UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE

UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RONALDO PIRES BORGES

IOT-IDS: UMA ABORDAGEM DE DETECÇÃO DE INTRUSÃO BASEADO EM APRENDIZADO DE MÁQUINA PARA SISTEMAS DE INTERNET DAS COISAS

MOSSORÓ

2024

RONALDO PIRES BORGES

IOT-IDS: UMA ABORDAGEM DE DETECÇÃO DE INTRUSÃO BASEADO EM APRENDIZADO DE MÁQUINA PARA SISTEMAS DE INTERNET DAS COISAS

Dissertação apresentada ao Mestrado em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação da Universidade do Estado do Rio Grande do Norte e Universidade Federal Rural do Semi-Árido para a obtenção do título de Mestre em Ciência da Computação.

Linha de Pesquisa: Engenharia de Software e Sistemas Computacionais.

Orientador: Silvio Roberto Fernandes de Araújo, Prof. Dr.

Coorientador: Paulo Gabriel Gadelha Queiroz, Prof. Dr.

MOSSORÓ

2024

B732i Borges, Ronaldo Pires .
IoT-IDS: uma abordagem de detecção de intrusão baseado em aprendizado de máquina para sistemas de internet das coisas / Ronaldo Pires Borges. - 2024.
82 f. : il.

Orientador: Sílvio Roberto Fernandes de Araújo.
Coorientador: Paulo Gabriel Gadelha Queiroz.
Dissertação (Mestrado) - Universidade Federal Rural do Semi-árido, Programa de Pós-graduação em Ciência da Computação, 2024.

1. Internet das Coisas. 2. segurança e privacidade. 3. Sistema de Detecção de Intrusão. 4. Aprendizado de Máquina. 5. Redes Neurais. I. Araújo, Sílvio Roberto Fernandes de, orient. II. Queiroz, Paulo Gabriel Gadelha , co-orient. III. Título.

Ficha catalográfica elaborada por sistema gerador automáto em conformidade com AACR2 e os dados fornecidos pelo) autor(a).
Biblioteca Campus Mossoró / Setor de Informação e Referência
Bibliotecária: Keina Cristina Santos Sousa e Silva
CRB: 15/120

RONALDO PIRES BORGES

IOT-IDS: UMA ABORDAGEM DE DETECÇÃO DE INTRUSÃO BASEADO EM APRENDIZADO DE MÁQUINA PARA SISTEMAS DE INTERNET DAS COISAS

Dissertação apresentada ao Mestrado em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação da Universidade do Estado do Rio Grande do Norte e Universidade Federal Rural do Semi-Árido para a obtenção do título de Mestre em Ciência da Computação.

Linha de Pesquisa: Engenharia de Software e Sistemas Computacionais.

Aprovado em: 30/08/2024

BANCA EXAMINADORA

_____
Silvio Roberto Fernandes de Araújo, Prof. Dr. (UFERSA)
Presidente

_____
Paulo Gabriel Gadelha, Prof. Dr. (UFERSA)
Membro Examinador

_____
Profª Dra. Monica Magalhaes Pereira (UFRN)
Membro Examinador

_____
Prof. Dr. Sebastião Emidio Alves Filho (UERN)
Membro Examinador

Dedico este trabalho à minha amada esposa Vivianne e filhas Luísa e Júlia, fontes de inspiração e motivação em cada passo desta jornada. Vocês são a razão de tudo o que realizo.

AGRADECIMENTOS

A Deus pela saúde concedida que me permitiu perseverar nos estudos e realizar este trabalho.

À minha família, pelo apoio incondicional em todos os momentos. Agradeço especialmente à minha esposa, Vivianne, pela compreensão e por assumir com tanto amor e dedicação a responsabilidade de cuidar e educar nossas filhas durante o meu afastamento.

À minha sogra, Maria do Socorro, minha eterna gratidão por sua generosidade em suprir minha ausência em casa, dedicando seu tempo e energia para ajudar nossa família.

Ao meu irmão, Robson Borges, pela parceria em todo o curso, pelas trocas de ideias valiosas e pelo incentivo.

Ao meu orientador, Sílvio Roberto, por seus ensinamentos, paciência e pela presença constante ao longo desta jornada. Agradeço pela parceria e pelo apoio que foram fundamentais para a realização deste trabalho.

Ao meu coorientador, Paulo Gabriel, por ser um verdadeiro guia acadêmico, sempre disposto a compartilhar seu conhecimento e orientar-me com sabedoria em cada etapa deste percurso.

Ao meu parceiro de revisão, Arthur Lennon, pela contribuição e dedicação em uma das importantes etapas deste processo.

