprio para os experimentos do protocolo CoAP, pois, na literatura não existe *dataset* disponível para realizar a comparação.

Esses *datasets* possuem características do funcionamento do protocolo IoT em uma rede de computadores. Ainda, foram divididos em 4 *datasets* representando o tráfego normal, ataque de *Agressive Scan*, ataque de *Scan* ao protocolo UDP e o ataque de força bruta de acesso de login (SSH) por meio da ferramenta de intrusão *Sparta*. Além disso, também foram extraídos os campos necessários para compor o *dataset*, por meio de filtragem com as ferramentas Tawk e Awkf (BURSCHKA STEFAN; DUPASQUIER, 2020). Portanto, todos os *datasets* envolvidos nesta pesquisa contém essas características.

As informações do *dataset* precisam atender aos requisitos para pesquisa de análise de tráfego de rede para detecção de anomalias em rede, pois, a comunicação entre dispositivos IoT apresenta protocolos e mensagens entre si singulares. Inicialmente, a construção do *dataset* foi realizada com a coleta de dados de tráfego de rede por meio do *software* Tshark (WIRESHARK, 2005), a partir disso, 4 arquivos do tipo PCAP foram criados, consecutivamente, 4 arquivos do tipo CSV também foram criados.

3.9 Tratamento dos Dados

O tratamento dos dados foi realizado para selecionar os campos de acordo com as características de cada protocolo IoT, MQTT e CoAP, e também dos fluxos unidirecional e bidirecional. Ademais, esse tratamento foi realizado por meio de ferramentas do *software* Tranalyzer no Linux e posteriormente com a biblioteca Pandas da linguagem Python.

Esse tratamento foi necessário para formatar os dados dos fluxos bidirecionais, pois, somente o tráfego unidirecional pode ser concebido por meio da ferramenta Tranalyzer. Desse

[3] https://ieee-dataport.org/open-access/mqtt-iot-ids2020-mqtt-internet-things-intrusion-detection-dataset

modo, esses *datasets* assemelham-se aos *datasets* utilizados por (HINDY et al., 2020b) em sua pesquisa.

3.10 Definição dos Hiperparâmetros

A validação cruzada foi utilizada pelos modelos para criar um aprendizado confiável nos modelos de ML, por isso, foi necessário parametrizar o modelo para utilizar 5 dobras (*folds*) e informado a proporção de 75% para treinamento e 25% para aprendizado dos modelos. Por fim, as médias de todas as execuções foram extraídas e utilizadas para as tabelas e gráficos nos resultados.

3.11 Parâmetros de Automatização do Script

Os parâmetros foram definidos no script com as funções dos algoritmos de ML. Dessa forma, ele foi executado 4 vezes, sendo duas vezes para o protocolo MQTT, e duas vezes para o protocolo CoAP, divididos, igualmente, em unidirecional e bidirecional. A partir disso, definiu-se 250 repetições em cada algoritmo, e, também, foi utilizada a biblioteca PyJoules, implementada na linguagem Python para medir o consumo de energia de cada algoritmo de ML. Com isso, foi possível realizar as avaliações dos modelos de ML e o consumo de energia deles.

3.12 Execução dos Modelos de ML

A implementação do protocolo CoAP no ambiente de simulação foi realizada por meio de software utilizando a biblioteca AIOCoAP. Ademais, contém os elementos - servidor, ciente e *proxy* - de uma rede para dispositivos IoT, e, também, é possível realizar uma instalação com recursos mínimos de *hardware*, por meio do clone do repositório github no endereço eletrônico [4]. A partir da clonagem do repositório github do pacote do projeto AIOCoAP, observou-se que os códigos fonte para implementação das funções de servidor, cliente e *proxy*, estão disponíveis, diante disso, neste trabalho, utilizou-se somente a implementação para as funções de servidor e cliente. O servidor de aplicação contém as respostas às mensagens para solicitações dos clientes. Fundamentando-se nisso, quando executa-se essa instância desse pacote, uma aplicação padrão sob o protocolo CoAP é iniciada utilizando-se a porta 5683 sobre o protocolo UDP.

Já o cliente necessita de parâmetros para ser iniciado na linha de comando. Os parâmetros utilizados neste trabalho são os seguintes:

- aiocoap-client -m get: Esse parâmetro é utilizado para as solicitações sob o método get;

[4] https://github.com/chrysn/aiocoap

- aiocoap-client -m get coap://192.168.1.1/whoami: Esses parâmetros informa que as solicitações deverão utilizar o método get para a URI sobre o IP 192.168.1.1 e a solicita resposta para a mensagem whoami.

Ademais, os treinamentos e testes dos algoritmos de ML foram realizados em uma máquina com a seguinte configuração: sistema operacional Linux Ubuntu, Processador Intel 7ª geração, 16 GB de memória RAM e disco rígido SSD. Essas execuções foram realizadas de forma sequencial, ou seja, para cada fluxo - unidirecional e bidirecional -, bem como para cada protocolo IoT.

Os algoritmos de classificação de ML foram aplicados por meio de um script na linguagem python. Esse script foi adaptado a partir do trabalho de (HINDY et al., 2020b). Nesse sentido, esses algoritmos necessitam de alguns parâmetros para realizar a execução. A partir daí, a figura 22 a seguir demonstra a execução do script denominado classification_measure.py no terminal Linux, bem como os parâmetros necessários.

```
root@router:~# ./classification_measure.py --mode 1 --output /out --verbose True
```

Figura 22 – Execução do Script

Fonte: Elaborado pelo autor (2021)

Os modos de entrada do script, representado pelo parâmetro "–mode n", onde "n" representa o numeral 1 ou 2. O primeiro representa a entrada de dados com tráfego de comunicação unidirecional, e o outro representa a entrada de dados com tráfego bidirecional. Outrossim, as identificações dos fluxos são caracterizadas pelo número de identificação de cada um deles. Além disso, o parâmetro "/out", refere-se ao nome do diretório que será preenchido com os arquivos de resultados avaliativos dos algoritmos de ML. Também foi utilizada a biblioteca PyJoules para medir o consumo de energia de cada algoritmo. Por fim, são realizadas as avaliações de desempenho dos modelos de ML e o consumo de energia deles.

3.13 Validação dos Modelos de ML

Os critérios de avaliação utilizados neste trabalho são baseados nos resultados das revisões sistemáticas. Desse modo, os critérios que serão utilizados são: Verdadeiro positivo, Falso positivo, Falso verdadeiro e Falso negativo, segundo (BATISTA et al., 2003) conceitua-se cada um desses critérios da seguinte maneira:

- **Positivo verdadeiro (PV)**: é a porcentagem de casos positivos classificados como pertencentes à classe positiva real;
- **Falso positivo (FP)**: é a porcentagem de casos negativos classificados incorretamente como pertencentes à classe positiva;

- **Falso verdadeiro (VN)**: é a porcentagem de casos negativos classificados corretamente como pertencentes à classe negativa;
- **Falso negativo (FN)**: é a porcentagem de casos positivos classificados incorretamente como pertencentes à classe negativa.

A validação quíntupla é utilizada para avaliar cada modelo de ML. A principal métrica é a acurácia, a equação 2 é uma demonstração matemática da obtenção do valor da acurácia geral, onde o valor positivo verdadeiro (PV) representa as instâncias reais. Assim, a quantidade de ataques verdadeiros, neste trabalho, são representados por valores de positivo verdadeiros. Dessa maneira, O falso verdadeiro (VN) representa o tráfego benigno, assim, esse tipo de tráfego indica que a instância não representa um ataque. Além disso, o falso positivo indica a quantidade de ataques classificados incorretamente e o falso negativo (FN) representa o tráfego benigno classificado incorretamente. Por fim, o valor da acurácia geral é obtido pela razão entre o número total de predições corretas sobre o número total de previsões.

$$Acuracia = \frac{PV + VN}{TP + FP + VN + FN} \tag{2}$$

Outras métricas foram utilizadas para avaliar as classes dos modelos: Revocação (*recall*), Precisão (*precision*) e Pontuação F1 ($F1-score$). Segundo (HATTORI et al., 2008), a precisão indica qual a fração das classes estimadas foram realmente detectadas como verdadeiras. A partir disso, a equação 3 a seguir demonstra a definição dessa métrica.

$$Precisao = \frac{PV}{PV + FP} \tag{3}$$

Também define-se a relação entre falsos positivos e precisão como sendo a seguinte: quanto menor o valor de falsos positivos, maior será a precisão. Uma outra métrica que será utilizada é a revocação. Segundo (HATTORI et al., 2008), essa métrica descreve a proporção de acertos ao classificar elementos de uma certa classe dentro do modelo de ML.

$$Revocacao = \frac{PV}{PV + FN} \tag{4}$$

A relação entre falso negativo e revocação é definida da seguinte forma: quanto menor o valor de falsos negativos, maior será o valor de revocação.

Outrossim, a métrica F1-score que faz o balanceamento harmônico entre as métricas precisão e revocação, utilizada com o objetivo de analisar as duas medidas anteriores de maneira uniformizada. A equação 5 mostra esse modelo.

$$F1 - Score = \frac{2}{\frac{1}{precisao} + \frac{1}{Revocacao}} \tag{5}$$

Ainda, o peso médio (*weigthed average*) é calculado para a métrica precisão, revocação e F1-*Score*.

Por fim, essas métricas são comumente observadas em trabalhos relacionados aos modelos de ML, pois, os dados têm problemas de classes desbalanceadas. Esses problemas segundo (BATISTA et al., 2003) "correspondem a domínios nos quais uma classe é representada por um grande número de exemplos, enquanto que a outra é representada por poucos exemplos". Portanto, esse problema, também, está nos dados que poderão estar balanceados. Isso pode ser observado nas tabelas de instâncias dos arquivos de dados PCAP, no capítulo de resultados 4, onde as tabelas de instâncias mostram os números das instâncias.

3.14 Coleta dos Dados de Energia

A coleta de dados energéticos na fase de aprendizado e testes nos modelos de ML foi feito por meio da biblioteca PyJoules escrita na linguagem de programação Python, com a inclusão da instrução @measure_energy nas funções de aprendizagem e testes dos algoritmos de ML, assim, coletou-se informações sobre tempo de execução e estimativa de consumo energético deles, e, escrevendo-os em arquivo do tipo CSV.

3.15 Validação dos Dados Energéticos

A validação dos dados energéticos foi realizada por meio do cálculo da média do tempo de execução e do consumo de cada um dos componentes de *hardware* (CPU, GPU, Package e RAM). Além disso, realizou-se o cálculo do intervalo de confiança dos dados de consumo de energia deles.

Outrossim, o intervalo de confiança desses dados é calculado com o objetivo de mostrar um intervalo de estimativas de valores reais possíveis. Dessa forma, observa-se um intervalo de valores que representam uma probabilidade numérica. Também, dentro desse intervalo espera-se que a probabilidade desse valor também esteja (HORST, 1999).

3.16 Avaliação de Ferramentas e Recursos

3.16.1 Ferramenta CoAP-SHELL

A ferramenta foi utilizada na fase inicial do trabalho para interagir com servidores na Web habilitados com o protocolo CoAP disponíveis nos sítios: coap://californium.eclipse.org e coap://coap.me. Também, fornece ajuda na linha de comando para auxiliar na construção e sintaxe, a seguir é listado alguns recursos que também foram usados na ferramenta.

- Métodos CoAP GET, PUT, POST e DELETE;

- Observação de recursos do CoAP;
- Descoberta de recursos CoAP;
- Filtros por href, ct, rt, obs ... ;
- Trocas de mensagens síncronas e assíncronas (argumento –async);
- Troca de mensagens confirmadas e não confirmadas;
- Preenchimento automático de TAB para comandos e argumentos;
- Comandos extensivos ajudam (digite ajuda);
- Chaves plugáveis ou armazenamentos confiáveis e credenciais.

A interação com a ferramenta foi importante para entender o funcionamento do CoAP, capturar tráfego de rede de solicitações e respostas do cliente ao servidor. Entretanto, a ferramenta não suporta automatização de solicitações de interações, por esse motivo não pode ser utilizada nos experimentos.

3.16.2 AVALIAÇÃO DE MULTICAST NO COAP

Na fase inicial da pesquisa utilizou-se da difusão seletiva para enviar solicitações a um conjunto de sensores e foi ativado por meio de software no pacote AICOAP. Entretanto, muitos fóruns na Internet contém o questionamento sobre o uso desse recurso em um cenário com CoAP, por exemplo, no endereços a seguir: <https://stackoverflow.com/questions/43421239/multicast-coap-request-doesnt-reach-the-server> e <https://github.com/mcollina/node-coap/issues/143>. Dessa forma, verificou-se que por meio das pesquisas e documentações oficiais disponíveis na Internet o recurso de multicast não está com seu suporte completo para o CoAP.

O relatório da descoberta de serviços multicast depende da quantidade de *hosts* que respondem à requisição, por isso, essa lista poderá ser extensa. Entretanto, a aplicação não pode ser utilizada para realizar os testes no cenário dos experimentos, pois falta documentação e bibliotecas para a instalação e funcionamento dela.

3.16.3 Quick CoAP Multicast Discovery

A finalidade de utilização da ferramenta foi testar o multicast com uma interface GUI (Graphical User Interface). Porém, a documentação da ferramenta não está completa e as bibliotecas para o seu funcionamento não estão disponíveis.

4 RESULTADOS

O capítulo de resultados foi dividido em duas seções para apurar a compreensão dos resultados. Dessa forma, a primeira seção trata dos resultados obtidos das métricas por meio da execução dos modelos de ML. Já a segunda seção mostra os resultados da medição de energia durante a execução dos algoritmos.

4.1 AVALIAÇÃO DE DESEMPENHO DOS MODELOS DE ML

As instâncias ou ocorrências são valores encontrados em cada *dataset* representados pelos arquivos do tipo CSV. Ademais, na tabela 5 observa-se os valores dessas instâncias para o protocolo MQTT. Além disso, na tabela 6 observa-se, também, os valores das instâncias para o protocolo CoAP. Assim, as linhas desses arquivos representam a quantidade de ataques e tráfego benigno. Dessa forma, os algoritmos de ML utilizam esses dados para o processo de aprendizado e treinamento.

Tabela 5 – Tabela de Instâncias no Dataset MQTT

Num	Nome do arquivo	Tamanho do arquivo pcap	Número de instâncias Unidirecionais		Número de instâncias Bidirecionais	
			Benigno	Ataque	Benigno	Ataque
1	Normal	192,5 MB	172313	0	167684	0
2	scan_A (agressive)	16,2 MB	31881	20295	31553	20278
3	scan_sU (UDP)	41,3 MB	34487	22451	33539	19
4	Sparta	3,4 GB	1122002	967441	1117901	967441

Tabela 6 – Tabela de Instâncias no Dataset CoAP

Num	Nome do arquivo	Tamanho do arquivo pcap	Número de instâncias Unidirecionais		Número de instâncias Bidirecionais	
			Benigno	Ataque	Benigno	Ataque
1	CoAP_Normal	193 MB	487775	0	305522	0
2	CoAP_Scan_A (agressive)	16,3 MB	73181	73006	73066	72802
3	CoAP_Scan_sU (UDP)	41,4 MB	32135	154446	20388	48
4	CoAP_Sparta	3,7 GB	901355	865468	886739	864363

Nas tabelas 5 e 6 de distribuição de instâncias, observam-se os nomes dos arquivos, o tamanho deles, os números das instâncias legítimas e de ataque divididas por fluxos unidirecionais e bidirecionais.

Além disso, observa-se que os valores das instâncias entre o MQTT e o CoAP são diferentes, isso, por causa do ambiente e da maneira como o cliente foi configurado para solicitar informações aos servidores CoAP, e, porque (HINDY et al., 2020b) não deixa claro a configuração para as solicitações e respostas das mensagens entre cliente e servidores. Por exemplo, na linha 1 das duas tabelas observa-se que a quantidade de instâncias do tráfego legítimo na coluna dos dados de instância unidirecionais apresenta 172313 para o protocolo MQTT e 487775 para o CoAP, apesar de ter tamanho de arquivos com tamanhos similares.

Também, observa-se na tabela 6 na linha 3 "CoAP_Scan_Su" que 20388 instâncias foram classificadas como legítimas e somente 48 instâncias foram classificadas como ataque, isso, ocorre por causa dos dados do ataque ao protocolo UDP, a ferramenta NMAP, que é utilizada para esse tipo de ataque, produz informações com um intervalo de tempo baixo, consecutivamente, o arquivo bruto é preenchido rapidamente.

Os resultados de desempenho da acurácia geral de cada algoritmo são divididos por fluxos e são mostrados nas tabelas 7 e 8. Nessas tabelas, observa-se os valores da acurácia geral dos fluxos unidirecional e bidirecional para os protocolos MQTT e CoAP.

Inclusive, observa-se que as colunas denominadas algoritmos mostram os modelos de ML. Já as colunas unidirecionais e bidirecionais representam os dados percentuais da acurácia para o fluxo unidirecional e para o fluxo bidirecional, respectivamente. A seguir observa-se a tabela 7 para os resultados do protocolo MQTT.

Os resultados da métrica acurácia geral de cada algoritmo são divididos por fluxos e são mostrados nas tabelas 7 e 8. Nessas tabelas, observa-se os valores da acurácia geral dos fluxos unidirecional e bidirecional para os protocolos MQTT e CoAP.

Tabela 7 – Acurácia dos Algoritmos ML para o protocolo MQTT

Algoritmos	Unidirecionais	Bidirecionais
Decision Tree	99,977	99,854
SVM Kernel RBF	99,710	99,942
K-Nearest Neighbours	100,000	99,951
Gaussian Naive Bayes	99,222	99,848
Random Forests	99,954	99,951
Logistic Regression	69,743	99,946
SVM Linear	58,861	99,928

Os resultados da métrica acurácia geral das técnicas de ML são observados pelo gráfico que é representado pela figura 23 para os fluxos unidirecionais e bidirecionais do protocolo MQTT. Além disso, observa-se o aumento do desempenho dos algoritmos para o fluxo bidirecional.

No gráfico representando pela figura 23, visualiza-se uma melhora da métrica acurácia do desempenho dos sete algoritmos de ML. Desses, visualiza-se uma eficácia nos desempenhos

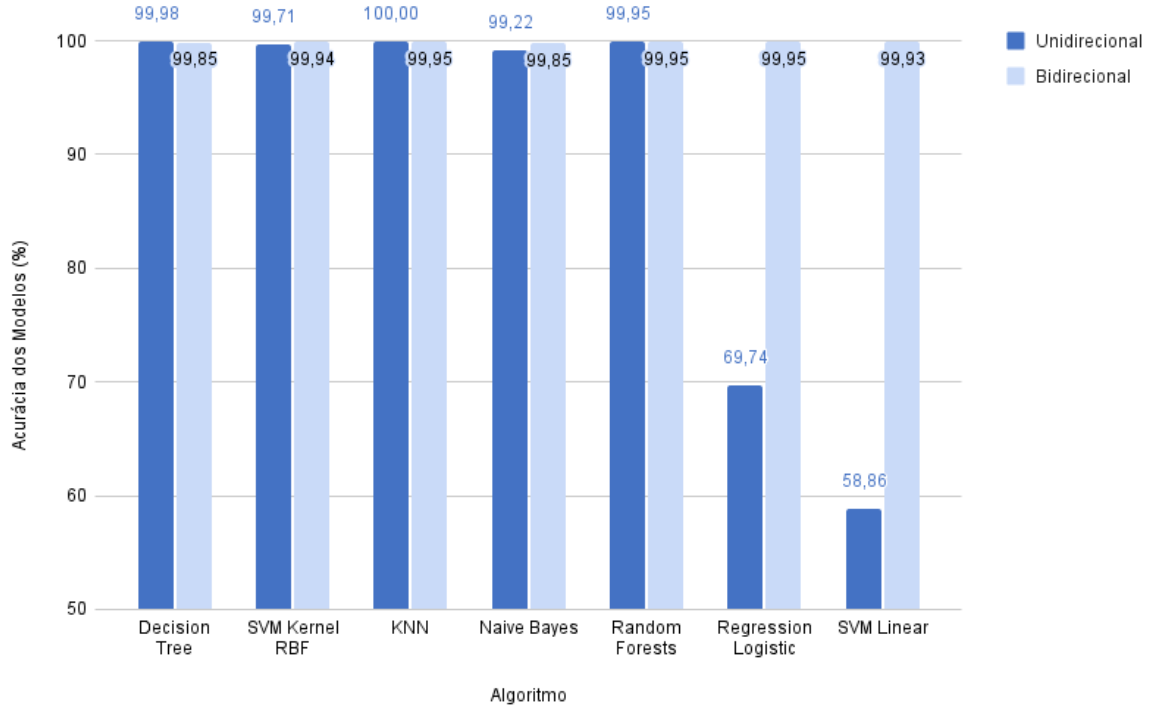


Figura 23 – Acurácias dos modelos de ML para os fluxos unidirecionais e bidirecionais do protocolo MQTT

Fonte: Elaborado pelo autor (2021)

dos algoritmos *Logistic Regression* que obteve uma melhora no desempenho de 69,74% no fluxo unidirecional para 99,95% no fluxo bidirecional. Neste outro, denominado de SVM *Linear* obteve uma melhora no desempenho de 58,86% no fluxo unidirecional para 99,93% no fluxo bidirecional.