EPÍGRAFE

“Tudo que eu queria dizer
Alguém disse antes de mim
Tudo que eu queria enxergar
Já foi visto por alguém
Nada do que eu sei me diz quem eu sou
Nada do que eu sou de fato sou eu
Tudo que eu queria fazer
Alguém fez antes de mim
Tudo que eu queria inventar
Foi criado por alguém
Nada do que eu sou me diz o que sei
Nada do que eu sei de fato é meu
...
Sempre que eu tento entender
Nada explica muito bem
Sempre a explicação me diz o que sei:
Sempre que eu sei, alguém me ensinou?
...
Agora reinvento
E refaço a roda, fogo, vento
E retomo o dia, sono, beijo
E repenso o que já li
Redescubro um livro, som, silêncio,
Foguete, beija-flor no céu,
Carrossel, da boca um dente
Estrela cadente
Tudo que irá existir
Tem uma porção de mim
Tudo que parece ser eu
É um bocado de alguém
Tudo que eu sei me diz do que sou
Tudo que eu sou também será seu”

(Móveis Coloniais de Acaju)

RESUMO

Com o uso crescente de dispositivos inteligentes em diversos mercados e aplicações, a segurança e a privacidade em sistemas de Internet das Coisas (IoT) tornam-se cada vez mais relevantes. Isso se deve ao fato de que os dados coletados e transmitidos por esses sistemas podem ter um valor significativo, inclusive financeiro. No entanto, os mecanismos de segurança e privacidade de dados já consagrados na computação em geral não se aplicam aos sistemas IoT devido às questões de arquitetura computacional e a natureza limitada desses dispositivos. A disponibilização de ferramentas para desenvolvedores de sistemas IoT que permitam construir aplicações com menor risco à privacidade dos dados se apresenta como uma proposta valiosa. Assim, este trabalho propõe uma abordagem implementada em uma ferramenta de detecção de invasão, com o objetivo de mitigar as questões de segurança e privacidade de dados inerentes aos protocolos IoT. Para isso, foram empregadas técnicas de aprendizado de máquina, com destaque para redes neurais. A avaliação da abordagem foi realizada em um ambiente com dispositivos IoT reais e simulações de ataques.

Palavras-chave: IoT, segurança e privacidade, IDS, Aprendizado de Máquina, Redes Neurais.

ABSTRACT

With the increasing use of smart devices across various markets and applications, the security and privacy of Internet of Things (IoT) systems have become increasingly relevant. This is due to the fact that the data collected and transmitted by these systems can have significant value, including financial. However, established data security and privacy mechanisms in general computing do not apply to IoT systems due to architectural issues and the limited nature of these devices. Providing tools for IoT system developers that allow for the construction of applications with reduced risk to data privacy is seen as a valuable proposition. Thus, this work proposes an approach implemented in an intrusion detection tool, aimed at mitigating the issues of data security and privacy inherent in IoT protocols. For this purpose, machine learning techniques were employed, with an emphasis on neural networks. The evaluation of the approach was conducted in an environment with real IoT devices and attack simulations.

Keywords: IoT, security and privacy, IDS, Machine Learning, Neural Networks.

LISTA DE FIGURAS

Figura 1 Segmentação de Mercado IoT.....17
Figura 2 Números reais e projeções de quantidade de dispositivos IoT 17
Figura 3 Modelos de arquitetura IoT baseados em camadas.....21
Figura 4 Comunicação IoT através da pilha de protocolos TCP/IP 24
Figura 5 Classificação dos IDSs.....27
Figura 6 Pirâmide da Internet of Things.....30
Figura 7 Taxonomia de ataques de segurança em IoT baseada em camadas. 31
Figura 8 Quantidade de ataques e vulnerabilidades conhecidas dos protocolos 33
Figura 9 Ilustração da Arquitetura do Kitsune. 41
Figura 10 Ilustração da Arquitetura do IoT-IDS. 43
Figura 11 Diagrama de Sequência representando a interação entre os principais componentes do IoT-IDS.....45
Figura 12 Arquitetura do ambiente de testes do EdgeIIoTset. 46
Figura 13 Algoritmos para gerar estatísticas de trafego de rede a partir de arquivo '.pcap'....47
Figura 14 Histograma das casses do EdgeIIoTset_netStat.....48
Figura 15 Matriz de correlação dos atributos do EdgeIIoT_netStat.....48
Figura 16 Diagrama de arquitetura do modelo netStat_classifier 50
Figura 17 Acurácia com os dados de treinamento e validação.....51
Figura 18 Relatório de classificação.....51
Figura 19 Matriz de confusão gerada com os dados de teste 52
Figura 20 Dispositivos que compõem o ambiente de testes.....61
Figura 21 Comando ip r e resultados.....63
Figura 22 Aplicação Netdiscover e resultados 63
Figura 23 Aplicação Nmap e resultados.....64
Figura 24 Execução de um ataque com a ferramenta arpspoof.....65
Figura 25 Núcleo do script de ataques DoS. 66
Figura 26 OS Fingerprinting executado com a indicação do SO. 66
Figura 27 Resultados das classificações do IoT-IDS durante pentest.....67
Figura 28 Gráficos das saídas RSME do Kitsune_IDS durante pentest. As figuras a e c representam os dados em sua escala total. As figuras b e d limitam o RSME até 5. 68

Figura 29 Consumo médio de memória e CPU sem IDS, com o IoT-IDS e com o KitsuneIDS com tráfego normal e sob ataques. 69
Figura 30 Consumo de CPU e memória ao longo do tempo sem IDS, com o IoT-IDS e com o KitsuneIDS com tráfego normal e sob ataques. 71

LISTA DE QUADROS

Quadro 1 Lista de Protocolos IoT nos níveis da pilha de protocolos da Internet. 23
Quadro 2 Descrição e objetivos de ciberataques 32
Quadro 3 Vantagens e desvantagens das abordagens de contramedidas à ataques. 35
Quadro 4 Principais características dos protocolos de aplicação IoT. 37
Quadro 5 Descrição das Estatísticas Incrementais Atenuadas calculadas. 42
Quadro 6 Dados extraídos de um pacote de rede pelo FeatureExtractorRT 43
Quadro 7 Trabalhos Relacionados 57

LISTA DE ABREVIAÇÕES E ACRÔNIMOS

COAP:	*Constrained Application Protocol*	*25*
CSA:	*Connectivity Standards Alliance*	*25*
CTI:	*Cyber Threat Intelligence*	*56*
CVE:	*Common Vulnerabilities and Exposures*	*33*
DDoS:	*Distributed Denial of Service*	*37*
DoS:	*Deny of Service*	*37*
EITF:	*Internet Engineering Task Force*	*25*
ML:	*Machine Learning*	*34*
OCF:	*Open Connectivity Foundation*	*25*
PLC:	*Programmable Logic Controllers*	*29*
RFID:	*Radio Frequency Identification*	*16*
RPL:	*Routing Protocol for Low Power and Lossy Networks*	*25*
SBC:	*Simgle Board Computer*	*61*
SCADA:	*Supervisory Control and Data Acquisition*	*29*
VM:	*Virtual Machine*	*62*

SUMÁRIO

1. INTRODUÇÃO 16
1.1. Hipótese 18
1.2. Questões de pesquisa 19
1.3. Objetivos 19
1.3.1. Objetivo geral 19
1.3.2. Objetivos específicos 19
1.4. Escopo 20
1.5. Organização do documento 20

2. FUNDAMENTAÇÃO TEÓRICA 21
2.1. Internet das Coisas 21
2.1.1. Protocolos de comunicação IoT 22
2.2. Segurança da informação 25
2.2.1. Segurança da informação em IoT 29
2.2.2. IoT e os ciberataques 31
2.2.3. Medidas de Mitigação de Ataques em IoT 33
2.3. Considerações Finais 36

3. IOT-IDS 39
3.1. Abordagem 39
3.2. Arquitetura 41
3.3. NetStatClassifier 45
3.3.1. Conjunto de dados 45
3.3.2. Preparação dos Dados 49
3.3.3. Otimização de Hiperparâmetros 50
3.3.4. Construção e Treinamento 50
3.3.5. Avaliação do Modelo 51
3.4. Trabalhos relacionados 53
3.5. Considerações finais 59

4. ESTUDO EMPIRICO 60
4.1. Protocolo 60

4.1.1. Configuração do ambiente de testes 60
4.1.2. Testes de Penetração 62
4.2. Resultados 67
4.3. Discussão 72
4.4. Considerações finais 72