Na tabela 8 a seguir estão as médias da acurácia dos algoritmos de ML para o protocolo CoAP. A partir da execução deles obteve-se os seguintes valores da métrica acurácia.

Tabela 8 – Acurácia dos modelos ML para o CoAP

Algoritmos	Unidirecionais	Bidirecionais
Decision Tree	99,958	99,855
SVM Kernel RBF	81,893	99,740
K-Nearest Neighbours	98,994	99,822
Gaussian Naive Bayes	47,354	98,849
Random Forests	99,948	99,855
Logistic Regression	80,138	99,735
SVM Linear	67,606	99,742

No gráfico 24 a seguir estão representados os resultados da desempenho dos algoritmos de ML dos fluxos unidirecionais e bidirecionais para o CoAP. Ademais, observa-se um aumento considerável, quando se executa os algoritmos de ML para o fluxo bidirecional.

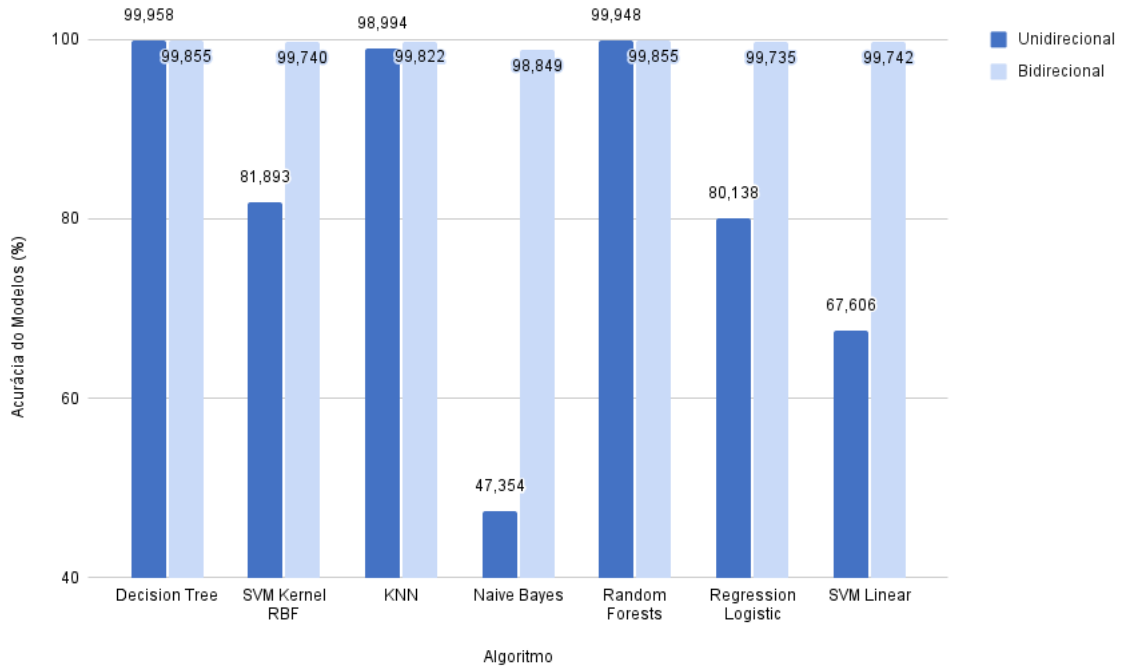


Figura 24 – Acurácias dos modelos de ML para os fluxos unidirecionais e bidirecionais do protocolo CoAP

Fonte: Elaborado pelo autor (2021)

No gráfico representando pela figura 24, visualiza-se uma melhora da métrica acurácia do desempenho dos sete algoritmos de ML. Desses, visualiza-se uma eficácia no desempenho do algoritmo Naive Bayes que obteve uma melhora no desempenho de 47,354% no fluxo unidirecional para 98,849% no fluxo bidirecional. Neste outro, denominado de *Logistic Regression* obteve uma melhora no desempenho de 80,138% no fluxo unidirecional para 99,735% no fluxo bidirecional. No SVM *Linear* que obteve um aumento de 67,606% no fluxo unidirecional para 99,742% no fluxo bidirecional e SVM com *Kernel* RBF que obteve um aumento de 81,893% no fluxo unidirecional para 99,740% no fluxo bidirecional.

Além disso, outras métricas são avaliadas e são associadas aos modelos de algoritmos. Ademais, a métrica precisão é avaliada pela porcentagem de porção dos positivos reais, comparando-os aos casos que devem ser avaliados como afirmativos. Já a métrica revocação realiza a medição de quantos positivos reais são previstos e a métrica *F1-Score* é uma média ponderada entre as métricas precisão e revocação. Além disso, esses valores foram obtidos por meio da validação cruzada de cada modelo de ML.

A seguir encontram-se os gráficos representados pelas figuras 25 e 26, onde mostra o desempenho da métrica de precisão para a classe legítima dos algoritmos de ML aplicados aos MQTT e CoAP, respectivamente.

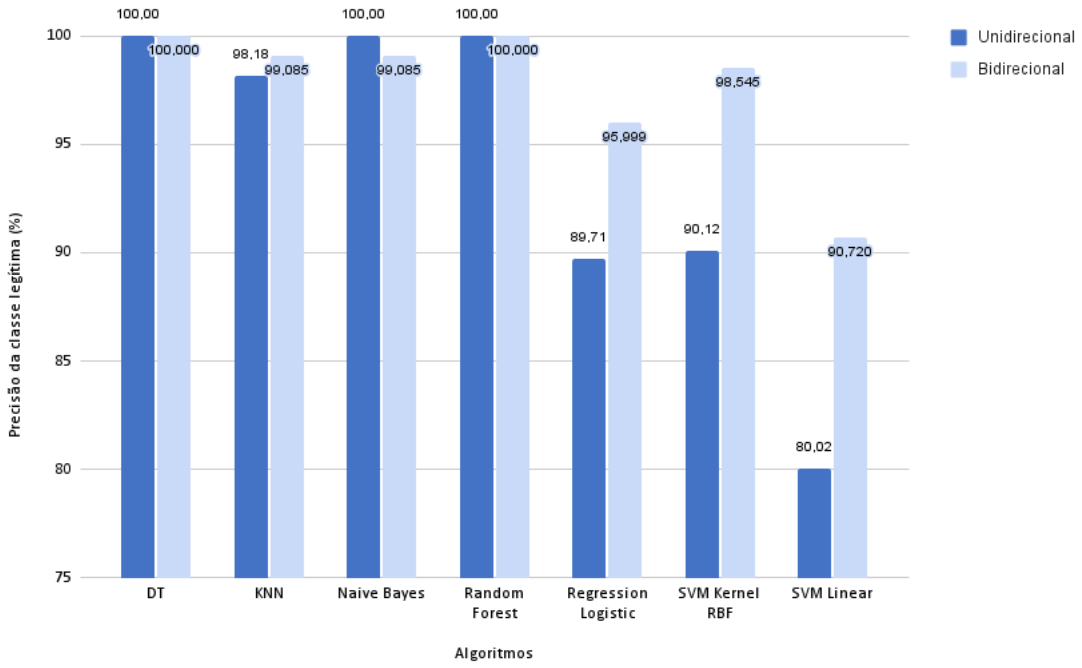


Figura 25 – Precisão dos modelos de ML para classe legítima sob os fluxos unidirecionais e bidirecionais do protocolo MQTT

Fonte: Elaborado pelo autor (2021)

No gráfico representado pela figura 25, visualiza-se uma melhora da métrica precisão do desempenho dos sete algoritmos de ML. Desses, visualiza-se uma eficácia no desempenho do algoritmo *Logistic Regression* que obteve uma melhora no desempenho de 89,71% no fluxo unidirecional para 95,999% no fluxo bidirecional. Neste outro, denominado de SVM com *Kernel* RBF obteve uma melhora no desempenho de 90,12% no fluxo unidirecional para 98,545% no fluxo bidirecional. No SVM *Linear* obteve um aumento de 80,02% no fluxo unidirecional para 90,720% no fluxo bidirecional.

No gráfico representado pela figura 26, visualiza-se uma melhora da métrica precisão do desempenho dos sete algoritmos de ML. Desses, visualiza-se uma eficácia no desempenho do algoritmo *Logistic Regression* que obteve uma melhora no desempenho de 89,137% no fluxo unidirecional para 89,847% no fluxo bidirecional. Neste outro, denominado de SVM com *Kernel* RBF obteve uma melhora no desempenho de 88,281% no fluxo unidirecional para 89,294% no fluxo bidirecional. No SVM *Linear* obteve um aumento de 79,836% no fluxo unidirecional para 90,808% no fluxo bidirecional.

No gráfico representado pela figura 25, observa-se a melhora nos algoritmos quando se aplica para o tráfego bidirecional. Ou seja, ao ser observado pelo trabalho de (HINDY et al., 2020b) chegou-se a essa mesma conclusão com o protocolo MQTT corroborando com os resultados obtidos neste trabalho.

Além disso, nos gráficos representados pelas figuras 27 e 28, encontram-se os resultados

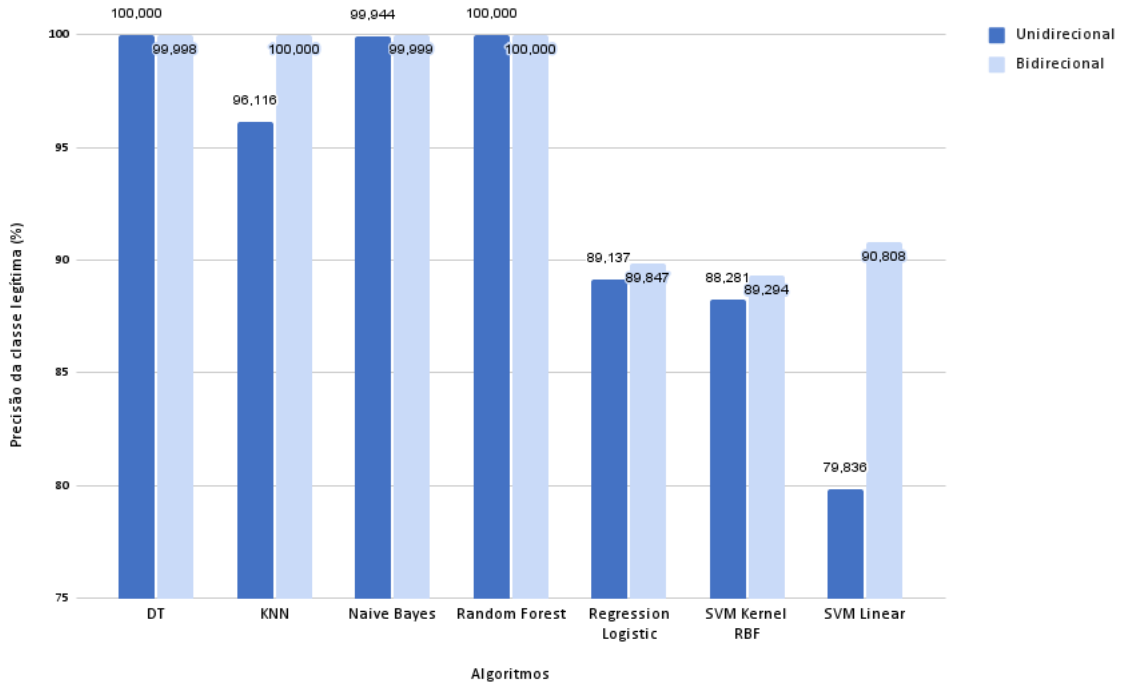


Figura 26 – Precisão dos modelos de ML para classe legítima sob os fluxos unidirecionais e bidirecionais do protocolo CoAP

Fonte: Elaborado pelo autor (2021)

para a classe legítima dos algoritmos de ML da métrica revocação a partir da execução dos modelos de ML para o MQTT e CoAP, respectivamente. Neles é possível visualizar que as métricas utilizadas para avaliar os modelos de ML para o MQTT são similares, porém os dois mostram melhores resultados quando aplicados ao fluxo bidirecional.

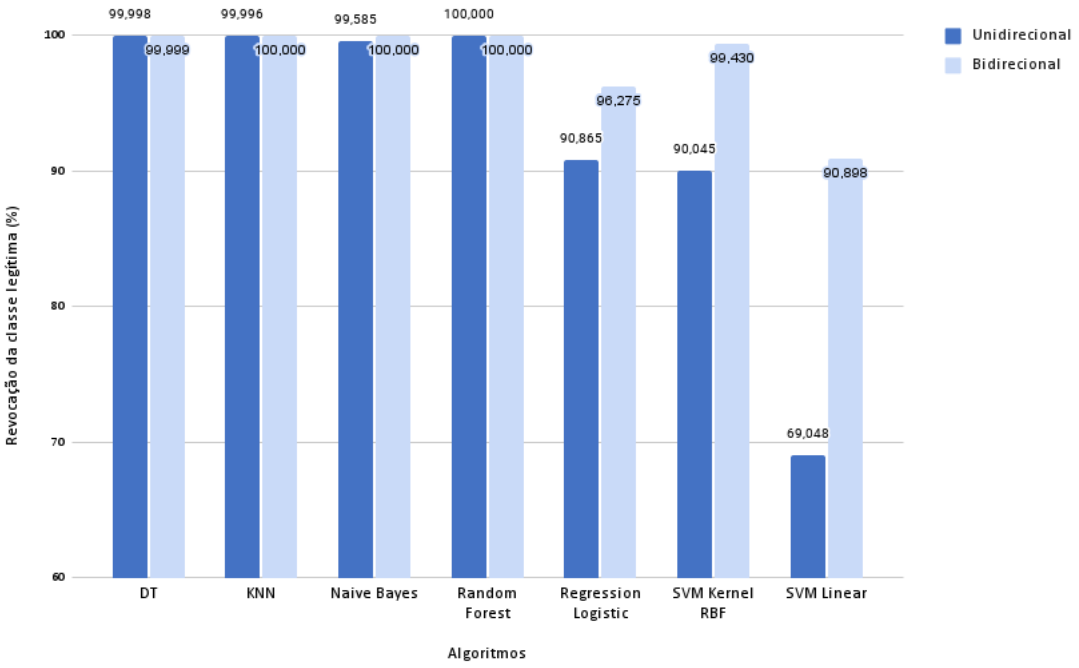


Figura 27 – Métrica Revocação dos modelos de ML para classe legítima sob os fluxos unidirecionais e bidirecionais do protocolo MQTT

Fonte: Elaborado pelo autor (2021)

No gráfico representado pela figura 27, visualiza-se uma melhora da métrica revocação do desempenho dos sete algoritmos de ML. Desses, visualiza-se uma eficácia no desempenho do algoritmo *Logistic Regression* que obteve uma melhora no desempenho de 90,865% no fluxo unidirecional para 96,275% no fluxo bidirecional. Neste outro, denominado de SVM com *Kernel* RBF obteve uma melhora no desempenho de 90,045% no fluxo unidirecional para 99,430% no fluxo bidirecional. No SVM *Linear* obteve um aumento de 69,048% no fluxo unidirecional para 90,898% no fluxo bidirecional.

No gráfico representado pela figura 28, visualiza-se uma melhora da métrica revocação do desempenho dos sete algoritmos de ML. Desses, visualiza-se uma eficácia no desempenho do algoritmo KNN que obteve uma melhora no desempenho de 96,154% no fluxo unidirecional para 100,00% no fluxo bidirecional e no SVM *Linear* obteve uma melhora no desempenho de 86,730% no fluxo unidirecional para 90,899% no fluxo bidirecional.

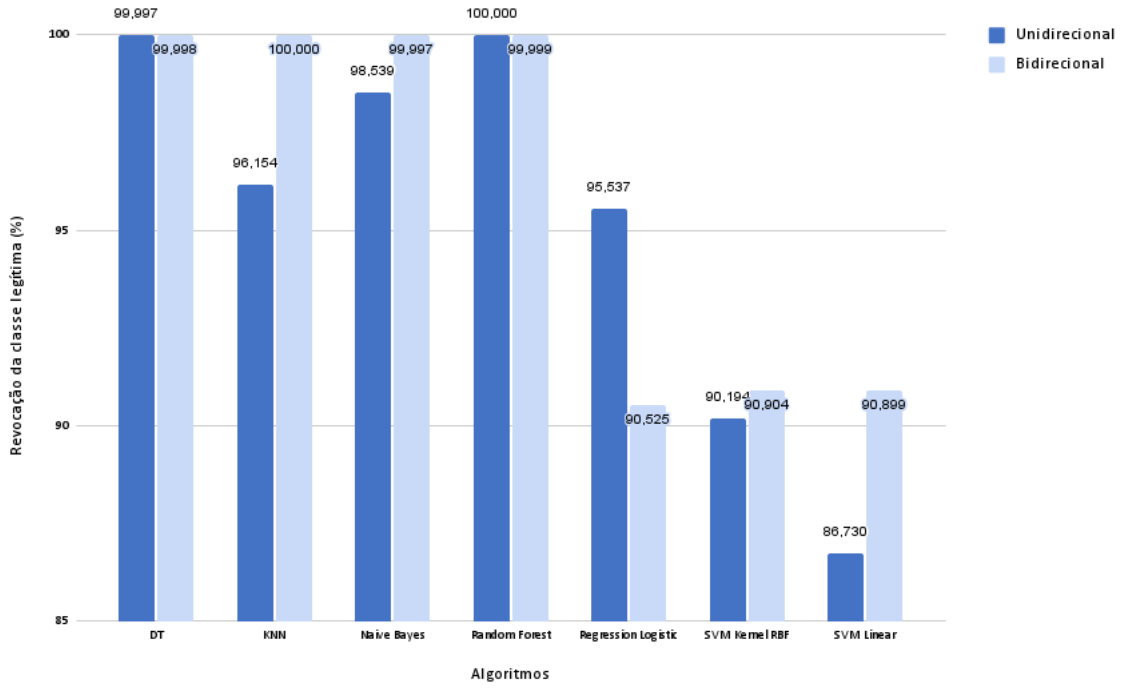


Figura 28 – Métrica Revocação dos modelos de ML para classe legítima sob os fluxos unidirecionais e bidirecionais do protocolo CoAP

Fonte: Elaborado pelo autor (2021)

Nos gráficos, representado pelas figuras 29 e 30, encontram-se os resultados para a classe Sparta da métrica de precisão a partir da execução dos modelos de ML para o MQTT e CoAP, respectivamente.

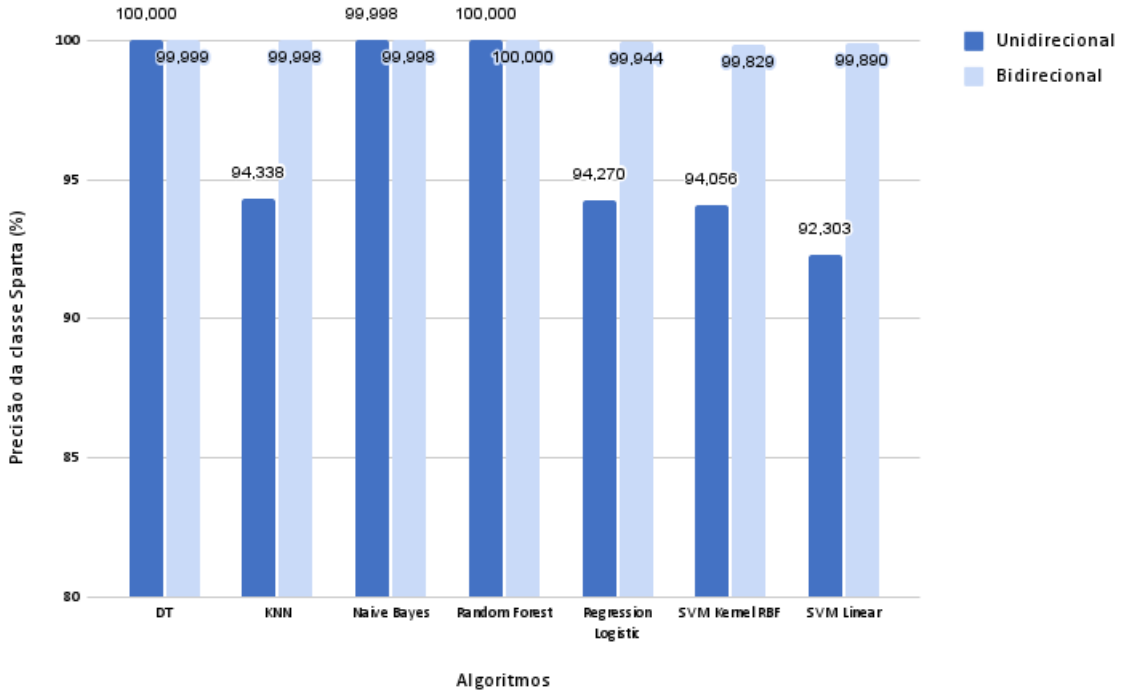


Figura 29 – Métrica Precisão dos modelos de ML para classe Sparta sob os fluxos unidirecionais e bidirecionais do protocolo MQTT

Fonte: Elaborado pelo autor (2021)

No gráfico representado pela figura 29, visualiza-se uma melhora da métrica precisão do desempenho dos sete algoritmos de ML. Desses, visualiza-se uma eficácia no desempenho do algoritmo KNN que obteve uma melhora no desempenho de 94,338% no fluxo unidirecional para 99,998% no fluxo bidirecional. Neste outro, denominado de *Logistic Regression* obteve uma melhora no desempenho de 94,270% no fluxo unidirecional para 99,944% no fluxo bidirecional. No SVM com *Kernel* RBF obteve uma melhora de 94,056% no fluxo unidirecional para 99,829% no fluxo bidirecional e no SVM *Linear* obteve um aumento de 92,303% no fluxo unidirecional para 99,890% no fluxo bidirecional.

No gráfico representado pela figura 30, visualiza-se uma piora da métrica precisão do desempenho dos sete algoritmos de ML. Desses, visualiza-se uma ineficácia no desempenho do algoritmo KNN que obteve uma piora no desempenho de 99,974% no fluxo unidirecional para 94,320% no fluxo bidirecional. Neste outro, denominado de *Logistic Regression* obteve uma piora no desempenho de 99,999% no fluxo unidirecional para 92,757% no fluxo bidirecional. No SVM com *Kernel* RBF obteve uma piora de 95,481% no fluxo unidirecional para 92,926% no fluxo bidirecional e no SVM *Linear* obteve uma piora de 96,034% no fluxo unidirecional para 93,242% no fluxo bidirecional.

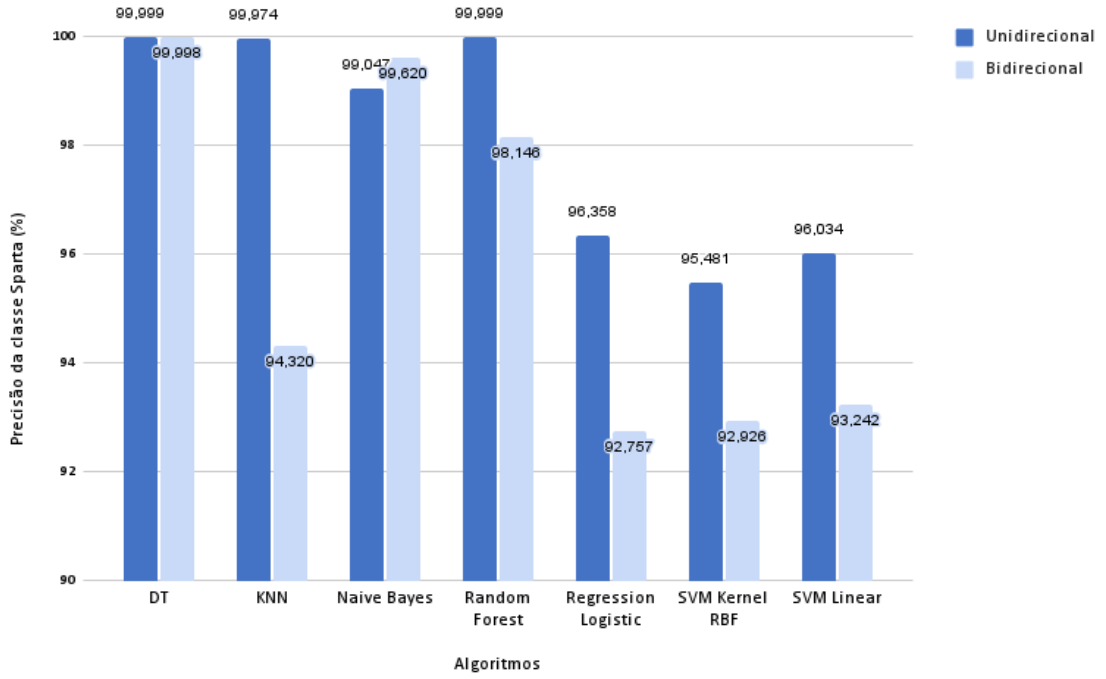


Figura 30 – Métrica Precisão dos modelos de ML para classe Sparta sob os fluxos unidirecionais e bidirecionais do protocolo CoAP

Fonte: Elaborado pelo autor (2021)

Nos gráficos representados pelas figuras 31 e 32, encontram-se os resultados para a classe sparta da métrica Revocação a partir da execução dos modelos de ML para o MQTT e CoAP, respectivamente.

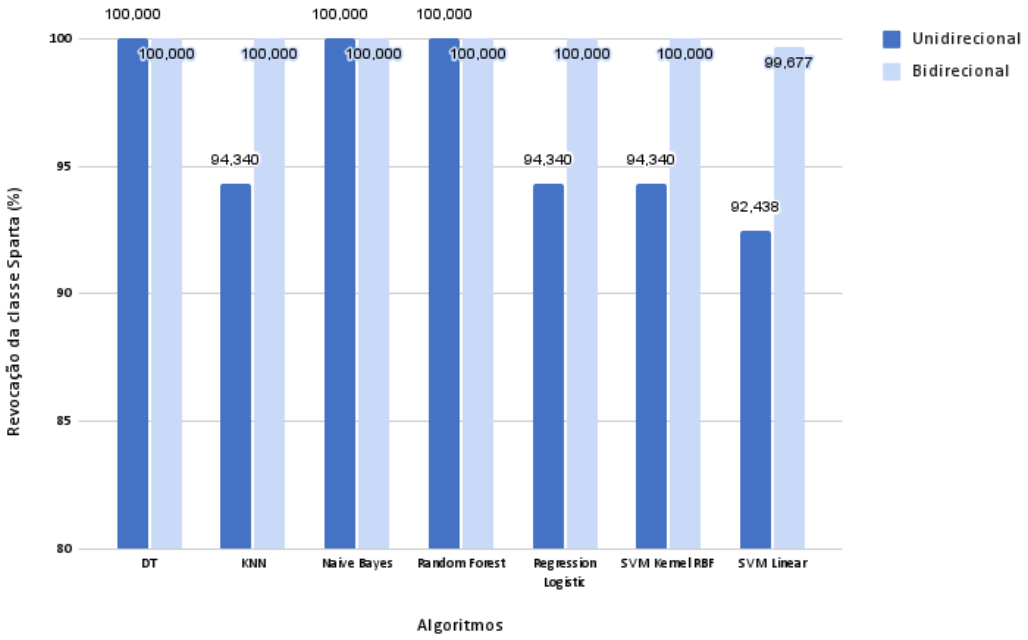


Figura 31 – Métrica Revocação dos modelos de ML para classe Sparta sob os fluxos unidirecionais e bidirecionais do protocolo MQTT

Fonte: Elaborado pelo autor (2021)

No gráfico representado pela figura 31, visualiza-se uma melhora da métrica precisão do desempenho dos sete algoritmos de ML. Desses, visualiza-se uma eficácia no desempenho do algoritmo KNN que obteve uma melhora no desempenho de 94,340% no fluxo unidirecional para 100,00% no fluxo bidirecional. Neste outro, denominado de *Logistic Regression* obteve uma melhora no desempenho de 94,340% no fluxo unidirecional para 100,00% no fluxo bidirecional. No SVM com *Kernel* RBF obteve uma melhora de 94,340% no fluxo unidirecional para 100,00% no fluxo bidirecional e no SVM *Linear* obteve uma aumento de 92,438% no fluxo unidirecional para 99,677% no fluxo bidirecional.

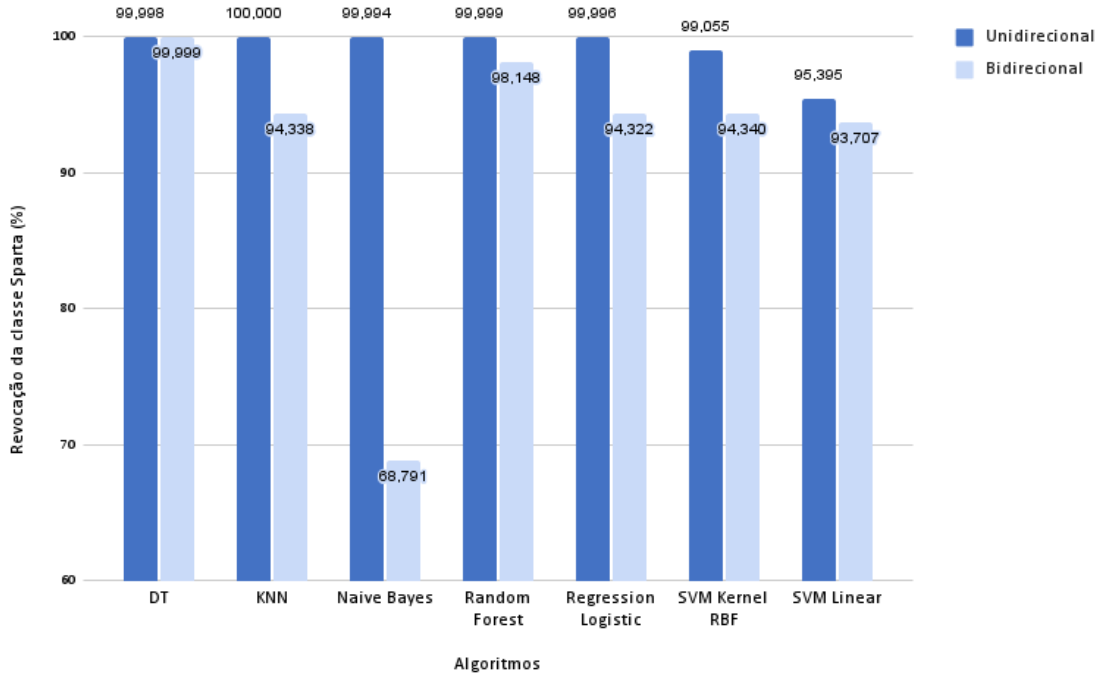


Figura 32 – Métrica Revocação dos modelos de ML para classe Sparta sob os fluxos unidirecionais e bidirecionais do protocolo CoAP

Fonte: Elaborado pelo autor (2021)

No gráfico representado pela figura 32, visualiza-se uma piora da métrica revocação do desempenho dos sete algoritmos de ML. Desses, visualiza-se uma ineficácia no desempenho do algoritmo KNN que obteve uma piora no desempenho de 100,00% no fluxo unidirecional para 94,338% no fluxo bidirecional. Neste outro, denominado de *Naive Bayes* obteve uma piora no desempenho de 99,994% no fluxo unidirecional para 68,791% no fluxo bidirecional. No *Logistic Regression* obteve uma piora de 99,996% no fluxo unidirecional para 94,322% no fluxo bidirecional. No SVM *Linear* obteve uma aumento de 95,395% no fluxo unidirecional para 93,707% no fluxo bidirecional.

No gráfico a seguir representado pela figura 33, encontra-se o desempenho dos algoritmos de ML para a métrica da precisão do peso médio aplicados ao protocolo MQTT.

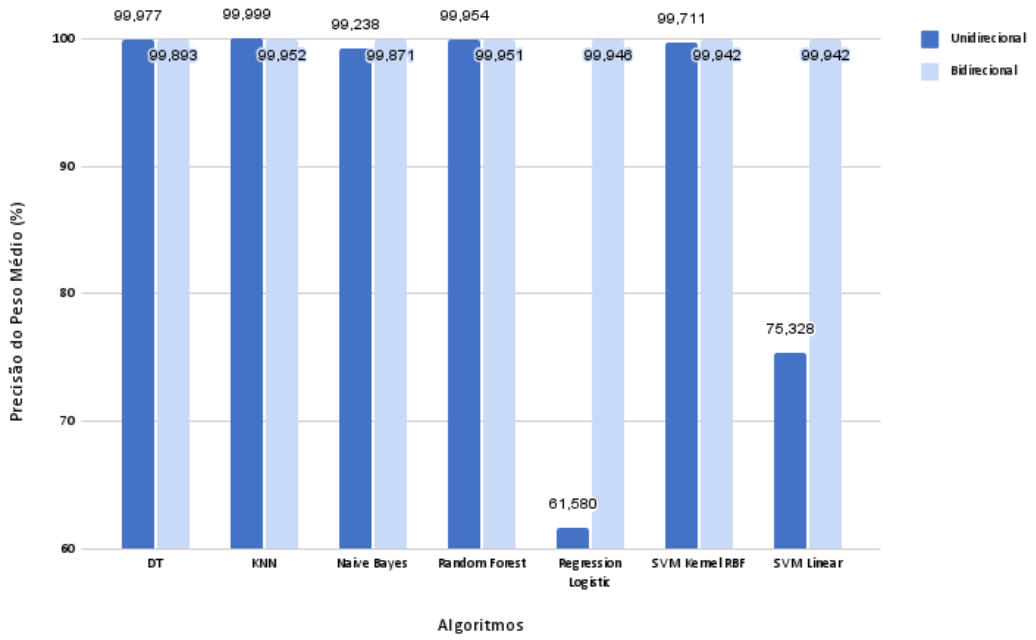


Figura 33 – Métrica Precisão dos modelos de ML para o Peso Médio sob os fluxos unidirecionais e bidirecionais do protocolo MQTT

Fonte: Elaborado pelo autor (2021)

No gráfico representado pela figura 33, visualiza-se uma melhora da métrica precisão do desempenho dos sete algoritmos de ML. Desses, visualiza-se uma eficácia no desempenho do algoritmo *Logistic Regression* que obteve uma melhora no desempenho de 61,580% no fluxo unidirecional para 99,946% no fluxo bidirecional e no SVM *Linear* obteve uma aumento de 75,328% no fluxo unidirecional para 99,942% no fluxo bidirecional.

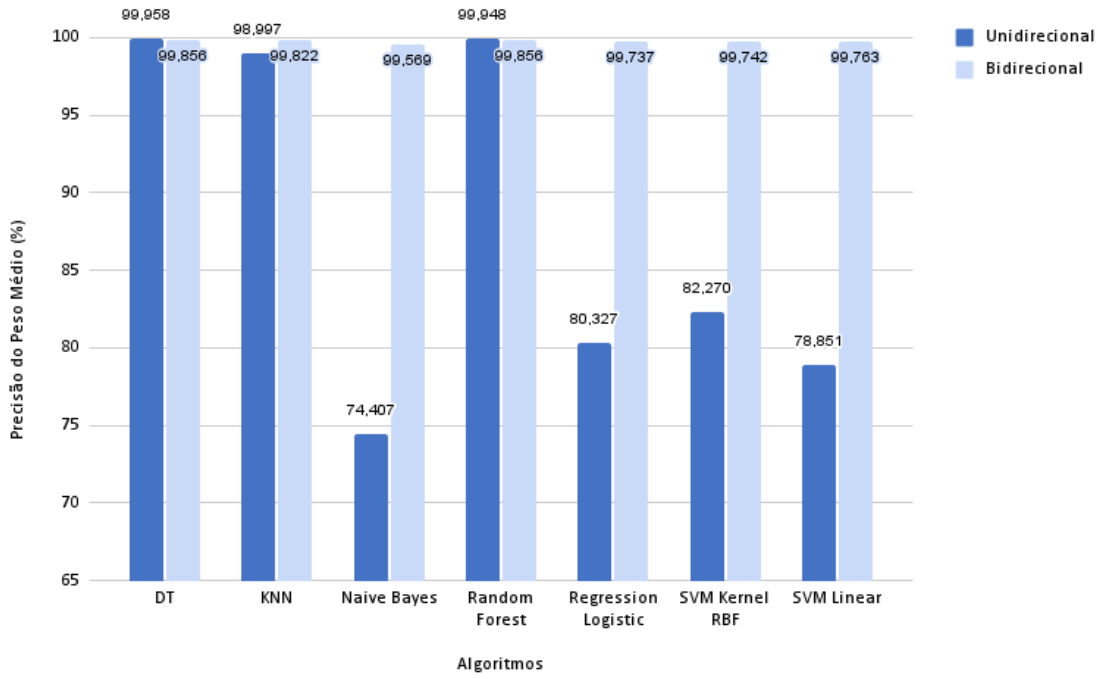


Figura 34 – Métrica Precisão dos modelos de ML para o Peso Médio sob os fluxos unidirecionais e bidirecionais do protocolo CoAP

Fonte: Elaborado pelo autor (2021)

No gráfico representado pela figura 34 observa-se um aumento considerável do algoritmo *Naive Bayes* comparando-o com os outros algoritmos. Esse algoritmo mostrou uma elevação de desempenho de 74,407% quando se aplicou o conjunto de dados representando o fluxo unidirecional e 99,569% quando se aplicou o conjunto de dados representando o fluxo bidirecional do protocolo CoAP.

Nos gráficos representados pelas figuras 35 e 36 apresentam os desempenhos dos algoritmos de ML para a métrica revocação para os protocolos MQTT e CoAP.

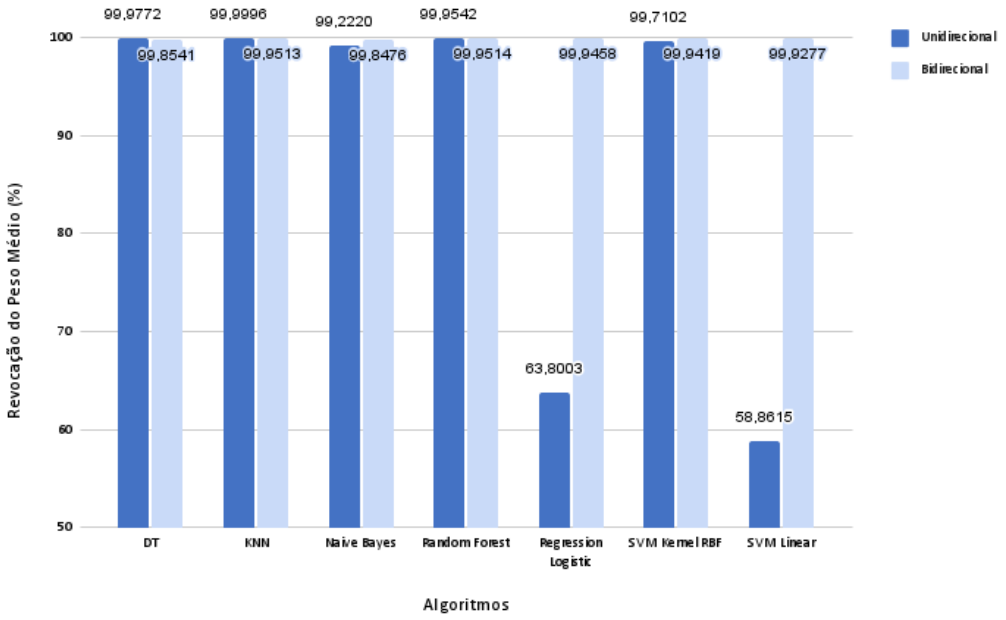


Figura 35 – Métrica Revocação dos modelos de ML para o Peso Médio sob os fluxos unidirecionais e bidirecionais do protocolo MQTT

Fonte: Elaborado pelo autor (2021)

No gráfico representado pela figura 35 observa-se um aumento do desempenho do algoritmo SVM *Linear*, quando aplicou-se o *dataset* do fluxo bidirecional, houve um aumento de 58,8515% para 99,9277%.