5. CONCLUSÃO 74
5.1. Ameaças à validade do estudo 74
5.2. Contribuições 75
5.3. Perspectivas de Continuidade 76

REFERÊNCIAS 77

1. INTRODUÇÃO

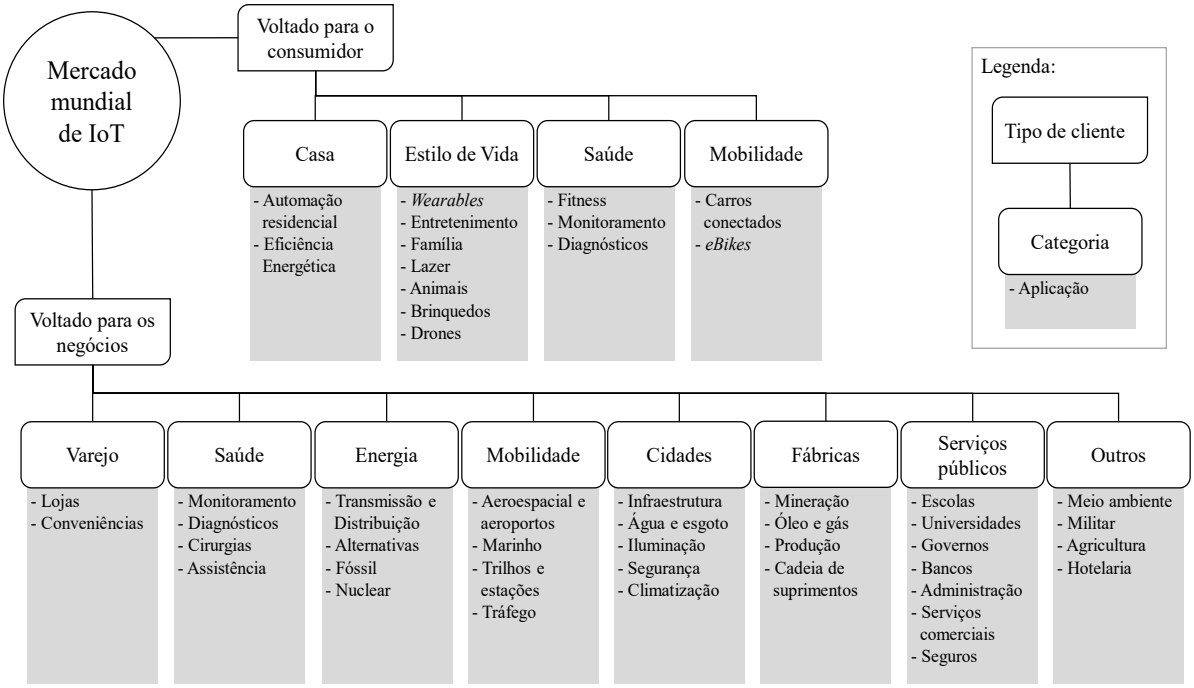
A conectividade sem fios e a miniaturização dos dispositivos eletrônicos trouxe a computação móvel e ubíqua. Enquanto a primeira refere-se à exploração da conexão de dispositivos que podem ser movimentados fisicamente, a segunda diz respeito à integração dos dispositivos de computação com o mundo físico. O termo computação ubíqua significa computação em toda parte e foi introduzido por Weiser (Coulouris *et al.*, 2013). Tal integração com o mundo físico seria uma forma nova de pensar o mundo considerando os ambientes ocupados pelas pessoas, de forma que os computadores, com formas e funções diferentes, desaparecessem em consequência da psicologia humana. Quando as pessoas aprendem o suficiente sobre algo, ou acostumam-se com isso, elas passam a usá-lo de forma inconsciente (Weiser, 1999).

Uma das formas de implementar a computação ubíqua é por meio da Internet das Coisas (do inglês *Internet of Things - IoT*). O termo foi proposto por Kevin Ashton (MIT Auto-ID) em 2000 fazendo referência à ligação de informações de etiquetas RFID (*Radio Frequency Identification*) à Internet. Sethi e Sarangi (2017) definem IoT um ambiente complexo no qual objetos inteligentes são interconectados por meio de uma rede de comunicação para coletar e trocar dados para diversos fins. Apesar da definição bastante concisa, ela pode abranger diferentes aplicações como são apresentadas na Figura 1. Nesta figura a IoT Analytics (2014) apresenta uma segmentação de mercado voltada para o consumidor e para negócios, ambos com diversos nichos.

Nos dois últimos relatórios da (IoT Analytics, 2022, 2023) as previsões do número de dispositivos IoT conectados apresentaram um desvio entre as estimativas causado pela crise global da escassez de chips iniciada em 2020 por consequência da pandemia de COVID-19 (Frieske e Stieler, 2022). Ainda assim, em 2022, houve um aumento de 18% no número de dispositivos, totalizando cerca de 14,4 bilhões de dispositivos até o final daquele ano. Em 2023, o crescimento continuou, com um aumento de 16%, resultando em aproximadamente 16,7 bilhões de dispositivos conectados. Como pode ser visto na Figura 2, as projeções indicam que até 2027 o número de dispositivos conectados globalmente poderá alcançar cerca de 29,7 bilhões.

Esse aumento no número de dispositivos da IoT tem um potencial de gerar um grandioso volume de dados. Cada dispositivo conectado atua como uma fonte contínua de informações, coletando dados sobre o ambiente, comportamento dos usuários e o estado dos sistemas em que estão integrados. Essa coleta massiva de dados é crucial, pois permite a transformação desses dados brutos em informações acionáveis, que são essenciais para diversas aplicações, como cuidados com a saúde, cidades inteligentes, indústria etc. (Bellini, Nesi e Pantaleo, 2022; Cook, Rehman e Khan, 2023).

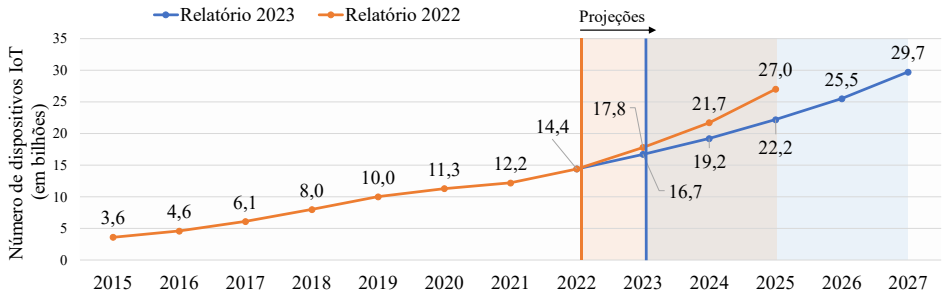
Figura 1 Segmentação de Mercado IoT



Fonte: Adaptado de IoT Analytics (2014).

A informação, que é fundamentalmente um conjunto de dados, adquire maior significado e valor quando transformada em um recurso útil. Siebel (2021) exemplifica o valor da informação ao mencionar uma empresa petrolífera que, através da implantação de um aplicativo de manutenção preditiva baseado em IA para cerca de 500 mil válvulas de refinaria,

Figura 2 Números reais e projeções de quantidade de dispositivos IoT



Fonte: Autoria própria com dados da IoT Analytics (2022, 2023)

pode gerar economias de centenas de milhões de dólares por ano em custos de manutenção e aumentar a eficiência operacional. Entretanto, a adulteração desses dados pode transformar essa economia em prejuízo, evidenciando a importância de garantir a integridade das informações. Algarni, Alkhelaiwi e Karrar (2021) , destacam que a proliferação de dispositivos IoT, muitas vezes lançados no mercado priorizando a funcionalidade em detrimento da segurança, o que resulta em um cenário vulnerável a diversos tipos de ataques.

As estratégias de segurança convencionalmente empregadas em sistemas de TI nem sempre são aplicáveis à arquitetura IoT. Um exemplo é a impossibilidade do uso de *softwares* antivírus em dispositivos como fechaduras inteligentes IoT, pois os antivírus são desenvolvidos para uma arquitetura especifica incompatível com dispositivos IoT. Garantir a execução eficiente de serviços de segurança, como autorização, autenticação, controle de acesso e privacidade, representam desafios consideráveis, ao mesmo tempo em que se busca minimizar o impacto na experiência do usuário. É importante ressaltar que aprimoramentos na segurança de um sistema frequentemente resultam em comprometimentos na performance (Moraes e Hayashi, 2021).

Em uma revisão, Algarni, Alkhelaiwi e Karrar (2021) apontam que os desafios de segurança da informação em IoT incluem:

- A heterogeneidade dos dispositivos e protocolos dificulta a interoperabilidade;
- Os recursos computacionais, de memória e energia limitados, o que complica a implementação de medidas de segurança;
- A gestão e atualização de software que são complexas em grandes escalas.
- Privacidade de dados, exigindo proteção robusta;
- Autenticação e autorização, cruciais para evitar acessos não autorizados;
- Segurança de comunicação, necessária para proteger os dados transmitidos;
- Segurança de *software* e *firmware*, que requerem desenvolvimento seguro e atualizações regulares;
- Ataques de *malware*, que ameaçam os dispositivos interconectados;
- Segurança da infraestrutura de nuvem, essencial devido à dependência de serviços em nuvem.

1.1. Hipótese

Com base no contexto e desafios apresentados, a hipótese abordada neste trabalho é que o uso de Sistemas de Detecção de Intrusão (IDS) pode ser útil na mitigação de riscos advindos parte dos desafios de segurança apontados, como a ausência de autenticação dos

dispositivos, ausência de criptografia dos dados (por escolha do desenvolvedor ou do usuário) e ataques. Sendo um IDS uma camada adicional aos métodos tradicionais como criptografia e autenticação.

1.2. Questões de pesquisa

As questões de pesquisa foram elaboradas considerando o contexto de segurança e privacidade na Internet das Coisas. Estas questões estão divididas em uma questão geral que orienta todo o estudo, questões de conhecimento que buscam entender o que já se sabe sobre o problema, e questões de projeto direcionadas para o desenvolvimento de uma solução específica.

Questões:

1. Quais tecnologias podem ser usadas para construir um IDS para sistemas IoT?
2. Até que ponto o uso de um IDS é capaz de proteger um sistema IoT?
3. Qual o nível de impacto no desempenho que o IDS pode causar?

1.3. Objetivos

O público-alvo deste trabalho são profissionais que projetam, criam e implementam soluções de *hardware* e *software* para sistemas IoT. Os objetivos foram definidos com base nas questões de pesquisa apresentadas anteriormente e são apresentados a seguir.

1.3.1. Objetivo geral

Aumentar a segurança da informação na comunicação de sistemas IoT. Para alcançar o objetivo geral alguns objetivos específicos devem ser atingidos.