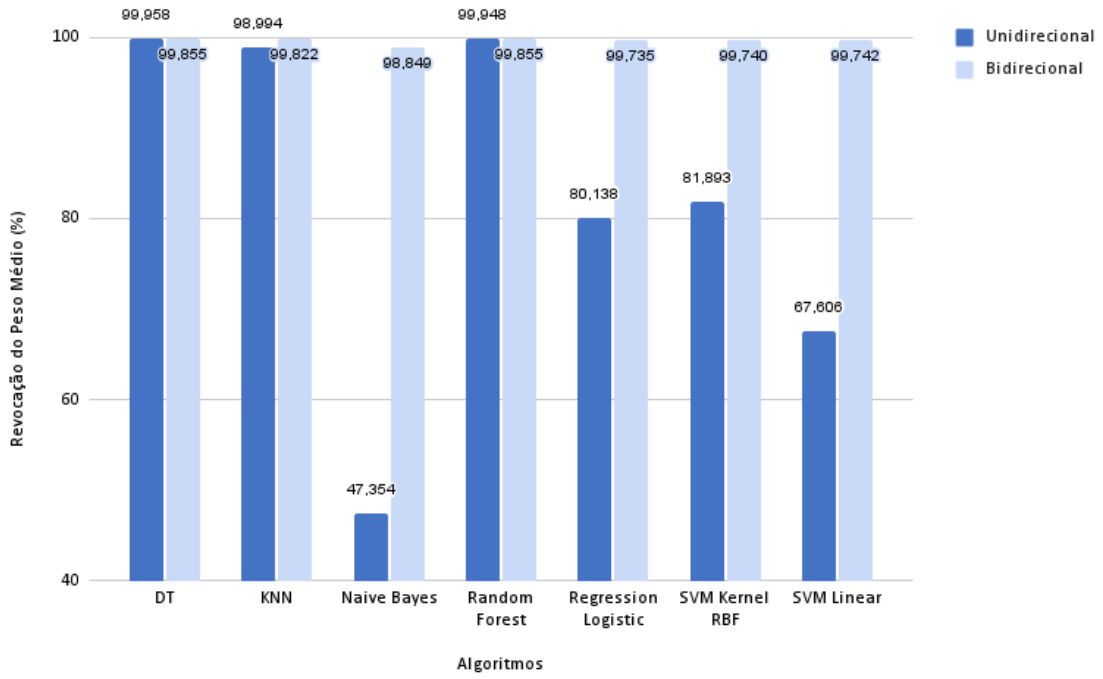


Figura 36 – Métrica Revocação dos modelos de ML para o Peso Médio sob os fluxos unidirecionais e bidirecionais do protocolo CoAP

Fonte: Elaborado pelo autor (2021)

No gráfico representado pela figura 36 observa-se um aumento considerável de desempenho do algoritmo *Naive Bayes* quando aplicou-se o *dataset* do fluxo bidirecional, visualiza-se que houve um aumento de 47,364% no fluxo unidirecional para 98,849% no fluxo bidirecional.

4.2 RESULTADOS DO CONSUMO DE ENERGIA

As medições de energia foram realizadas na fase de treinamento e testes dos algoritmos de ML (*Decision Tree, KNN, Naive Bayes, Random Forests, Logistic Regression , SVM Linear* e SVM com *Kernel* RBF), isso, para contribuir com o desenvolvimento de um conjunto de IDS com troca de regras entre si baseados em semântica e colaboração, assim, o escopo desses resultados limitou-se somente ao consumo de energia durante o treino e avaliação dos modelos de ML.

Os resultados são representados por meio de gráficos, divididos nos protocolos MQTT e CoAP. Também, neste trabalho, utilizou-se a medida de tempo *timestamp*, onde sua variação é representada por meio da medida da unidade segundos (s) para os gráficos relacionados ao tempo de execução, ademais, foi utilizado a unidade microjoule uJ para os demais gráficos para representar o consumo de energia.

Além disso, os gráficos contêm duas informações e são classificados por elementos de hardware (Núcleo do processador, Memória RAM, Processador da placa de vídeo e *Package* do Processador), essa informações são: as médias dos tempos de execução e do consumo de energia dos algoritmos e os intervalos de confiança do consumo de energia e do tempo de execução dos algoritmos de ML. Eles foram aplicados aos protocolos MQTT e CoAP para os fluxos unidirecionais e bidirecionais. Ademais, aplica-se o intervalo de confiança de 95% para estimar a precisão dos dados.

Ainda assim, essas medições foram realizadas durante a execução dos algoritmos de modelos ML, o código-fonte do script foi adaptado com a implementação do RAPL (*Running Average Power Limit*) por meio da biblioteca PyJoules (INRIA, 2019) na linguagem de programação Python.

Ademais, evitou-se dados *outliers* severos, pois, retirou-se alguns dados iniciais para evitar variações de extremos de dados, porém essas informações encontram-se publicadas em sua íntegra no repositório github deste trabalho [1].

4.2.1 Resultados do tempo de execução dos algoritmos

Os resultados apresentados nesta seção representam a estimativa de consumo de energia virtual durante a execução dos algoritmos de ML. Ademais, os gráficos apresentam os resultados comparativos entre os protocolos MQTT e CoAP. Além disso, o resultado para os fluxos unidirecionais e bidirecionais, também, são apresentados.

As figuras 37 apresentam os resultados para os algoritmos de ML aplicados ao fluxo unidirecional. Esses gráficos são divididos para o protocolo MQTT (a) e CoAP (b). O gráfico (a) apresenta os tempos médios de execução dos algoritmos de ML para o protocolo MQTT (a) e o CoAP (b).

[1] https://github.com/moronivieira/IDS-CoAP

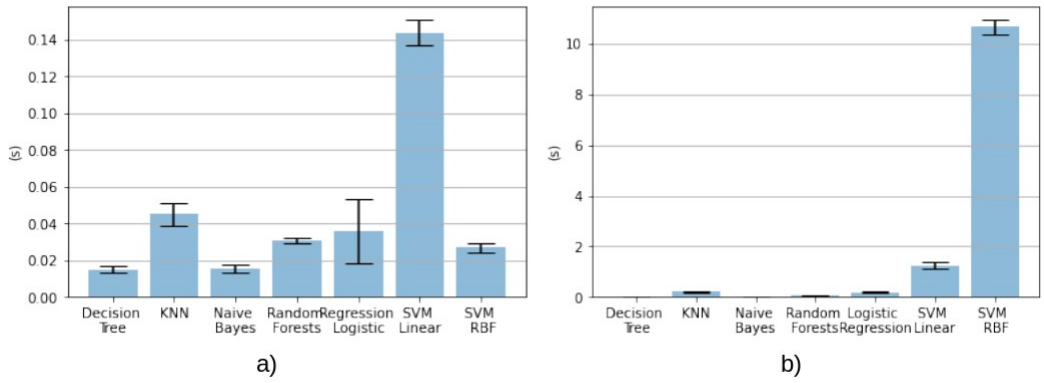


Figura 37 – Média e Intervalo de Confiança do Tempo de Execução dos Algoritmos de ML do Fluxo Unidirecional

Fonte: Elaborado pelo autor (2021)

Os gráficos representados pela figura 37, mostram um tempo de execução diferentes entre os dois protocolos, ou seja, no gráfico (a) a maior média de tempo de execução ocorreu na execução do algoritmo SVM *Linear*, já no gráfico (b), a maior média de tempo de execução ocorreu com o algoritmo SVM com *Kernel* RBF. Ademais, observa-se que o menor tempo médio de execução dos algoritmos de ML para o protocolo MQTT foi o algoritmo *Decision Tree*, já para o protocolo CoAP a menor média de tempo de execução foi obtida pelos protocolos *Decision Tree, Naive Bayes e Random Forests*.

A seguir são apresentados os gráficos representados pela figura 38, que mostra os resultados do consumo dos algoritmos de ML sob o fluxo bidirecional do MQTT e CoAP.

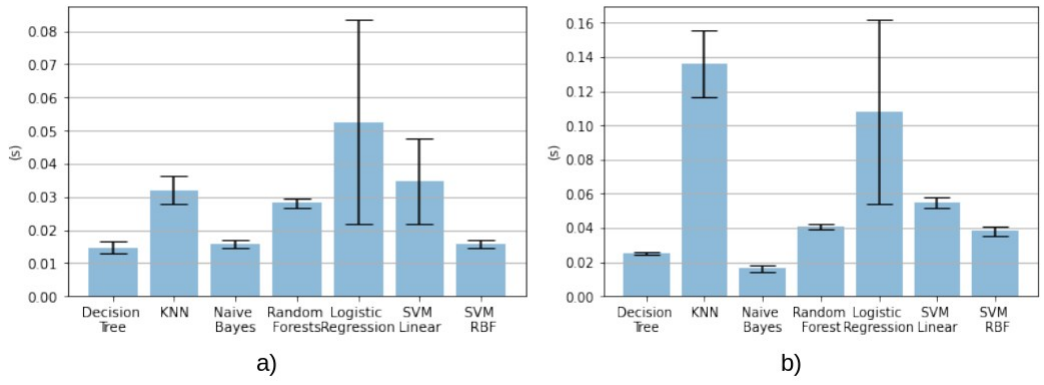


Figura 38 – Média e Intervalo de Confiança do Tempo de Execução dos Algoritmos de ML do Fluxo Bidirecional

Fonte: Elaborado pelo autor (2021)

No gráfico representado pela figura 38 (a) mostra os resultados obtidos do protocolo MQTT, ou seja, nele observa-se que o algoritmo de *Logistic Regression* alcançou a maior média

no tempo de execução dos algoritmos se comparados com os outros. Ainda, a menor média do tempo de execução dos algoritmos ocorreu no *Decision Tree* e os algoritmos *Naive Bayes* e SVM com *Kernel* RBF obtiveram os segundos lugares, respectivamente. Além disso, o gráfico representado pela figura 38 (b), observa-se os resultados para o protocolo CoAP, e, também, visualiza-se que o algoritmo KNN obteve a maior média de tempo de execução, seguido pelo algoritmo *Logistic Regression* e SVM *Linear*. Ainda, observa-se que a maior variação de intervalo de confiança foi obtida no algoritmo *Logistic Regression* seguida pelo KNN.

4.2.2 Consumo de energia do núcleo processador

Os gráficos representados pela figura 39 mostram a média e o desvio padrão do consumo de energia do núcleo do processador para os fluxos unidirecionais (a) e bidirecionais (b)

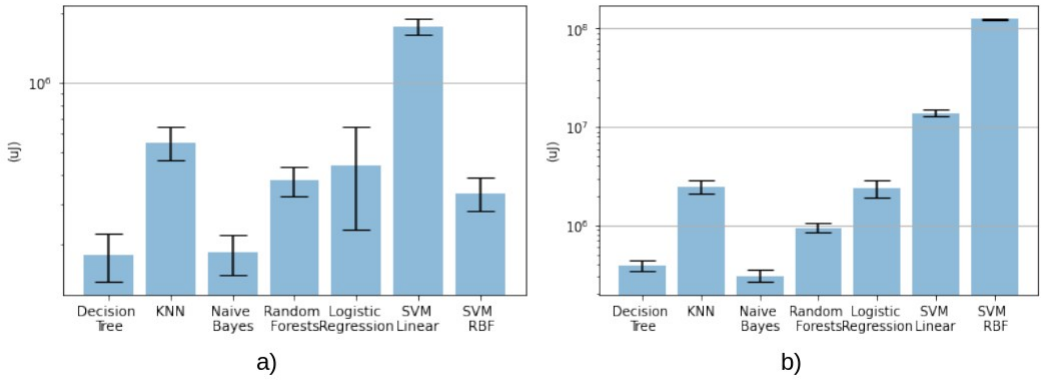


Figura 39 – Média e Intervalo de Confiança da Execução dos Algoritmos de ML no Fluxo Unidirecional sob o Núcleo do Processador

Fonte: Elaborado pelo autor (2021)

No gráfico representado pela figura 39 (a), observa-se que o maior consumo médio foi do algoritmo SVM *Linear*, já no gráfico 39 (b) mostra que o maior consumo médio foi realizado pelo algoritmo SVM com *Kernel* RBF.

A seguir visualiza-se o gráfico representado pela figura 40, que mostra os resultados do consumo de energia do núcleo do processador para os fluxos bidirecionais do protocolo MQTT (a) e CoAP (b).

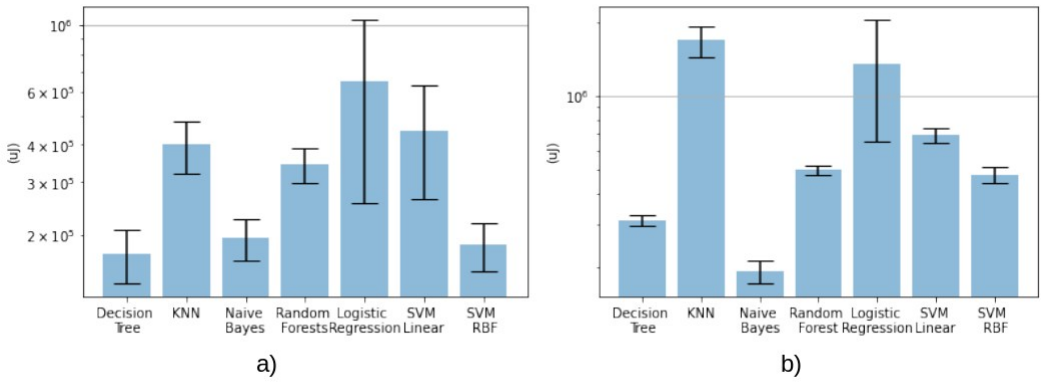


Figura 40 – Média e Intervalo de Confiança da Execução dos Algoritmos de ML no Fluxo Bidirecional sob o Núcleo do Processador

Fonte: Elaborado pelo autor (2021)

O gráfico representado pela figura 40 (a) mostra que o maior consumo médio de energia foi realizado pelo algoritmo de *Logistic Regression*, observa-se, também, que o maior consumo médio no cenário com CoAP foi realizado pelo algoritmo KNN. Além disso, observa-se que a maior variação do intervalo de confiança ocorreu no algoritmo *Logistic Regression* para os dois cenários. Ademais, os algoritmos que obtiveram a menor média de consumo energético para o protocolo MQTT foram: *Decision Tree, Naive Bayes* e SVM com *Kernel* RBF. Já para os algoritmos aplicados ao protocolo CoAP foi o *Naive Bayes* e o *Decision Tree*.

4.2.3 Consumo de energia do package

Os gráficos a seguir, representados pelas figuras 41 e a figura 42, mostram a média e o desvio padrão do consumo de energia do *package* do processador para o fluxo unidirecional dos protocolos MQTT e CoAP.

No gráfico representado pela figura 41 (a), observa-se maior consumo médio foi do algoritmo SVM *Linear* para o MQTT, já para o CoAP 41 (b) observa-se que o maior consumo médio foi realizado pelo algoritmo SVM com *Kernel* RBF. Além disso, observa-se que o algoritmo *Decision Tree* obteve a menor média de consumo de energia para o protocolo MQTT. Já o protocolo *Naive Bayes* obteve a menor média de consumo para o protocolo CoAP.

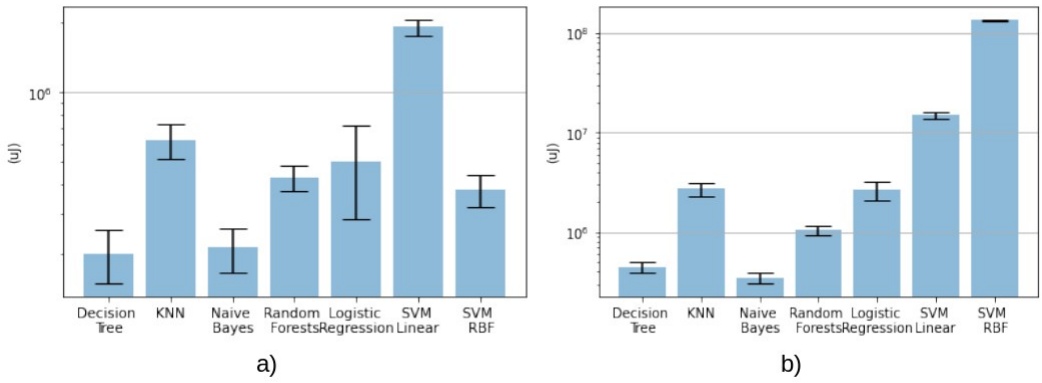


Figura 41 – Média e Intervalo de Confiança da Execução dos Algoritmos de ML no Fluxo Unidirecional sob o Package do Processador

Fonte: Elaborado pelo autor (2021)

A seguir são apresentados os gráficos representados na figura 42 que apresenta os resultados para o fluxo bidirecional do MQTT e CoAP.

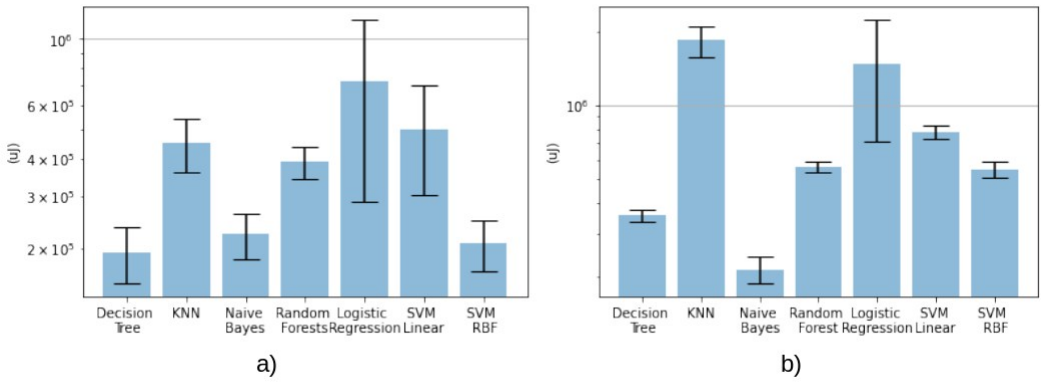


Figura 42 – Média e Intervalo de Confiança da Execução dos Algoritmos de ML no Fluxo Bidirecional sob o Package do Processador

Fonte: Elaborado pelo autor (2021)

No gráfico representado pela figura 42 (a), observa-se que a maior média de consumo de energia foi realizada pelo algoritmo de *Logistic Regression* para o cenário com o protocolo MQTT. Outrossim, no gráfico representado pela figura 42, observa-se que o maior consumo médio foi obtido pelo algoritmo KNN, seguido do algoritmo *Logistic Regression*. Também, visualiza-se que a maior variação do intervalo de confiança foi o do algoritmo *Logistic Regression* entre os dois protocolos. Outrossim, observa-se que o algoritmo *Decision Tree* obteve a menor média de consumo de energia para o MQTT e o *Naive Bayes* obteve a menor média de consumo para o CoAP.

4.2.4 Consumo de energia da memória RAM

Os gráficos representados pelas figuras 43 e 44 mostram a média e o desvio padrão do consumo de energia de memória RAM para os fluxos unidirecionais (a) e bidirecionais (b).