1.3.2. Objetivos específicos

- Identificar e comparar as principais abordagens e tecnologias de segurança e privacidade de dados em sistemas IoT;
- Desenvolver uma abordagem de segurança da informação considerando o contexto de sistemas IoT e suas restrições;
- Implementar um componente IDS específico para IoT aplicando a abordagem desenvolvida;

- A partir da implementação realizar um estudo empírico sobre o uso do IDS em ambiente IoT e seus impactos;

1.4. Escopo

Este estudo foca na implementação de um Sistema de Detecção de Intrusão de Rede (Network Intrusion Detection System - NIDS) para ambientes de IoT, com análise de tráfego baseada em machine learning. O NIDS monitora a rede em tempo real para identificar padrões anômalos que possam indicar atividades maliciosas. Essa abordagem utiliza algoritmos de machine learning que inspecionam pacotes diretamente na pilha de protocolos TCP/IP, visando identificar potenciais ameaças. O trabalho também avalia a viabilidade de um NIDS operando em hardware com recursos limitados e analisa seu impacto no desempenho do servidor. Os capítulos 2 e 3 aprofundam o contexto, justificando as escolhas de abordagem e arquitetura.

1.5. Organização do documento

Este trabalho está organizado em capítulos. No Capítulo 2, são apresentados os fundamentos teóricos que embasam a pesquisa, incluindo uma revisão sobre a Internet das Coisas (IoT), protocolos de comunicação, e os principais desafios de segurança e privacidade associados a esses sistemas. O Capítulo 3 discute a proposta do IoT-IDS, incluído a abordagem a arquitetura do sistema, os métodos de detecção de intrusões utilizados, e as técnicas de aprendizado de máquina aplicadas. Em seguida, no Capítulo 4, é realizado um estudo empírico que avalia o desempenho do sistema proposto em cenários reais e simulados, apresentando e discutindo os resultados obtidos. Por fim, o Capítulo 5 conclui o trabalho, resumindo as contribuições da pesquisa, discutindo as limitações encontradas e sugerindo direções para trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados os conceitos básicos de Internet das Coisas (IoT), tais como a definição, a arquitetura da IoT e os protocolos de comunicação usados nesse contexto. Além disso, são discutidos os aspectos de Segurança da Informação forma geral e em IoT, além das contramedidas.

2.1. Internet das Coisas

Conceitualmente, a IoT foi apresentada na introdução como um ambiente complexo no qual objetos inteligentes são interconectados por meio de uma rede de comunicação para coletar e trocar dados para diversos fins (Sethi e Sarangi, 2017). Do ponto de vista da arquitetura, Mazon-Olivo e Pan (2022) identificam seis modelos de arquiteturas IoT, variando de três a seis camadas. A definição de arquiteturas em camadas é utilizada no contexto da IoT porque fornece uma forma estruturada e organizada de descrever e projetar os sistemas da IoT (Sethi e Sarangi, 2017).

Embora mais camadas de arquitetura forneçam mais detalhes e especificidade em termos de funcionalidade e gerenciamento, a essência das operações pode ser capturada em três camadas principais: Camada de Percepção, Camada de Rede e Camada de Aplicação. (Bakhshi, Balador e Mustafa, 2018). Na Figura 3, são ilustrados os diferentes modelos de arquitetura, identificando como a arquitetura de três camadas representa as demais. A variedade de termos

Figura 3 Modelos de arquitetura IoT baseados em camadas.

<table>
<tr><th>3 Camadas</th><th>4 Camadas</th><th>5 Camadas</th><th>Baseado em SOA</th><th>Baseado em Middleware</th><th>6 Camadas</th></tr>
<tr><td rowspan="2">Aplicação, Computação em Nuvem</td><td>Aplicação</td><td>Empresarial</td><td>Aplicação</td><td>Aplicação</td><td>Empresarial</td></tr>
<tr><td rowspan="2">Serviço, Suporte, Middleware</td><td>Aplicação</td><td>Composição de Serviço</td><td>Middleware</td><td>Aplicação</td></tr>
<tr><td>Serviço, Middleware, Processamento</td><td>Gestão de Serviços</td><td>Coordenação</td><td>Serviço, Processamento, Armazenamento</td></tr>
<tr><td rowspan="2">Rede, Computação em Névoa</td><td rowspan="2">Rede</td><td rowspan="2">Rede, Transporte, Enlace, Abstração</td><td rowspan="2">Abstração de Objetos</td><td>Backbone de Rede</td><td>Rede</td></tr>
<tr><td>Acesso</td><td>Acesso, Adaptação, Observador</td></tr>
<tr><td>Percepção, Dispositivos, WSN, Computação de Borda</td><td>Percepção, Sensor-Atuador, Detecção</td><td>Percepção, Dispositivos, Sensores e Atuadores, Objetos</td><td>Objetos</td><td>Tecnologia de Borda</td><td>Objetos, Percepção</td></tr>
</table>

Fonte: Adaptado de Mazon-Olivo e Pan (2022).

usados decorre do fato de que os autores compilaram os termos encontrados em diversos trabalhos diferentes, que coincidem em descrição.