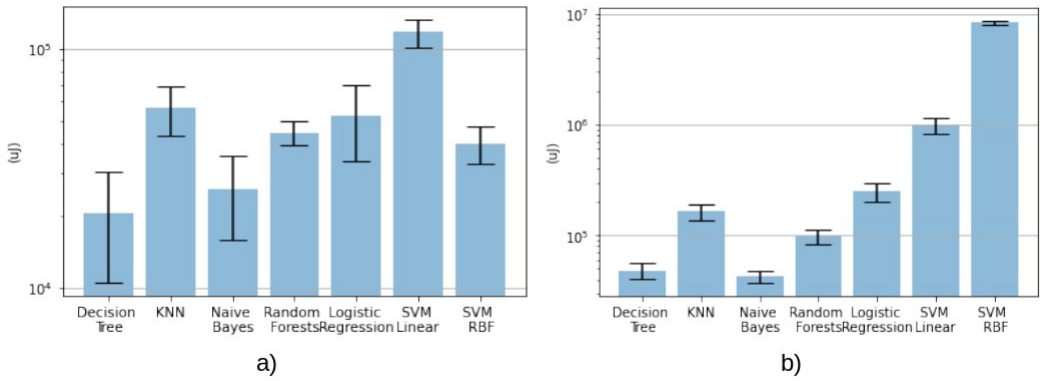


Figura 43 – Média e Intervalo de Confiança da Execução dos Algoritmos de ML no Fluxo Unidirecional sob a Memória RAM

Fonte: Elaborado pelo autor (2021)

No gráfico representado pela figura 43 (a), observa-se maior consumo médio foi do algoritmo SVM *Linear* para o MQTT, já para o gráfico do CoAP 43 (b), observa-se que o maior consumo médio foi realizado pelo algoritmo SVM com *Kernel* RBF. Ainda, é mostrado no gráfico que o algoritmo *Decision Tree* obteve a menor média de consumo de energia para o MQTT e o algoritmo *Naive Bayes* obteve a menor média de consumo para o CoAP.

A seguir são apresentados os gráficos representados pela figura 44 e apresentam os resultados para o fluxo bidirecional do MQTT e CoAP do consumo de energia da memória RAM.

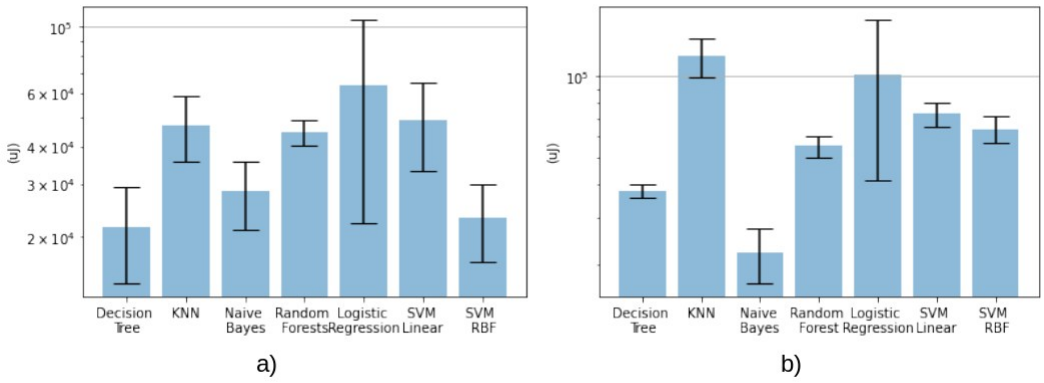


Figura 44 – Média e Intervalo de Confiança da Execução dos Algoritmos de ML no Fluxo Bidirecional sob a Memória RAM

Fonte: Elaborado pelo autor (2021)

No gráfico representado pela figura 44 (a), observa-se que a maior média de consumo foi do algoritmo de *Logistic Regression* para o cenário com o protocolo MQTT, já para o protocolo CoAP representado pela figura 44 (b), observa-se que o maior consumo médio foi realizado pelo algoritmo KNN. Além disso, visualiza-se que a maior variação do intervalo de confiança nos dois cenários foi obtida pelo algoritmo de *Logistic Regression*. Também, visualiza-se nos gráficos que o algoritmo *Decision Tree* obteve a menor média de consumo de energia para o MQTT e o *Naive Bayes* obteve a menor média de consumo para o CoAP.

4.2.5 Consumo de energia da GPU

Os gráficos representados pelas figuras 45 e 46 mostram a média e o desvio padrão do consumo de energia da GPU (*Graphics Processing Unit*) para os fluxos unidirecionais (a) e bidirecionais (b) dos protocolos MQTT e CoAP.

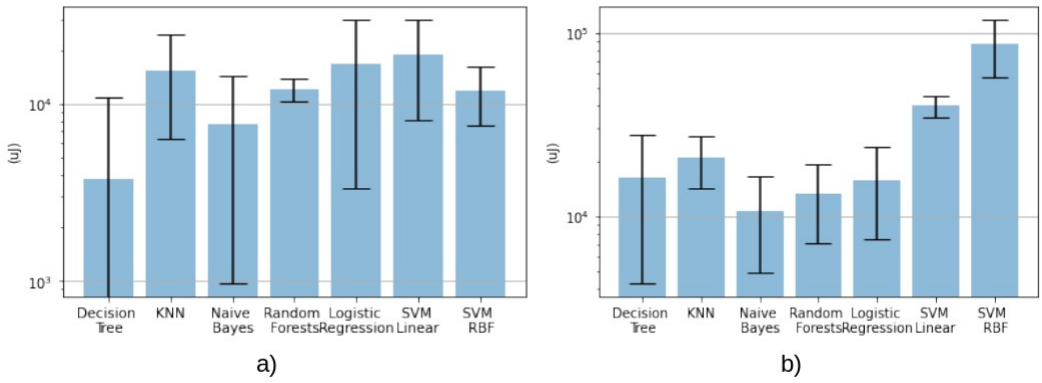


Figura 45 – Média e Intervalo de Confiança da Execução dos Algoritmos de ML no Fluxo Unidirecional sob a GPU

Fonte: Elaborado pelo autor (2021)

No gráfico representado pela figura 45 (a), observa-se maior consumo médio foi do algoritmo SVM *Linear* para o MQTT, já para o CoAP 45 (b), observa-se que o maior consumo médio foi realizado pelo algoritmo SVM com *Kernel* RBF. Além disso, visualiza-se que há variações substanciais nos algoritmos executados sob o protocolo MQTT. Ademais, observa-se que o algoritmo *Decision Tree* obteve a menor média de consumo de energia para o MQTT e o *Naive Bayes* obteve a menor média de consumo para o CoAP.

A seguir são apresentados os gráficos representados pela figura 46, que mostra os resultados para o fluxo bidirecional do MQTT e CoAP.

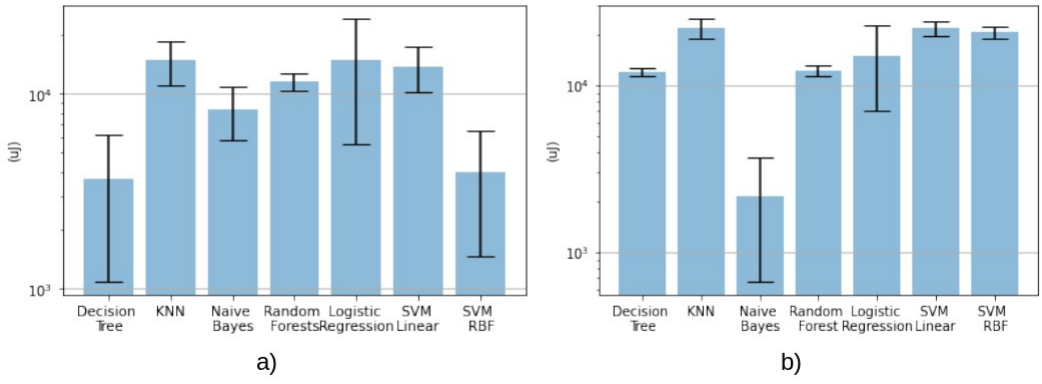


Figura 46 – Média e Intervalo de Confiança da Execução dos Algoritmos de ML no Fluxo Bidirecional sob a GPU

Fonte: Elaborado pelo autor (2021)

No gráfico representado pela figura 46 (a), observa-se que as maiores médias de consumo foram dos algoritmos de *Logistic Regression* e KNN para o cenário com o protocolo MQTT, já para o protocolo CoAP representado pela figura 46 (b), observa-se que maior consumo médio foi

realizado pelo algoritmo SVM com Kernel RBF. Além disso, visualiza-se que a maior variação do intervalo de confiança nos dois cenários foi obtida pelo algoritmo de *Decision Tree*. Ainda, observa-se que o algoritmo *Decision Tree* obteve a menor média de consumo de energia para o MQTT e o *Naive Bayes* obteve a menor média de consumo para o CoAP.

5 CONCLUSÃO

A pesquisa aplicada neste trabalho motivou-se pela pesquisa exploratória, a qual identificou baixos números de trabalhos práticos sobre o contexto IoT e IA. A partir disso, também, determinou-se os elementos para a condução desta pesquisa. Dessa forma, dirigiu-se esta pesquisa com base em um trabalho relacionado ao tema deste projeto.

Além disso, algumas ferramentas e recursos foram avaliados para o protocolo CoAP, já nos métodos e ferramentas explorou-se o desenvolvimento de um cenário virtual para obtenção de dados do CoAP. Ademais, foram avaliados os ataques cibernéticos: *Scan Agressive*, *Scan* UDP, *Sparta* e Força Bruta no contexto de dispositivos IoT. Ainda, os algoritmos supervisionados de ML: *Logistic Regression, k-Nearest Neighbours, Gaussian Naive Bayes, Decision Trees, Random Forests, Support Vector Machine (linear and RBF kernel)* foram avaliados para melhorar a detecção de ataques no ambiente IoT.

Os algoritmos de ML foram utilizados para detectar ataques e foram otimizados por meio da escolha das características dentro dos conjuntos de dados. Também, a escolha dos valores dos hiperparâmetros de otimização nos modelos de ML e a quantidade de execuções dos algoritmos foi expandida para melhorar essa otimização.

Também, foi incluída a mensuração do consumo de energia dos componentes de hardware (*Package*, CPU, GPU e Memória RAM) na fase de aprendizado e testes dos modelos de ML, adaptando-se à aplicação. A partir disso, gerou-se informações de estimativa de consumo de energia dos componentes físicos do computador para uma distribuição de consumo de energia e distribuição de regras baseadas em semântica no contexto de IDS distribuído.

Seguindo, o resultado dessas medições abordaram as informações obtidas na fase de execução dos experimentos. As avaliações consideraram as métricas obtidas por meio de duas revisões sistemáticas (OLIVEIRA et al., 2020) e anexo D. Dessa forma, observou-se os valores das métricas acurácia, revocação, precisão e F1-*score* obtendo aumento dos valores dessa métrica dos algoritmos em quase todos os cenários de ataques analisados, assim, esse tipo de abordagem de IDS mostrou-se eficiente.

Dessa forma, os resultados experimentais mostraram que os fluxos bidirecionais são adequados para obter um ótimo desempenho nos algoritmos por causa de suas características, pois, obteve-se aumento na média ponderada em todos os cenários. Dessa forma, obteve-se aumento da métrica revocação na média ponderada no MQTT de $\sim$99,7% para $\sim$99,9%, já no CoAP ocorreu o aumento de $\sim$81,9% para $\sim$99,7%, nos fluxos unidirecionais e bidirecionais, respectivamente. Ademais, a média ponderada da métrica precisão para o MQTT houve um aumento de $\sim$99,7% para $\sim$99,9% e no CoAP houve $\sim$82,3% para $\sim$99,8%, nos fluxos unidirecionais e bidirecionais, respectivamente.

Assim, conclui-se que no aspecto de detecção de ataques por meio de algoritmos de ML para classificação entre ataques e tráfego legítimo obteve-se uma eficiência satisfatória para MQTT e CoAP quando aplicados no fluxos bidirecionais, por ora, os resultados mostraram que, de acordo com os modelos de ML, é possível realizar a detecção das anomalias em um cenário IoT que utiliza esses protocolos. Dessa forma, constatou-se que os futuros projetos, envolvendo a utilização dos protocolos MQTT e CoAP com sistemas de detecção de intrusão baseados em ML, deverão ser planejados para obter dados dos fluxos bidirecionais dos dois protocolos.

Ainda, realizou-se uma avaliação do consumo de energia nos componentes de hardware: CPU, *Package*, GPU e Memória RAM. Tal avaliação dos hardwares gerou uma estimativa de consumo energético durante a fase de aprendizagem e testes dos algoritmos de ML. Essa contribuição foi feita neste trabalho para uma perspectiva de divisão de consumo energético entre IDS e troca de regras de filtragem entre eles.

Com isso, verificou-se que os algoritmos *Decision Tree, Naive Bayes, Logistic Regression, Random Forests* e KNN obtiveram as menores médias de tempo de execução para os algoritmos de ML aplicados para o fluxo unidirecional no CoAP alcançando média de tempo entre 0,01s e 0,05s.

Além disso, verificou-se, ainda, que os mesmos algoritmos de ML, no protocolo MQTT no fluxo unidirecional, obtiveram as menores médias de tempo de execução apresentando valores menores do que 1s. Ainda, no núcleo do processador, o consumo de energia do fluxo unidirecional com o protocolo MQTT ao aplicar o algoritmo *Decision Tree* obteve uma média de consumo menor do que 10^6 uJ, alcançando o menor consumo de energia em detrimento à comparação do consumo de energia dos outros protocolos analisados. Por fim, no protocolo CoAP foi o *Naive Bayes* que também obteve uma média de consumo menor que 10^6 uJ.

Ademais, para a memória RAM no fluxo unidirecional no protocolo MQTT o algoritmo que obteve a menor média de consumo de energia foi o *Decision Tree*, que ficou abaixo de 10^5 uJ e abaixo de todas as médias entre os algoritmos. Já no protocolo CoAP, o menor consumo de energia foi obtido pelo algoritmo Naive Bayes, que obteve uma média menor que 10^5 uJ e foi menor que os outros algoritmos em detrimento aos protocolos analisados.

Continuando, observou-se que o algoritmo que obteve a menor média de consumo na GPU, aplicando-se o fluxo unidirecional no MQTT foi o *Decision Tree*, cuja média de consumo era menor que 10^4 uJ. Já no CoAP o algoritmo que obteve o menor consumo foi o *Naive Bayes* com um valor um pouco acima de 10^4 uJ.

Outrossim, observou-se que no componente de hardware *Package*, aplicando-se o fluxo unidirecional para o MQTT o algoritmo que obteve a menor média de consumo de energia foi o *Decision Tree*, obtendo um valor abaixo de 10^5 uJ. Já para o CoAP o algoritmo *Naive Bayes* obteve a menor média de consumo, obtendo um valor de média abaixo de 10^6 uJ.

Bem como, verificou-se que os algoritmos: *Decision Tree, Naive Bayes*, SVM com *kernel*

RBF, *Random Forests* e KNN obtiveram as menores médias de tempo de execução para os algoritmos de ML aplicados para o fluxo bidirecional no MQTT alcançando médias de tempo entre 0,01s e 0,04s. Já para o CoAP no fluxo bidirecional, os algoritmos que obtiveram as menores médias de tempos de execução foram os algoritmos: *Naive Bayes, Decision Tree* e SVM com *Kernel* RBF com valores menores do que 0,04s.

Ainda, no núcleo do processador, o consumo de energia do fluxo bidirecional com o protocolo MQTT ao aplicar o algoritmo *Decision Tree* obteve uma média de consumo menor do que $2x10^5$ uJ, alcançando o menor consumo de energia em detrimento à comparação do consumo de energia dos outros protocolos analisados. Por fim, no protocolo CoAP foi o *Naive Bayes* que também obteve uma média de consumo menor que 10^6 uJ.

Ademais, para a memória RAM no fluxo bidirecional no protocolo MQTT o algoritmo que obteve a menor média de consumo de energia foi o *Decision Tree*, que ficou abaixo de de $3x10^4$ uJ e abaixo de todas as médias entre os algoritmos. Já no protocolo CoAP, o menor consumo de energia foi obtido pelo algoritmo *Naive Bayes*, que obteve uma média menor que 10^5 uJ e foi menor que os outros algoritmos em detrimento aos protocolos analisados.

Continuando, observou-se que o algoritmo que obteve a menor média de consumo na GPU, aplicando-se o fluxo unidirecional no MQTT foi o *Decision Tree*, cuja média de consumo era menor que 10^4 uJ. Já no CoAP o algoritmo que obteve o menor consumo foi o *Naive Bayes* com um valor um pouco acima de 10^4 uJ.

Outrossim, observou-se que no componente de hardware *Package*, aplicando-se o fluxo unidirecional para o MQTT o algoritmo que obteve a menor média de consumo de energia foi o *Decision Tree*, obtendo um valor abaixo de $2x10^5$ uJ. Já para o CoAP o algoritmo que obteve a menor média de consumo foi o *Naive Bayes*, obtendo um valor de média abaixo de 10^5 uJ.

Logo, considerando o consumo energético dos componentes de hardware e os desempenhos dos algoritmos de ML conclui-se que o melhor algoritmo para classificar o tráfego entre ataques e legítimos com os dados sintéticos produzidos nesta pesquisa em um fluxo unidirecional é o *Decision Tree* no MQTT e no CoAP é o *Naive Bayes*.

Por fim, considerando o desempenho dos algoritmos de ML no fluxo bidirecional e os consumos de energia deles, o algoritmo *Decision Tree* foi o melhor avaliado para o MQTT e o *Naive Bayes* para o CoAP.

5.1 Trabalhos Futuros

A continuação desta pesquisa pode ser realizada por meio da utilização dos algoritmos de ML com variação dos hiperparâmetros para determinar um limiar de desempenho desses algoritmos em outros ambientes controlados.

Outrossim, outra contribuição seria a exploração em um cenário real com dispositivos

IoT e com ferramentas de ataque para gerar anomalias de rede. Dessa forma, é possível realizar testes conclusivos sobre os usos dos algoritmos.

Além disso, o avanço dos mecanismos de detecção poderiam ser expandidos para mitigação das anomalias de rede em uma rede de dispositivos IoT, com estudo aprofundado para embarcar esses mecanismos com equipamentos com pouca memória e processamento limitado. Portanto, esses equipamentos se tornam eficientes na detecção de anomalias.

5.2 Trabalhos completos publicados em anais de congresso

1. OLIVEIRA, Luciana P. ; Vieira, Moroni N. ; LEITE, Gabriel B. ; ALMEIDA, E. L. V. . Evaluating energy efficiency and security for Internet of Things: A Systematic Review. In: nternational Conference on Advanced Information Networking and Applications (AINA), 2020, Caserta. The 34th International Conference on Advanced Information Networking and Applications (AINA), 2020.

5.2.1 Capítulos de livros publicados

1. Oliveira, Luciana P. ; Vieira, Moroni N. ; Leite, Gabriel B. ; de Almeida, Edson L. V. . Evaluating Energy Efficiency and Security for Internet of Things: A Systematic Review. Advances in Intelligent Systems and Computing. 1ed.: Springer International Publishing, 2020, v. , p. 217-228.

REFERÊNCIAS BIBLIOGRÁFICAS

ABIODUN, O. I. et al. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, Elsevier, v. 4, n. 11, p. e00938, 2018. Citado na página 28.

AG, P. *What is MQTT? Definition and Details*. 2021. <https://www.paessler.com/it-explained/mqtt>. Acesso em: 10 Jul 2021). Citado na página 37.

ALJAWARNEH, S.; ALDWAIRI, M.; YASSEIN, M. B. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *Journal of Computational Science*, Elsevier, v. 25, p. 152–160, 2018. Citado na página 21.

ALLAHYARI, M. et al. A brief survey of text mining: Classification, clustering and extraction techniques. *arXiv preprint arXiv:1707.02919*, 2017. Citado na página 21.

ALTMAN, N. S. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, Taylor Francis, v. 46, n. 3, p. 175–185, 1992. Disponível em: <https://amstat.tandfonline.com/doi/abs/10.1080/00031305.1992.10475879>. Citado na página 28.

AMARAL, J. P. et al. Policy and network-based intrusion detection system for ipv6-enabled wireless sensor networks. In: IEEE. *2014 IEEE international conference on communications (ICC)*. [S.l.], 2014. p. 1796–1801. Citado na página 24.

ANDY, S.; RAHARDJO, B.; HANINDHITO, B. Attack scenarios and security analysis of mqtt communication protocol in iot system. In: IEEE. *2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*. [S.l.], 2017. p. 1–6. Citado 2 vezes nas páginas 18 e 36.

AZZOLA, F. *CoAP Protocol: Step-by-Step Guide - DZone IoT*. DZone, 2019. Disponível em: <https://dzone.com/articles/coap-protocol-step-by-step-guide>. Acesso em: 26 Jan. 2021. Citado na página 39.

BANKS ANDREW; GUPTA, R. *MQTT Version 3.1.1*. 2015. <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.pdf>. Acesso em: 14 Out 2021. Citado na página 37.

BASSI, A. et al. *Enabling things to talk*. [S.l.]: Springer Nature, 2013. Citado na página 35.

BATISTA, G. E. d. A. P. et al. *Pré-processamento de dados em aprendizado de máquina supervisionado*. Tese (Doutorado) — Universidade de São Paulo, 2003. Citado 2 vezes nas páginas 60 e 62.

BISWAS, S. K. et al. Intrusion detection using machine learning: A comparison study. *International Journal of pure and applied mathematics*, v. 118, n. 19, p. 101–114, 2018. Citado na página 21.

BURSCHKA STEFAN; DUPASQUIER, B. *Tranalyzer Documentation*. 2020. Disponível em: <https://tranalyzer.com/documentation>. Acesso em: 21 Jun 2021. Citado 3 vezes nas páginas 47, 56 e 58.

CARNEIRO, B.; FEITOSA, E. L. Device fingerprinting: Conceitos e técnicas, exem-plos e contramedidas. *Minicursos do XIV Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais–SBSeg*, 2014. Citado na página 45.

CARO, N. D. et al. Comparison of two lightweight protocols for smartphone-based sensing. In: IEEE. *2013 IEEE 20th Symposium on Communications and Vehicular Technology in the Benelux (SCVT)*. [S.l.], 2013. p. 1–6. Citado na página 18.

CAVALCANTI, T. *Aula 08 - Scikit-Learn - máquina de vetores de suporte*. 2021. <https://www.codigofluente.com.br/aula-08-scikit-learn-maquina-de-vetores-de-suporte/>. Acesso em: (Accessado on 27 Jun 2021). Citado na página 32.

CERT.BR. *CERT.br Stats (janeiro a junho de 2020)*. 2020. <https://www.cert.br/stats/incidentes/2020-jan-jun/tipos-ataque.html>. Acesso em: (Accessado on 26 Jun 2021). Citado na página 19.

CERT.BR. *Estatísticas do CERT.br – Honeypots*. 2020. Disponível em: <https://www.cert.br/stats/honeypots/>. Acesso em: 14 Abr 2021. Citado na página 19.

CERVANTES, C. et al. Detection of sinkhole attacks for supporting secure routing on 6lowpan for internet of things. In: IEEE. *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. [S.l.], 2015. p. 606–611. Citado na página 24.

CIKLABAKKAL, E. et al. Artemis: an intrusion detection system for mqtt attacks in internet of things. In: IEEE. *2019 38th Symposium on Reliable Distributed Systems (SRDS)*. [S.l.], 2019. p. 369–3692. Citado na página 24.

COLMANT, M. et al. Wattskit: Software-defined power monitoring of distributed systems. In: IEEE. *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. [S.l.], 2017. p. 514–523. Citado na página 49.

COLMANT, M. et al. Process-level power estimation in vm-based systems. In: *Proceedings of the Tenth European Conference on Computer Systems*. [S.l.: s.n.], 2015. p. 1–14. Citado na página 49.

COMBS, G. *tshark - The Wireshark Network Analyzer 3.4.6*. 1998. <https://www.wireshark.org/docs/man-pages/tshark.html#:~:text=DESCRIPTION,the%20packets%20to%20a%20file.> Acesso em: 01 Jul 2021. Citado na página 47.

CONTI, M. *Secure wireless sensor networks*. [S.l.]: Springer, 2015. Citado na página 36.

CROSS-VALIDATION. 2020. Disponível em: <https://scikit-learn.org/stable/modules/cross_validation.html>. Acesso em: 09 Jun 2021. Citado 2 vezes nas páginas 33 e 34.

CRUZ, A. et al. Sistema integrado hardware-software-mobile-web para controle de acesso com estudo de caso no ifrn–campus nova cruz. Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, 2020. Citado na página 18.

DAMODARAN, P. *CoAP can put IoT to REST - Musings of a Reductionist*. 2015. <http://bytecontinnum.com/2015/08/rest-for-wot-with-coap/>. Acesso em: 10 Jul 2021. Citado na página 42.

DUNKELS, A.; GRONVALL, B.; VOIGT, T. Contiki-a lightweight and flexible operating system for tiny networked sensors. In: IEEE. *29th annual IEEE international conference on local computer networks*. [S.l.], 2004. p. 455–462. Citado na página 23.

EASTEP, J. et al. Global extensible open power manager: A vehicle for hpc community collaboration on co-designed energy management solutions. In: SPRINGER. *International Supercomputing Conference*. [S.l.], 2017. p. 394–412. Citado na página 49.

FARNAAZ, N.; JABBAR, M. Random forest modeling for network intrusion detection system. *Procedia Computer Science*, Elsevier, v. 89, p. 213–217, 2016. Citado na página 30.

FERREIRA, U. J. S. F. et al. Análise de tecnologias de virtualização e hardware de baixo custo para infraestutura de nuvem de pequeno porte. Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, 2017. Citado na página 18.

FETZER, J. H. What is artificial intelligence? In: *Artificial Intelligence: Its Scope and Limits*. [S.l.]: Springer, 1990. p. 3–27. Citado na página 20.

GROUP, S. research. *PowerAPI - Overview*. 2020. Disponível em: <https://powerapi-ng.github.io/powerapi_howitworks.html#usage>. Acesso em: 19 Mai 2021. Citado na página 49.

GURUNATH, R. et al. An overview: security issue in iot network. In: IEEE. *2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), 2018 2nd International Conference on*. [S.l.], 2018. p. 104–107. Citado na página 44.

HALDER, S.; GHOSAL, A.; CONTI, M. Efficient physical intrusion detection in internet of things: A node deployment approach. *Computer Networks*, Elsevier, v. 154, p. 28–46, 2019. Citado na página 36.

HATTORI, L. et al. On the precision and accuracy of impact analysis techniques. In: IEEE. *Seventh IEEE/ACIS International Conference on Computer and Information Science (icis 2008)*. [S.l.], 2008. p. 513–518. Citado na página 61.

HEBA, F. E. et al. Principle components analysis and support vector machine based intrusion detection system. In: IEEE. *2010 10th international conference on intelligent systems design and applications*. [S.l.], 2010. p. 363–367. Citado na página 32.

HEDDEGHEM, W. V. et al. Energy in ict-trends and research directions. In: IEEE. *2009 IEEE 3rd International Symposium on Advanced Networks and Telecommunication Systems (ANTS)*. [S.l.], 2009. p. 1–3. Citado na página 21.

HINDY, H. et al. *Abertay Machine Learning Group*. 2020. Disponível em: <https://github.com/AbertayMachineLearningGroup/MQTT_ML>. Acesso em: 18 jul 2020. Citado na página 54.

HINDY, H. et al. Machine learning based iot intrusion detection system: An mqtt case study. *arXiv preprint arXiv:2006.15340*, 2020. Citado 12 vezes nas páginas 24, 52, 53, 54, 55, 56, 58, 59, 60, 65, 68 e 122.

HORST, P. S. *Avaliaçao do conhecimento adquirido por algoritmos de aprendizado de máquina utilizando exemplos*. Tese (Doutorado) — Universidade de São Paulo, 1999. Citado na página 62.

HUSSAIN, L. et al. Detecting congestive heart failure by extracting multimodal features and employing machine learning techniques. *BioMed research international*, Hindawi, v. 2020, 2020. Citado na página 30.

INC., G. *QUIC, a multiplexed stream transport over UDP - The Chromium Projects*. 2021. <https://www.chromium.org/quic>. Acesso em: (Accessed on 06/16/2021). Citado na página 23.

INRIA, U. *pyJoules*. 2019. Disponível em: <https://pypi.org/project/pyJoules/>. Acesso em: 21 Jun 2021. Citado 3 vezes nas páginas 49, 50 e 80.

ISHIDA, M.; TAKAKURA, H.; OKABE, Y. High-performance intrusion detection using optigrid clustering and grid-based labelling. In: IEEE. *2011 IEEE/IPSJ International Symposium on Applications and the Internet*. [S.l.], 2011. p. 11–19. Citado na página 29.

ISI-TICS. *Principais sub divisões e aplicabilidade da aprendizagem de máquina*. 2018. Disponível em: <https://isitics.com/2018/05/10/principais-sub-divisoes-e-aplicabilidade-da-aprendizagem-de-maquina/>. Acesso em: 20 jul 2020. Citado 3 vezes nas páginas 26, 27 e 28.

IZEBOUDJEN, N.; LARBES, C.; FARAH, A. A new classification approach for neural networks hardware: from standards chips to embedded systems on chip. *Artificial Intelligence Review*, Springer, v. 41, n. 4, p. 491–534, 2014. Citado na página 26.

JOSHITTA, R. S. M.; AROCKIAM, L. Security in iot environment: a survey. *International Journal of Information Technology and Mechanical Engineering*, v. 2, n. 7, p. 1–8, 2016. Citado na página 45.

JUNIOR, G. Máquina de vetores suporte: estudo e análise de parâmetros para otimização de resultado. *Ciência da Computação, Universidade Federal de Pernambuco*, 2010. Citado 2 vezes nas páginas 32 e 33.

KOKILA, R.; SELVI, S. T.; GOVINDARAJAN, K. Ddos detection and analysis in sdn-based environment using support vector machine classifier. In: IEEE. *2014 Sixth International Conference on Advanced Computing (ICoAC)*. [S.l.], 2014. p. 205–210. Citado na página 31.

KRISHNA, C.; SAMPATH, N. Healthcare monitoring system based on iot. In: IEEE. *2017 2nd International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)*. [S.l.], 2017. p. 1–5. Citado 2 vezes nas páginas 17 e 35.

KUMAR, J. S.; PATEL, D. R. A survey on internet of things: Security and privacy issues. *International Journal of Computer Applications*, Foundation of Computer Science, v. 90, n. 11, 2014. Citado na página 45.

KUMAR, P.; DEZFOULI, B. Implementation and analysis of quic for mqtt. *Computer Networks*, Elsevier, v. 150, p. 28–45, 2019. Citado 2 vezes nas páginas 23 e 24.

KURKOVSKY, S.; WILLIAMS, C. Raspberry pi as a platform for the internet of things projects: Experiences and lessons. In: *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. [S.l.: s.n.], 2017. p. 64–69. Citado na página 17.

KUROSE, J.; ROSS, K. *Computer Networking: A Top Down Approach*. [S.l.]: Addison-Wesley, 2012. Citado 2 vezes nas páginas 40 e 45.

LASZLO, E. *Science and the Akashic field: An integral theory of everything*. [S.l.]: Simon and Schuster, 2007. Citado na página 45.

LATAH, M.; TOKER, L. Artificial intelligence enabled software-defined networking: a comprehensive overview. *IET Networks*, IET, v. 8, n. 2, p. 79–99, 2018. Citado 2 vezes nas páginas 26 e 27.

LUXBURG, U. V.; SCHÖLKOPF, B. Statistical learning theory: Models, concepts, and results. In: *Handbook of the History of Logic*. [S.l.]: Elsevier, 2011. v. 10, p. 651–706. Citado na página 28.

LYON, G. F. *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. [S.l.]: Insecure. Com LLC (US), 2008. Citado na página 46.

METONGNON, L.; SADRE, R. Beyond telnet: Prevalence of iot protocols in telescope and honeypot measurements. In: *Proceedings of the 2018 Workshop on Traffic Measurements for Cybersecurity*. [S.l.: s.n.], 2018. p. 21–26. Citado na página 19.

MIGLIARDI, M.; MERLO, A. Modeling the energy consumption of distributed ids: A step towards green security. In: IEEE. *2011 Proceedings of the 34th International Convention MIPRO*. [S.l.], 2011. p. 1452–1457. Citado na página 21.

MIORANDI, D. et al. Internet of things: Vision, applications and research challenges. *Ad hoc networks*, Elsevier, v. 10, n. 7, p. 1497–1516, 2012. Citado na página 44.

MUN, D.-H.; DINH, M. L.; KWON, Y.-W. An assessment of internet of things protocols for resource-constrained applications. In: IEEE. *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*. [S.l.], 2016. v. 1, p. 555–560. Citado na página 18.

NAIK, N. Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. In: IEEE. *2017 IEEE international systems engineering symposium (ISSE)*. [S.l.], 2017. p. 1–7. Citado na página 18.

NOUREDDINE, A.; ROUVOY, R.; SEINTURIER, L. A review of energy measurement approaches. *ACM SIGOPS Operating Systems Review*, ACM New York, NY, USA, v. 47, n. 3, p. 42–49, 2013. Citado na página 56.

OLIVEIRA, L. P.; SADOK, D. F. H. Pronet framework: Network management using semantics and collaboration. In: IEEE. *2013 5th International Conference on Intelligent Networking and Collaborative Systems*. [S.l.], 2013. p. 414–421. Citado na página 21.

OLIVEIRA, L. P. et al. Evaluating energy efficiency and security for internet of things: A systematic review. In: SPRINGER. *International Conference on Advanced Information Networking and Applications*. [S.l.], 2020. p. 217–228. Acesso em: 22 jul 2020. Citado 5 vezes nas páginas 24, 49, 51, 89 e 122.

ORACLE, V. *VirtualBox*. 2015. Citado na página 52.

PENG, C.-Y. J.; LEE, K. L.; INGERSOLL, G. M. An introduction to logistic regression analysis and reporting. *The journal of educational research*, Taylor & Francis, v. 96, n. 1, p. 3–14, 2002. Citado na página 28.

PERETTI, G.; LAKKUNDI, V.; ZORZI, M. Blinktoscoap: An end-to-end security framework for the internet of things. In: IEEE. *2015 7th International Conference on Communication Systems and Networks (COMSNETS)*. [S.l.], 2015. p. 1–6. Citado 2 vezes nas páginas 23 e 24.

PESCATORE, J.; SHPANTZER, G. Securing the internet of things survey. *SANS Institute*, p. 1–22, 2014. Citado na página 44.

PRADA, M. A. et al. Communication with resource-constrained devices through mqtt for control education. *IFAC-PapersOnLine*, Elsevier, v. 49, n. 6, p. 150–155, 2016. Citado na página 36.

PRIMARTHA, R.; TAMA, B. A. Anomaly detection using random forest: A performance revisited. In: IEEE. *2017 International conference on data and software engineering (ICoDSE)*. [S.l.], 2017. p. 1–6. Citado 2 vezes nas páginas 30 e 31.

QTCOM, N. H. *The Qt Company*. 2017. <https://www.qt.io/company>. Acesso em: 10 Jul 2021). Citado na página 43.

QUINA, A.; STAVLIOTIS, L. *SPARTA | Ferramentas de teste de penetração*. 2015. <https://tools.kali.org/information-gathering/sparta>. Acesso em: 29 Jun 2021. Citado na página 47.

RAHMAN, R. A.; SHAH, B. Security analysis of iot protocols: A focus in coap. In: IEEE. *2016 3rd MEC international conference on big data and smart city (ICBDSC)*. [S.l.], 2016. p. 1–7. Citado na página 17.

RAIKAR, M. M. et al. Data traffic classification in software defined networks (sdn) using supervised-learning. *Procedia Computer Science*, Elsevier, v. 171, p. 2750–2759, 2020. Citado na página 27.

RANDHAWA, R. H.; HAMEED, A.; MIAN, A. N. Energy efficient cross-layer approach for object security of coap for iot devices. *Ad Hoc Networks*, Elsevier, v. 92, p. 101761, 2019. Citado 3 vezes nas páginas 23, 24 e 122.

RAZA, S. et al. Lithe: Lightweight secure coap for the internet of things. *IEEE Sensors Journal*, IEEE, v. 13, n. 10, p. 3711–3720, 2013. Citado 2 vezes nas páginas 23 e 24.

RAZA, S.; WALLGREN, L.; VOIGT, T. Svelte: Real-time intrusion detection in the internet of things. *Ad hoc networks*, Elsevier, v. 11, n. 8, p. 2661–2674, 2013. Citado na página 24.

RAZAFIMANDIMBY, C.; LOSCRI, V.; VEGNI, A. M. A neural network and iot based scheme for performance assessment in internet of robotic things. In: IEEE. *2016 IEEE first international conference on internet-of-things design and implementation (IoTDI)*. [S.l.], 2016. p. 241–246. Citado na página 20.

SANCHES, M. K. *Aprendizado de máquina semi-supervisionado: proposta de um algoritmo para rotular exemplos a partir de poucos exemplos rotulados*. Tese (Doutorado) — Universidade de São Paulo, 2003. Citado na página 27.

SECURITY, H. N. *41.6 billion IoT devices will be generating 79.4 zettabytes of data in 2025*. 2019. Disponível em: <https://www.helpnetsecurity.com/2019/06/21/connected-iot-devices-forecast/>. Acesso em: 20 jul 2020. Citado na página 17.

SENGUPTA, J.; RUJ, S.; BIT, S. D. A comprehensive survey on attacks, security issues and blockchain solutions for iot and iiot. *Journal of Network and Computer Applications*, Elsevier, v. 149, p. 102481, 2020. Citado na página 17.

SERVICES, O. *SPARTA | Penetration Testing Tools*. 2021. <https://tools.kali.org/information-gathering/sparta>. Acesso em: 12 Jul 2021. Citado na página 55.

SHAIKH, S. A. et al. Network reconnaissance. *Network Security*, Elsevier, v. 2008, n. 11, p. 12–16, 2008. Citado na página 19.

SHELBY, Z.; HARTKE, K.; BORMANN, C. *The Constrained Application Protocol (CoAP)*. RFC Editor, 2014. RFC 7252. (Request for Comments, 7252). Disponível em: <https://rfc-editor.org/rfc/rfc7252.txt>. Citado 3 vezes nas páginas 40, 41 e 42.

SILVA, J. *Aplicação de Redes Definidas por Software em Internet das Coisas*. Tese (Doutorado), 04 2018. Citado na página 35.

SILVEIRA, A. C. d. M.; MELO, V. E. R. Resfar: um sistema de automação agrícola baseado em iot. Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, 2019. Citado na página 18.

TORRES, R. d. S.; SILVA, A. C. d. O. Sistema autônomo de logística industrial. Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, 2018. Citado na página 18.

TZOLOV, C. *tzolov/coap-shell*. 2018. Disponível em: <https://github.com/tzolov/coap-shell>. Acesso em: 26 Jan. 2021. Citado na página 43.

UFSCAR-LAPES. *StArt*. 2013. Disponível em: <http://lapes.dc.ufscar.br/tools/start_tool>. Citado na página 121.

WASILAK MACIEJ;AMSüSS, C. *The Python CoAP library*. 2014. Disponível em: <https://aiocoap.readthedocs.io/en/latest/#>. Acesso em: 26 Jan 2021. Citado na página 44.

WIRESHARK. 2005. Disponível em: <https://www.wireshark.org/docs/>. Citado 2 vezes nas páginas 53 e 58.

WONG, T.-T.; YANG, N.-Y. Dependency analysis of accuracy estimates in k-fold cross validation. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 29, n. 11, p. 2417–2427, 2017. Citado na página 34.

YANG, Y.; LI, J.; YANG, Y. The research of the fast svm classifier method. In: IEEE. *2015 12th international computer conference on wavelet active media technology and information processing (ICCWAMTIP)*. [S.l.], 2015. p. 121–124. Citado na página 32.

ZARPELÃO, B. B. et al. A survey of intrusion detection in internet of things. *Journal of Network and Computer Applications*, Elsevier, v. 84, p. 25–37, 2017. Citado na página 24.

ZHANG, B. et al. Network intrusion detection method based on pca and bayes algorithm. *Security and Communication Networks*, Hindawi, v. 2018, 2018. Citado na página 29.