A **Camada de Percepção**, também chamada de Computação de Borda (*Edge Computing*), lida com a coleta de dados do mundo real. Ela é composta por dispositivos equipados com sensores e/ou atuadores, responsáveis por monitorar e interagir com o ambiente físico. Os sensores capturam dados como temperatura, umidade, localização etc., enquanto os atuadores executam ações como ligar/desligar dispositivos, ajustar níveis, entre outros (Sethi e Sarangi, 2017; Swamy e Kota, 2020).

Na **Camada de Rede**, ou Computação em Névoa (*Fog Computing*), o foco muda para a transmissão de dados. Esta camada inclui gateways, roteadores e protocolos de comunicação. O papel desta camada é garantir a conectividade entre os dispositivos da camada de percepção, a Internet e a camada de aplicação. É aqui que os dados coletados são agregados, pré-processados e direcionados para o local de processamento (Sethi e Sarangi, 2017; Swamy e Kota, 2020; Mazon-Olivo e Pan, 2022). Os protocolos de comunicação são abordados na próxima seção.

Na **Camada de Aplicação**, ou Computação em Núvem (*Cloud Computing*), é onde os dados ganham significado e são utilizados para gerar valor. Essa camada abriga os softwares, plataformas e interfaces que processam, analisam e apresentam as informações coletadas nas camadas anteriores. É nesta camada que as regras de negócio são aplicadas e os serviços de IoT são executados, como, por exemplo, *Dashboards*, alertas, automações e tomadas de decisões (Sethi e Sarangi, 2017; Swamy e Kota, 2020; Mazon-Olivo e Pan, 2022).

2.1.1. Protocolos de comunicação IoT

Para entender melhor a camada de rede da arquitetura IoT, podemos usar como referência a pilha de protocolos da Internet (ou Pilha TCP/IP), que é o conjunto de protocolos mais amplamente usado para comunicação na Internet. Esse modelo organiza os protocolos em cinco níveis (ou camadas): nível físico, nível de enlace, nível de rede, nível de transporte e nível de aplicação. Cada nível provê seu serviço executando certas ações dentro dele e utilizando os serviços do nível diretamente abaixo dele, isso é denominado como modelo de serviço. Cada nível tem uma função específica na comunicação entre dispositivos, desde o estabelecimento da conexão física até a entrega dos dados na forma adequada para a aplicação (Kurose e Ross,

2021). No Quadro 1 são listados os protocolos de comunicação usados em cada nível do modelo TCP/IP em sistemas IoT.

Quadro 1 Lista de Protocolos IoT nos níveis da pilha de protocolos da Internet.

Pilha de protocolos da Internet (Modelo TCP/IP)	**Protocolo IoT**
5. Aplicação	AMQP, CoAP, DDS, HTTP, Matter, MQTT e XMPP
	DTLS, SSL, SASL, TLS, QUIC e MRP
4. Transporte	BTP, TCP e UDP
3. Rede	6LoWPAN, IPv4, IPv6 e RPL
2. Enlace	BLE, Ethernet, GSM, HomePlugGP, LPWAN, LTE, Thread, Wi-Fi ZigBee e Z-Wave
1. Física	IEEE 802.3, IEEE 802.11 e IEEE 802.15.4 (cabos de cobre, fibras óticas e ondas eletromagnéticas)

Fontes: (Sharma e Gondhi, 2018; Mishra e Kertesz, 2020; Kurose e Ross, 2021; Connectivity Standards Alliance, 2022)

Os sistemas IoT utilizam a pilha de protocolos para se comunicar da mesma forma que qualquer outra aplicação de rede ou Internet. Embora os nomes das camadas de aplicação e de rede apareçam nos modelos da arquitetura IoT e TCP/IP, elas não são exatamente equivalentes.

Daqui em diante, será utilizado o termo ‘nível’ para referir às camadas do modelo TCP/IP, com o objetivo de prevenir qualquer confusão.

Como descrito na seção anterior, a camada de rede na arquitetura IoT trata da transmissão dos dados coletados pela camada de percepção para outros dispositivos ou sistemas de processamento. Portanto, a camada de rede engloba funções que estão distribuídas nos níveis físico, enlace, rede e transporte do modelo TCP/IP, onde especificamente:

- **O nível físico:** Trata da transmissão e recepção de bits brutos através do meio físico de comunicação. Ela define as características elétricas, ópticas ou de radiofrequência do meio, bem como os métodos de codificação dos dados (Kurose e Ross, 2021);
- **O nível de enlace**: Responsável pela comunicação entre nós adjacentes em um mesmo enlace de comunicação. Ela encapsula datagramas em quadros, controla o acesso ao meio físico e pode fornecer mecanismos de detecção e correção de erros (Kurose e Ross, 2021);

- **O nível de rede**: É responsável pelo roteamento dos pacotes (datagramas) através da rede, endereçando origem e destino e definindo rotas de acesso entre diferentes redes. Essas tarefas são executadas tanto nos *hosts* quanto nos roteadores (Kurose e Ross, 2021).
- **A nível de transporte**: Responsável pela comunicação entre processos em execução em hospedeiros diferentes. Ela garante a entrega confiável de dados, controla o fluxo de dados e divide os dados da aplicação em segmentos (Kurose e Ross, 2021).