Apêndices

APÊNDICE A – CÓDIGO-FONTE DE BUSCA DOS ARTIGOS NO GOOGLE

```
#!/usr/bin/python3

############################################################
#  Programa desenvolvido para realizar consultas\ #
#  automatizadas no Google para determinadas\     #
#  palavras-chave e bases arbitrias                #
#                                                  #
#  Desenvolvido por: Moroni Vieira                 #
#  Update 3.0 - 26 Mai 2020                        #
#                                                  #
#  Este script foi desenvolvido para localizar cdigos #
#  fonte relacionados a uma lista de autores, essa #
#  lista foi preparada anteriormente utilizando a #
#  busca por artigos e retirando os nomes completos dos#
#  autores.                                        #
#                                                  #
#  Biblioteca utilizada disponvel em               #
#  https://github.com/abenassi/Google-Search-API  #
#                                                  #
############################################################

from google import google
import time
import random

###Variavel que define a quantidade de pginas por resultado
num_page=1

###Lista com as bases de busca, pode inserir ou retirar mais fontes
lista1 = ['source code','github','gitlab','sourceforge']

###Funcao de Busca
def BuscaGoogle():

    for i in lista1:
        arqresult=open('arq/v3/'+str(i),'w')
```

```
37          file = open('lista_autores6','r')
38
39          for p1 in lista1:
40              LerLinhas = file.readlines()
41              CountTmp=0
42
43              for p2 in LerLinhas:
44
45                  p2 = p2.rstrip("\n")
46                  search_results = google.search("\"" + p1 + "\"" + " and " +
                        "\"" + p2 + "\"",num_page)
47                  print(p1 + " " + p2 )
48                  for result in search_results:
49                      print(result.description,";",result.link)
50                      arqresult.write(str(result.description) + ';' +
                            str(result.link) + '\n')
51
52                  arqresult.write('########################################'+p1+',
                        '+p2+'\n')
53
54                  ####Temporizador para evitar o problema 429 - Too Many Requests
55                  ####Foram realizados testes em escale de 10 minutos, a soluo
                        foi encontrada
56                  ####quando a metrica do tempo ficou em 20 minutos
57                  if (CountTmp < 10):
58                      time.sleep(random.randrange(1,60))
59                      CountTmp += 1
60                  else:
61                      time.sleep( 1200 )
62                      CountTmp = 0
63
64          arqresult.close()
65  #
66  ########################
67  ### Mdulo principal
68  #
69  def main():
70      BuscaGoogle()
71  #
72  if __name__ == '__main__':
73      main()
74  #
```

APÊNDICE B – SCRIPT DE CONSULTA AOS SENSORES IOT

```
#####################################################################
#   Script desenvolvido para realizar consulta aos sensores  #
#   CoAP, cada mensagem e IP dos sensores so escolhidos #
#   aleatoriamente durante todo a execuo         #
#                     #
#   Autor: Moroni Vieira            #
#    verso : 6                    #
#   26  Maro  de 2021               #
#                     #
#####################################################################

#!/bin/bash

while :
do

   RAND=$(shuf -i 1-4 -n 1)

   case $RAND in
      1 )

         RAND1=$(shuf -i 1-12 -n 1)
         IP=‘sed -n "$RAND1"’p’ lista_ips‘
         aiocoap-client -m get coap://$IP/time
         ;;

      2 )
         RAND1=$(shuf -i 1-12 -n 1)
         IP=‘sed -n "$RAND1"’p’ lista_ips‘
         aiocoap-client -m get coap://$IP/other/block
         ;;

      3 )
         RAND1=$(shuf -i 1-12 -n 1)
         IP=‘sed -n "$RAND1"’p’ lista_ips‘
         aiocoap-client -m get coap://$IP/other/separate
```

```
37          ;;
38
39
40      4 )
41          RAND1=$(shuf -i 1-12 -n 1)
42          IP=‘sed -n "$RAND1"’p’ lista_ips‘
43          aiocoap-client -m get coap://$IP/whoami
44          ;;
45
46
47      * )
48          ;;
49
50
51  esac
52
53
54  done
```

APÊNDICE C – ARTIGO AINA 2020

Evaluating energy efficiency and security for Internet of Things: A Systematic Review

Luciana P. Oliveira; Moroni N. Vieira; Gabriel B. Leite; Edson L. V. de Almeida

Abstract The Internet of Things (IoT) contains devices to collect and transmit information that should not be captured by third parties. Although devices have internet connectivity, some devices have battery limited energy. Hence, it is important the conducting of evaluations to investigate, compare and propose security mechanisms with energy-efficient. However, there is a very low number of research using practical evaluations with real devices, as a measurement approach takes a long time to perform the experiments. Therefore, the aim of this paper is to analyze, categorize analytically and statistically the methodologies which have been presented in the area of IoT based on the Systematic Literature Review (SLR) method, in order to extract features of the research protocols (definition of a set of equipment, topology, traffic, parameters, metrics and other variables) to be used in new research that addresses the concepts of security, energy and IoT. The weaknesses and highlighting of studies is discussed as well as presenting some hints for addressing future research with lower execution times of experiments, as it will be possible to reuse the results of measurements already published for the same parameter and metrics.

1 Introduction

The Internet of Things (IoT) is constituted of devices that have similar characteristics to WSN (Wireless Sensor Networks) sensors, having limited energy to collect and transmit information that should not be captured by third parties. Therefore, works such as [1] consider IoT as the result of WSN with Internet access, being pos-

Luciana P. Oliveira; Gabriel B. Leite; Edson L. V. de Almeida
IFPB Campus João Pessoa, Av. Primeiro de Maio, 720 - Jaguaribe, João Pessoa - PB, e-mail: luciana.oliveira@ifpb.edu.br,gabriel.bezerra@academico.ifpb.edu.br,edson.l.v.almeida@outlook.com

Moroni N. Vieira
IFRN Campus Currais Novos, Rua Manoel Lopes Filho, 773 - Valfredo Galvão, Currais Novos - RN e-mail: moroni.neres@ifrn.edu.br

sible to remotely access several information from the environment in which wireless sensors were deployed [3],[4]. In addition, changing batteries in WSN and some IoT solutions should be avoided, so it is important to use security mechanisms with energy efficiency. This is because in some cases the devices are deployed in a hostile environment where there is no possibility of replacing or recharging the batteries. Therefore, the choice of low-energy security mechanisms is critical, and for this reason this research found more than 800 papers that evaluated energy consumption of WSN and IoT security solutions through simulations, analytical modeling or practical experiments by measurement.

This work found 362 secondary technical papers (surveys and reviews) and only 49 contained information about the IoT, energy and security. However, none of the secondary work have been focused to comprehend practical researches with real devices. Although it is possible to find 927 papers that executed evaluations, only 72 works described the experiments through practical analysis, because measurement approach takes a long time to perform the experiments and it is hard to reuse results from existing works, since there is not an addressing for a common research protocol to analyze the combining of the energy and security in IoT.

Therefore, the main aim of this research based on Systematic Literature Review (SLR) method is to comprehend the diversity of research protocols to investigate or compare the methodologies to evaluate security mechanisms with low energy consumption for IoT (including WSN). The main commitments of this study are highlighted as follows:

- Presenting a discussion of the research protocols (definition of a set of equipment, topology, traffic, parameters, metrics and other variables) that considered energy, security and sensors (IoT or WSN) in measurement approach.
- Presenting the weaknesses and highlighting among evaluations of the security mechanisms based on practical experiments.
- Designing recommendations to be addressing future research with energy, iot, security and measurement approach.

The organization of this systematic review is considered as follows. Section 2 presents the related works. The research methodology (SRL process) is clarified in Section 3. The discussion of the papers selected by SRL is described in section 4, including the weaknesses, highlighting and recommendations to be addressing future research with energy, iot, security and measurement approach. Finally, the conclusion and future works are in Section 5.

2 Related Works

This section presents a summary of some characteristics of the secondary studies that described information about IoT, security and energy issues.

The paper [4] is a review about Edge Computing in terms of the definition, characteristics and architectures, considering IoT and security (no deep), but it does not include energy consumption.

In the paper [5], a survey about sensors to monitor healthcare is presented, considering security, energy consumption and others contexts. However, this study does not analyze the measurements that associate energy consumption and security.

The work [6] is a survey about Smart Grid (SG). This explains how solution for minimizing the wastage of electrical energy can be used with IoT and other information about SG in terms of applications, architectures, prototypes, big data and communication IoT and non-IoT for smart grid. However, this study does not analyze existed experiments with IoT devices considering security.

In terms of [7], this work describe about data aggregation to avoid the transmission of the duplicate data. It compared the protocol considering intelligent technique used, energy consumption, cost reduction, security, accuracy, organisational type and metrics. However the analyzed metrics, security and energy were not studied in terms of existed real evaluations.

Similarly to paper [6], the paper [8] addresses energy in the context of smart power. It reviews the development and implementation of smart power meters, including smart electricity meters, smart heat meters, and smart gas meters. However, this study does not analyze existed experiments with IoT devices considering security.

An in-depth research and analyzes of the security issues were presented by [9], [10], [11]. The paper [9] merger security and smart city, including a view of the taxonomia for security, while [10], [11] associate WSN and security. However, both works did not analyze the studies in terms of parameters, metrics and other features that are presents in evaluations.

Therefore, this paper is the first study, using SLR method, that reviews the evaluation methods of security and energy in the IoT (including WSN). Briefly, table 1 represents a summary of the related secondary studies (the systematic literature review, survey or review - no systematic). It is clear that none of the papers have used the SLR method for study IoT, energy and security. In table 1, the paper with E+ contains study components (some comparative table or classification) of the energy, and S+ represents works that contains study components of the security. Therefore just one paper ([7]) contains some explicit information about security and energy, but the information in terms of measurement is superficially described (only one table that does not present parameter, metric and others important information about evaluation). According to the existing secondary papers, the existing deficiencies propose that is important to do a comprehensive literature review to address these weaknesses as follows:

- The present studies do not provide in-deth study about measurements in IoT.
- Some studies do not evaluate the important association among security, energy and IoT devices.
- The structure of the presented studies does not have the systematic review and the paper selection method is not clear.

Table 1 Secondary works in the combining of IoT, security and energy

Ref.	Review P. type	Year	Paper selection process	Measurement	Main topic	Security(S)Energy(E)	Covered years
[4]	Review	2019	Not Clear	No	Edge Computing	S-/E-	2015-2018
[5]	Survey	2010	Not Clear	No	Healthcare	S-/E-	2003-2009
[6]	Survey	2019	Not Clear	No	Smart Grid	S+/E-	2011-2015
[7]	Survey	2013	Not Clear	Partially	Data Aggregation	S+/E+	2006-2011
[8]	Review	2016	Not Clear	No	Smart Energy	S-/E+	2007-2015
[9]	Survey	2019	Not Clear	No	Smart City	S+/E-	2009-2018
[10]	Survey	2017	Not Clear	No	WSN	S-/E-	2007-2016
[11]	Survey	2018	Not Clear	No	WSN	S+/E-	2002-2017

3 Methodology

A systematic review according to [12] is a type of study conducted using a methodology that analyzes published articles as consequence the experiment to select works can be reproduced. This methodology is used to select the most relevant studies to, for example, identify similarities or contradictions between papers selected. This method can be conducted by manually process or by tools. For example, Start [13] is an example of a tool that can be used to produce a systematic review. This tool allows the importation of file with extension ".bib" that were exported from the electronic bases, allowing the management of a large number of articles to be classified in the three stages of the systematic review methodology: data planning; execution; and analysis. The first phase includes the formulation of research questions, definition of inclusion and exclusion criteria. The second phase is used to select relevant papers and to extract data that will analyzed in last phase. The third phase is executed to generate results of the graphics, tables and descriptions.

3.1 Research Questions (RQs)

In order to understand the evaluations of studies that merger energy, security and IoT, this SLR paper formulated eight question that will answers in last phase of the research:

1. How are the distribution of the research studies over the years?
2. Which domains are categorized the selected papers?
3. Do the experiments have a high level of reproducibility and precision?
4. What are the characteristics of the parameters (equipment, traffic types and others) most used to configure the measurement environment?

5. What are the most commonly used metrics?
6. What recommendations for future work that will conduct experiments in the field of IoT, energy and security?

3.2 Search process

The objective of an SLR is to conduct a review of relevant studies in order to quantity of evidence existing to address the RQs. This method is rigorous and unbiased, allowing to be reproduce by other people. For this reason, it is important to define which are electronic bases were used to search the papers and to inform the inclusion and exclusion criteria to select the papers.

This study was based on the following electronic databases:IEEE Xplore, ACM digital library and Science Direct. Each electronic resources accepts different string format and, for this reason, the Fig. 1 shows the search string defined to retrieve information for each electronic resources.

ACM	(+"Internet of Things" +"energy" +"consumption" +"Security") (+"IoT" +"energy" +"consumption" +"Security") (+"Sensor Network" +"energy" +"consumption" +"Security") (+"WSN" +"energy" +"consumption" +"Security")
IEEE	((((("Internet of Things") AND energy) AND consumption) AND security) ((((("IoT") AND energy) AND consumption) AND security) ((((("Sensor Network") AND energy) AND consumption) AND security) ((((("WSN") AND energy) AND consumption) AND security)
Science Direct	"Internet of Things" AND "energy" AND "consumption" AND "Security" "IoT" AND "energy" AND "consumption" AND "Security" "Sensor Network" AND "energy" AND "consumption" AND "security" "WSN" AND "energy" AND "consumption" AND "security"

Fig. 1 Search Strings

After the definition of the search string for each electronic base, this SLR identified 5671 papers. Therefore, the following inclusion (IC) and exclusion (EC) criteria were important to identified a viable number of papers to analyze:

- EC1: Works not written in English.
- EC2: Papers does not contain in the title or in the abstract the words in the context of security (security or secure or malicious or crypt or shared key or confidential or attack) and energy (energy or power consumption).
- EC3: Secondary studies (abstract or title contains the word review or survey).
- EC4: Duplicate articles.
- EC5: Secondary work identified after to read it, because relevant words (review or survey) were not in title or abstract.
- EC6: Incomplete work, because the paper has no value related to measurement experiments.
- EC7: Papers with irrelevant content, because after reading, they did not present measurement description in terms of the energy consumption and security.
- IC1: papers that were not excluded by EC1, EC2, EC3 and EC4
- IC2: papers did not exclude by any of the exclusion criteria.

3.3 Studies Selection

In this SRL, the paper was selected in execution phase that was subdivided into two stages with support of the Start Tool, as follows:

- Stage 1: Analyze title and summary of all articles to confront them with the exclusion criteria (EC1, EC2, EC3 and EC4). The selected paper in this stage were classified by inclusion criteria IC1.
- Stage 2: Read all paper in order to exclude works by CE5, EC6 and EC7. The selected paper in this stage were classified by IC1 and IC2.

As a result, in stage 1, a total of 927 studies were selected and 4744 were excluded. In stage 2, the full text of each primary study included as IC1 was read to include as IC2 and to extract information related to RQs. The primary studies included in the final selection corresponded to the relevant papers that meet the RQs addressed by this SRL. It was excluded 855 papers and 72 relevant studies were identified from the 3 electronic bases.

4 Results

This section presents and discusses the answers to six RQs in section 3.1 considering the results obtained in the analysis of the 72 papers that were included in the study. Moreover, more details about data results from this research can be find in https://sourceforge.net/projects/aina2020/files.

4.1 RQ1: How are the distribution of the research studies of the measurement in IoT over the years?

The studies found in this review are in a time period between 2006 and 2019. However, this studies extracted paper until on June 2019, for this reason the Fig. 2 a) shows the number of included papers by year of publication and it does show 2019. Thirty papers published before 2015 were included, since the inclusion and exclusion criteria placed no bar upon the year of publication. From 2016, it was possible to identify and include at least six paper with explicit IoT for each year. This indicates that the area has become more mature from 2016.

4.2 RQ2: Which domains are categorized the selected papers?

The papers were categorized in three main types of domains in order to study IoT: information security, energy measurement and research feature. The information

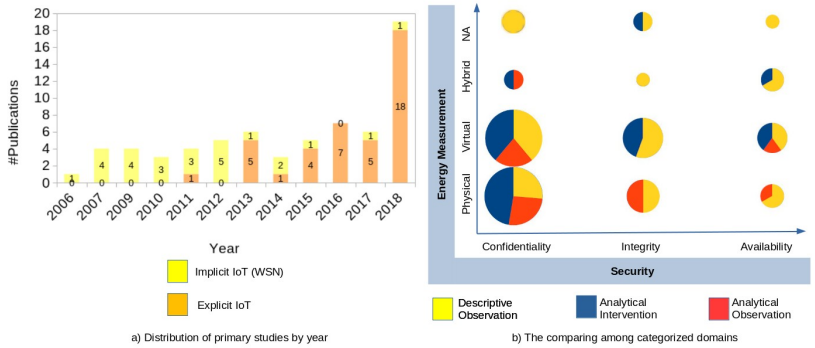


Fig. 2 Distribution of results

security was subdivided in Confidentiality (paper contains information of the mechanisms to avoid the extraction or monitoring of the information for unauthorised users), Integrity (works has studies about technical to avoid the modification or changing of the information and system setting for instance unauthorized access to sensitive information) and Availability (articles presents information to avoid the system close and unavailable for authorized users). The energy is the classification in terms of mechanism used to extract the energy consumption of security approach for IoT. This domain was splitted in Virtual (the measurement of the energy is based on some software or mathematical model), Physical (some real tool was used to identify the energy consumption), Hybrid (the energy is identified using the approach based on Virtual and Physical domains) and NA (the paper did not describe how to reach the energy results).

The research type was separated in three types of experimental studies: analytical intervention, analytical observation and descriptive observation. The analytical intervention are works that contain a proposal analyzed against other existing proposals in order to prove hypotheses regarding this new proposal. Analytical observation are studies that perform the analysis of experiments to compare two or more existing solutions to prove hypotheses. Descriptive observation are articles that investigate events through experiments that will generate hypotheses or when a proposal was not analyzed against other existing work.

The results in Fig. 2 b) show that no experiments with physical measurement were found in order to describe analytical intervantion in terms of availability and integrity. Moreover, there are few experiments considering availability. On the other hand, there is a large number of research related to analytical intervention in the confidentiality subarea of the information security. In terms of energy measurement, most articles are classified at the virtual level. That is, few studies actually measured the energy consumption with real equipment (oscilloscope, voltage meter, etc.).

4.3 RQ3: Do the experiments have a high level of reproducibility and precision?

Reproducibility is the combination observed when different operators measure the same part multiple times using the same meter (hardware) under the same conditions (environment configuration).

So, the experimental research should describe the environment in high detail of the hardware and parameters used in the configurations in order to ensure a high level of reproducibility (the same evaluation results or results with very little degree of difference). Moreover, it is important, because the measurement with real devices is very costly and, therefore, evaluating new solutions should avoid the cost of remeasuring new studies already evaluated.

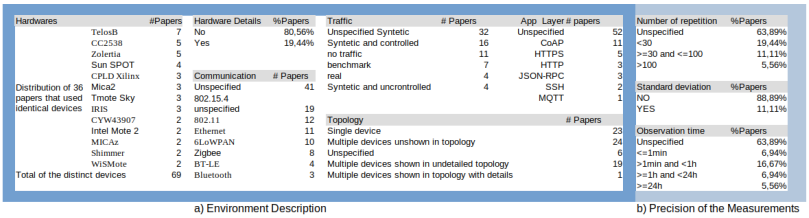


Fig. 3 Comparing the selected papers (enviroment and precision)

However, the Fig. 3 (a) shows that only 19.44% of the selected papers in this SRL provide details on specifying all hardware used (processor, memory and others information can be found in datasheet).

In addition, there are a large number of papers that did not specify important items in IoT solutions. 41 papers did not inform the communication technology used and 19 did not report the type of 802.15.4 technology used. This technology can be used with different types of routing that influences energy consumption, so it is extremely important to detail the information.

Although it is possible to build an IoT solution without using an application layer protocol standard, 52 articles have not been explicit. There is also a lot of work that does not specify the topology and traffic used. Moreover, 69 distincts devices (including different versions, for example, Raspberry pi zero is different from Raspberry pi 3) were used by 72 selected papers.

An example of the importance of richness of detail is the description of traffic in the experiment. When studying the delay of communication without bottlenecks is not important, controlled synthetic traffic should be used. The uncontrolled synthetic traffic allows the use of sending many messages in order to stress the equipment, for example, it can be used to calculate throughput. When using a brenchmark, it provides a high degree of data reliability because it will be synthetic traffic that can be more easily reproduced. Real traffic is important for an observation survey to

make assumptions about, but it is not appropriate to evaluate a new proposal, as it will probably not be possible to reproduce this fact from the past.

The precision refers to the closeness between repeated measurements on the same experimental design. It is related to the number of repetitions and standard deviation. For example, [14] informs that "standard" values for the number of repeated runs is usually 30 or 50, with occasional recommendations of using 100 or more, because the higher the number of measurement repetitions has as result the lower the standard deviation and the greater the accuracy of the mean value of the obtained results.

However, the Fig. 3 (b) shows low precision in existing experimental researches in security for IoT. Several papers unspecified standard deviation, time observation and number of repetition. Moreover, small number of article realized experimental with adequate values to obtain precision results.

4.4 RQ4: What are the characteristics of the parameters most used to configure the measurement environment?

Controlled variables are also known as adjusted parameters before making an observation in the experiment. This SRL found 72 selected papers and identified two main parameters (controlled variables) in Fig. 4 to evaluate low-power security for the IoT: workload and security key size.

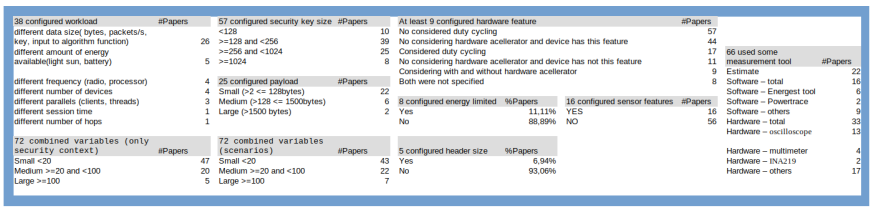


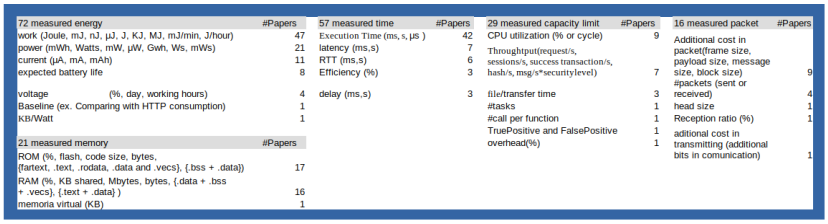
Fig. 4 Variables to configure the experimental research

In terms of workload, 26 papers configured the environment with different data sizes. In terms of the size of the security key, most of the 57 papers set this variable in the range of 128 and 256 bits. In overview, 56 selected papers did not consider sensor features (a kind of hardware in IoT) to configure the experimental. The number of scenarios was defined by counting the combinations of variables configured in the environment. For example, an article used 2 key sizes (AES 128 and AES 192) and 4 buffer sizes (16, 128, 512, 2048 bytes). In this example, 8 scenarios were counted. So, when comparing the count of general scenarios and, specifically, in terms of security, there was an increase in the number of small scenarios (32 for geral combinations - with and without security features - and 46 when analyzed with combinations containing security). In terms of hardware feature, 9 papers configured

the environment considering the hardware accelerator active and disabled. However, 44 papers did not consider this variable, although the device used in experiments had this feature.

4.5 RQ5: What are the most commonly used metrics?

Metrics are variables dependent on the parameters that were set to perform the experiment. This SRL identified two main metrics in Fig. 5: energy as metric analized by 72 papers and time is another metric referencied by 57 papers.



72 measured energy	#Papers
work (Joule, mJ, nJ, µJ, J, KJ, MJ, mJ/min, J/hour)	47
power (mWh, Watts, mW, µW, Gwh, Ws, mWs)	21
current (µA, mA, mAh)	11
expected battery life	8
voltage (%, day, working hours)	4
Baseline (ex. Comparing with HTTP consumption)	1
KB/Watt	1

57 measured time	#Papers
Execution Time (ms, s, µs)	42
latency (ms,s)	7
RTT (ms,s)	6
Efficiency (%)	3
delay (ms,s)	3

29 measured capacity limit	#Papers
CPU utilization (% or cycle)	9
Throughtput(request/s, sessions/s, success transaction/s, hash/s, msg/s*securitylevel)	7
file/transfer time	3
#tasks	1
#call per function	1
TruePositive and FalsePositive	1
overhead(%)	1

16 measured packet	#Papers
Additional cost in packet(frame size, payload size, message size, block size)	9
#packets (sent or received)	4
head size	1
Reception ratio (%)	1
aditional cost in transmitting (additional bits in comunication)	1

21 measured memory	#Papers
ROM (%, flash, code size, bytes, {fartext, .text, .rodata, .data and .vecs}, {.bss + .data})	17
RAM (%, KB shared, Mbytes, bytes, {.data + .bss + .vecs}, {.text + .data})	16
memoria virtual (KB)	1

Fig. 5 Metrics to measure the experimental research

Selected articles were related to low-power security for the IoT. This explains the 100% of the articles with the energy as the metric. However, Fig. 5 presents different units of energy captured by different tools presented in Fig. 4.

So, future work on security, energy and IoT should at least measure time execution and energy in terms of the work and power consumption. New papers with these metrics can be more easily compared to existing results from older works.

4.6 RQ6: What recommendations for future work that will conduct experiments in the field of IoT, energy and security?

This section presents a summary about recommendations for new experimental research in context of the security, energy and IoT that are enumerated in following:

- In terms of gaps, future researches must execute experiments that analyze availability and integrity, because there are lower number of the papers in these approach. For example, in the context of integrity, there are no analytical observation studies using some software to measure energy.
- In view of reproducibility and precision of the results, regardless of the security area in IoT, the description of new studies must reference a datasheet of each hardware used in research, to inform application layer, communication technology and traffic used in research to allow for high level of the reproducibility, as

this will provide hardware details for future reproductive of the research. Moreover, it is very important to present the figure with information of the location of the hardware in experimental environment, to describe the standard deviation, to have at least 100 as the number of repetition and a interval between 1 minuto and 1 hour as observation time.

- Considering configurable parameters in the evaluations, new papers should to use at least workload (ex. varying data size) and security key size between 128 and 1024 bits. However, the articles could be comparable with major number of paper if new experimental researches consider also the following parameters: payload with small size, duty cycling and hardware accelerator. In terms of the energy measurement, the authors should use at least one estimate model and oscilloscope for energy, because, when they present both results (estimating and real values), the article will provide information in a way that will be more easily compared with other articles.
- Taking account the metric types, new papers must use energy (work and power consumption) and time (execution) as main metrics. Additional the CPU utilization, additional cost in packet and memory ROM used should be used if the authors would like that their works can be compared with more existing papers. Other option is to amplify the capacity of the paper to be compared with other publication is when the article to explore the variation of metrics units for energy described in Fig. 5.

5 Conclusion

The presented results of this SLR aimed at investigating the characteristics of measurements in context of IoT, energy and security. This review has helped to understand the current state-of-the-art in methodologies to real experiments, and also to identify research gaps and future directions. After analyzing each measurement, they are classified according to their aims and the way they work. Thus, this SLR proposed three classification schemes: (i) by energy measurement type, (ii) by security, and (iii) by research type. Also, it has identified the main characteristics of the measurements (reproducibility, precision of the results, parameter used to configure the experiments and metrics).

The results of this SLR suggest that there is a predominance of real experiments which use software or some estimative to inform energy consumption of security approaches for IoT and a high number of distinct devices used in real measurement of energy consumption. This SLR encourages researches to continually execute real experiment in new IoT devices with at least one of the most used devices, because experiments with real devices contain important information to be used by energy consumption model and results of new devices should be compared with traditional device like TelosB that was referenced by 7 papers. Moreover, it was possible to find gaps in availability and integrity researches for IoT (subareas from information security).

In this direction, the results achieved in this study can be used as a guide for researches to identify which parameters and metrics best fit for new articles can be more broadly compared to existing publications. In addition, the problems of reproducibility and precision in selected papers from this SRL were identified.

References

1. M. T. Lazarescu, Wireless Sensor Networks for the Internet of Things: Barriers and Synergies,Springer International Publishing, Cham, 2017, pp. 155–186 (2017).
2. M. Elshrkawey, S. M. Elsherif, M. E. Wahed, An enhancement approach for reducing the energy consumption in wireless sensor networks, Journal of King Saud University - Computer and Information Sciences 30 (2) (2018) 259 – 267 (2018). doi:10.1016/j.jksuci.2017.04.002.
3. A. H. S. Menezes, K. R. M. de O., L. P. Oliveira, P. J. d. S. Oliveira, Iot environment to train service dogs, in: 2017 IEEE First Summer School on Smart Cities (S3C), 2017, pp. 137–140 (Aug 2017). doi:10.1109/S3C.2017.8501386.
4. I. Sittón-Candanedo, R. S. Alonso, J. M. Corchado, S. Rodríguez-González, R. Casado-Vara, A review of edge computing reference architectures and a new global edge proposal, Future Generation Computer Systems 99 (2019) 278 – 294 (2019). doi:10.1016/j.future.2019.04.016.
5. H. Alemdar, C. Ersoy, Wireless sensor networks for healthcare: A survey, Computer Networks 54 (15) (2010) 2688 – 2710 (2010). doi:10.1016/j.comnet.2010.05.003.
6. Y. Saleem, N. Crespi, M. H. Rehmani, R. Copeland, Internet of things-aided smart grid: Technologies, architectures, applications, prototypes, and future research directions, IEEE Access 7 (2019) 62962–63003 (2019). doi:10.1109/ACCESS.2019.2913984.
7. H. R. Dhasian, P. Balasubramanian, Survey of data aggregation techniques using soft computing in wireless sensor networks, IET Information Security 7 (4) (2013) 336–342 (December 2013). doi:10.1049/iet-ifs.2012.0292.
8. Q. Sun, H. Li, Z. Ma, C. Wang, J. Campillo, Q. Zhang, F. Wallin, J. Guo, A comprehensive review of smart energy meters in intelligent energy networks, IEEE Internet of Things Journal 3 (4) (2016) 464–479 (Aug 2016). doi:10.1109/JIOT.2015.2512325.
9. M. Sookhak, H. Tang, Y. He, F. R. Yu, Security and privacy of smart cities: A survey, research issues and challenges, IEEE Communications Surveys Tutorials 21 (2) (2019) 1718–1743 (Secondquarter 2019). doi:10.1109/COMST.2018.2867288.
10. H. C. Kantharaju, K. N. N. Murthy, A survey on enhancing system performance of wireless sensor network by secure assemblage based data delivery, in: 2017 International Conference on Recent Advances in Electronics and Communication Technology (ICRAECT), 2017, pp. 289–296 (March 2017). doi:10.1109/ICRAECT.2017.55.
11. A. Karakaya, S. Akleylek, A survey on security threats and authentication approaches in wireless sensor networks, in: 2018 6th International Symposium on Digital Forensic and Security (ISDFS), 2018, pp. 1–4 (March 2018). doi:10.1109/ISDFS.2018.8355381.
12. V. Ravindran, S. Subramanian, Systematic reviews and meta-analysis demystified, Indian Journal of Rheumatology 10 (05 2015). doi:10.1016/j.injr.2015.04.003.
13. S. Fabbri, C. Silva, E. Hernandes, F. Octaviano, A. Di Thommazo, A. Belgamo, Improvements in the start tool to better support the systematic review process, in: Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering, EASE '16, ACM, New York, NY, USA, 2016, pp. 21:1–21:5 (2016). URL http://doi.acm.org/10.1145/2915970.2916013
14. F. Campelo, F. Takahashi, Sample size estimation for power and accuracy in the experimental comparison of algorithms, Journal of Heuristics 25 (2) (2019) 305–338 (Apr 2019). URL https://doi.org/10.1007/s10732-018-9396-7

APÊNDICE D – RESUMO DA RSL

METODOLOGIA DA RSL

Desde que o assunto IoT surgiu, muitos desafios também evoluíram, com isso, a pesquisa acadêmica sobre esse tema é muito explorada atualmente, muitos estudos nesse contexto de redes IoT e segurança estão sendo o foco principal, inclusive eventos na área de pesquisa são específicos desses temas.

PLANEJAMENTO DA RSL

Primeiro foi definido qual o objetivo da RSL, o porquê da necessidade de estudar o tema, com isso, definiu-se a motivação para execução da pesquisa. Porém, antes de iniciar o planejamento foi identificado que existem estudos secundários que relacionavam-se com o tema em epígrafe, essa foi o ponto de partida para uma revisão terciária, onde foram observados as abordagens sobre os temas, se os estudo eram revisão sistemática ou somente um pesquisa sobre o contexto de IA e IoT, qual a descrição principal e o ano de cobertura. Dessa forma, concluiu-se que os temas são estudados de forma relacionada por tempo suficiente para seu amadurecimento, com isso, uma taxonomia poderia ser elaborada para organizar esses trabalhos por ano, forma de publicação, temas e técnicas abordadas.

Segundo Oliveira et al. (2020) há baixo número de pesquisas práticas com dispositivos reais, principalmente na área de disponibilidade da segurança da informação em redes IoT, com isso, surgiu a indagação sobre o estado da arte de novas abordagens de segurança à IoT que foram propostas para melhorar com SDN e IA. Diante disso, prosseguiu-se a RSL para a fase de elaboração do protocolo, nessa fase objetivou-se orientar o processo de pesquisa, as bases da pesquisa, o escopo de trabalho, os objetivos da pesquisa, questões da pesquisa, palavras chave que formam as expressões usadas na busca manual nas bases e o processo de seleção dos trabalhos encontrados.

O protocolo de RSL também auxilia na seleção, avaliação e classificação e na análise dos trabalhos selecionados. As questões de pesquisa definidas por essa revisão são as seguintes: (I) A distribuição por ano, eventos, periódicos, autores, abordagens de IA sobre o contexto de IoT e IA; (II) Quais algoritmos de IA propostos para resolver os tipos de ataques a IoT; (III) Quais os parâmetros (equipamentos, tipos de tráfego e outros) e métricas que foram usadas para avaliar os experimentos em trabalhos sobre o tema; (IV) Quais as recomendações para trabalhos futuros que conduziram os experimentos no contexto de IoT, SDN, IA e segurança para smart objects. Nessa fase, além de definir o protocolo da RSL, foi definida também a estratégia de busca por meio das strings de buscas, a partir disso, a busca foi realizada em três repositórios de pesquisas acadêmicas: IEEE Xplore, o Science Direct e o ACM Library. Dessa forma, foi desenvolvida uma estratégia principal, "(IoT OR Intelligent OR Learning OR Decision OR Adaptive OR SDN)", esta foi adaptada para que os repositórios aceitem as strings para buscar os trabalhos, a tabela 9 a seguir mostra as strings de busca para cada repositório.

Após a conclusão dessa etapa de busca, outro processo foi iniciado e realização desse

Tabela 9 – Tabela de Strings de Busca

BASE	STRING
ACM Library	(Title::((“IoT”“Intelligent” OR “Learning” OR “Decision” OR “Adaptative” OR “Optimization” OR “Inspired”) AND (“Software Defined Networking” OR “SDN”)))
IEEE Xplore	(“IoT” OR “Intelligent” OR “Learning” OR “Decision” OR “Adaptative” OR “Optimization” OR “Inspired”) AND (“Software Defined Networking” OR “SDN”)
Science Direct	(“IoT” “Intelligent” OR “Learning” OR “Decision” OR “Adaptative” OR “Optimization” OR “Inspired”) AND (“Software Defined Networking” OR “SDN”)

processo foi auxiliado computacionalmente pela ferramenta Start (UFSCAR-LAPES, 2013), essa foi utilizada para depositar os arquivos do tipo RIS e BIB, a partir disso foram extraídas informações dos trabalhos, por exemplo, título, abstract, autores e conclusão. Além disso, por meio da ferramenta foi possível gerar um banco de dados relacional que foi lido por uma linguagem DML (Data Definition Language), o programa SQLite foi utilizado para realizar consultas SQL (Standard Query Language).

EXECUÇÃO DA RSL A principal etapa foi a execução da RSL, durante essa fase foram encontrados o número de 21629 artigos no Science Direct, 5194 no IEEE Xplore e 1504 na ACM Library, esses estudos são classificados como primários e foram encontrados por meio de busca automática. Assim, foram encontrados 28327, as informações foram gerenciadas pelo software Start, o qual foi possível identificar trabalhos duplicados e analisar os artigos por meio de interface gráfica.

Outrossim, a automatização do processo permitiu a filtragem dos trabalhos, reduziu-se o conjunto de trabalhos de 28327 para 84 e esta seleção considerou um intervalo de anos de 2010 a 2020 e foi guiado pelos critério de inclusão (IC) e exclusão (EC), a tabela a seguir mostra esses critérios. Ainda, a partir dos critérios o número de artigos foi reduzido para 9. A tabela 10 a seguir mostra os critérios de inclusão e exclusão.

A tabela contém a identificação dos critérios de inclusão e exclusão e as descrições relacionadas, a filtragem dos artigos mostrou-se satisfatória pela escolha dos critérios, com isso, os trabalhos selecionados foram analisados para buscar correlação em repositórios de versionamento de código fonte comumente utilizados (Github, Gitlab e Source Forge). Ademais, os artigos apresentaram estudos relacionados com segurança, especificamente, detecção e mitigação de anomalias por meio de IA.

Ao analisar os trabalhos selecionados, observou-se que não tratavam do contexto IoT, abordaram problemas para detecção de anomalias em tráfego de rede, detecção de ataques DDoS e flooding, assim como detecção de botnet no contexto de redes SDN internas, Cloud e 5G. Ainda mais, as soluções envolvem utilização da teoria dos jogos, modelos de aprendizado baseadas em

Tabela 10 – Tabela dos critérios de inclusão e exclusão

ID	Critério
EC1	Trabalhos não escritos em Inglês
EC2	Artigos duplicados
EC3	Estudos fora do campo da ciência da computação
EC4	Estudos secundários - o resumo ou título contém a palavra revisão ou pesquisa
EC5	Artigos não contém no título ou no resumo as palavras no contexto de segurança ou seguro ou malicioso ou cripta ou chave compartilhada ou confidencial ou ataque
EC6	Trabalhos não relatam mecanismo de segurança direta e indiretamente em redes de IoT
EC7	Trabalhos não relatam abordagens de IA
IC1	Trabalhos não excluídos por nenhum dos critérios de exclusão.

deep learning.

Uma nova fase foi incluída para buscar trabalhos e códigos fonte relacionados, com a inclusão do google nos repositórios foram localizados um conjunto de 2812 linhas contendo os endereços eletrônicos relacionados aos autores dos trabalhos. Adicionalmente, foram encontrados trabalhos com códigos fonte, entretanto nesta fase os trabalhos eram difíceis de reproduzir, pois estavam em frameworks e plataformas com implementações específicas para avaliação nos ambientes de pesquisa descritas pelos autores.

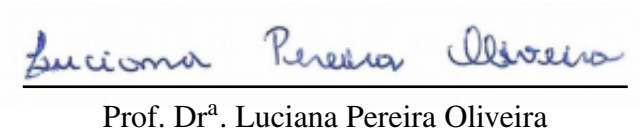
Diante dessas dificuldades, foi necessário avaliar uma nova maneira de conduzir as buscas, o trabalho de (OLIVEIRA et al., 2020) identificou alguns trabalhos no contexto de IoT, dentre os trabalhos localizados observou-se o paper de (RANDHAWA; HAMEED; MIAN, 2019), que apresenta uma abordagem de segurança por meio de um protocolo de segurança complementar sob o CoAP e da eficiência de recursos medidos por meio do consumo de energia. Entretanto, não foi localizado nenhum trabalho com código relacionado sobre esse tema. Dessa forma, o trabalho de (HINDY et al., 2020b), foi descoberto após novas pesquisas com as palavras-chave, disponibilidade, DDoS, Machine Learning. Esse estudo de caso, tem código-fonte e está relacionado com o tema dele, além de, está no contexto de IoT.

Além de encontrar o trabalho em tela, realizou-se uma reprodução dele em ambiente simulado para observar os resultados obtidos por experimentação por meio de sensores habilitados com MQTT e adaptando-o ao CoAP. Por fim, após essa etapa foi possível verificar que o trabalho mostrava-se habilitado para medição no contexto de segurança, por meio de algoritmos ML com foco em ataques de reconhecimento e acesso, além da detecção de variação energética durante a comunicação entre os dispositivos IoT.

Anexos

ENTREGA DA VERSÃO FINAL DE DISSERTAÇÃO

Eu, PROF. DR[A]. LUCIANA PEREIRA OLIVEIRA, autorizo o aluno(a) MORONI NERES VIEIRA a entregar a versão final da dissertação de mestrado, à secretaria do PPGTI, que foi por mim analisada e está de acordo com os apontamentos feitos pelos membros da banca de apresentação do referido aluno.



Prof. Dr[a]. Luciana Pereira Oliveira
Orientador

João Pessoa, 26 de Outubro de 2021.