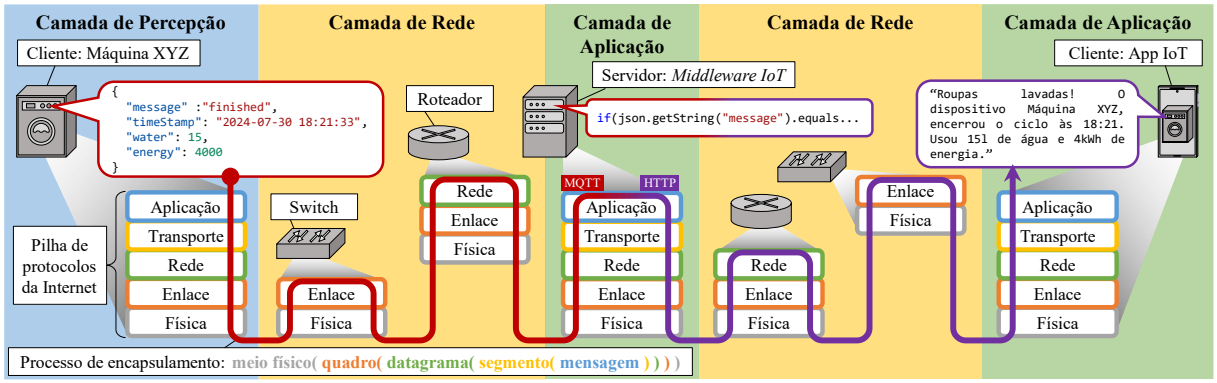
Assim como as camadas de percepção e aplicação da arquitetura IoT, podem ser compreendidas como nível de aplicação da pilha TCP/IP, pois:

- **O nível de aplicação:** Lida com a interação direta com as aplicações do usuário. Ela define os protocolos usados pelas aplicações para trocar mensagens, como HTTP e CoAP (Kurose e Ross, 2021).

Logo, um *host* IoT, equipado com um sensor, envia seus dados a um servidor na forma de mensagens. Portanto, assim como outros tipos de comunicação através da Internet, os protocolos de comunicação de dados para IoT usam dos níveis da pilha de protocolos TCP/IP (Swamy e Kota, 2020; Mazon-Olivo e Pan, 2022) . Na Figura 4 está ilustrada a comunicação de um dispositivo IoT até um *Middleware* e deste até uma aplicação cliente, ambos através da pilha de protocolos da Internet. A figura ilustra também como os componentes do sistema estão posicionados nas camadas da arquitetura IoT.

Vale ressaltar que vários padrões e formas de comunicação foram desenvolvidos e disponibilizados no mercado, resultando em uma indústria fragmentada e dificultando a interoperabilidade entre dispositivos de diferentes fornecedores. Para sanar essa dificuldade

Figura 4 Comunicação IoT através da pilha de protocolos TCP/IP



Fonte: Adaptado de Kurose e Ross (2021) e Swamy e Kota, (2020)

grandes alianças de padronização, como por exemplo a *Connectivity Standards Alliance*[1](CSA), *LoRa Alliance*[2] e *Thread Group*[3], foram formadas com base nos interesses de seleções de tecnologia e mercados comerciais (Sheng *et al.*, 2013; Zegeye, Jemal e Kornegay, 2023). Assim alguns protocolos de comunicação precisam de adaptações como *gateways* para compatibilizarem com a pilha TCP/IP, como os padrões BLE, LPWAN, ZigBee e Z-Wave (Swamy e Kota, 2020).

As limitações de potência e largura de banda dos dispositivos representam desafios na implantação do IPv6. Com extensas sobrecargas de protocolo contra memória e limitações computacionais dos dispositivos IoT, a *Internet Engineering Task Force* (IETF) assumiu a liderança na padronização de protocolos de comunicação para dispositivos com recursos limitados e desenvolveu vários protocolos de Internet, incluindo o *Routing Protocol for Low Power and Lossy Networks* (RPL), o *Constrained Application Protocol* (CoAP) e MAC IEEE802.15.4 entre outros (Palattella *et al.*, 2013; Sheng *et al.*, 2013). Há outras iniciativas para a criação de padrões abertos e protocolos de rede comuns, como por exemplo, o *Open Connectivity Foundation* (OCF)[4].

2.2. Segurança da informação

Para trafegar qualquer tipo de dados/informações em redes locais ou na Internet é importante que estes estejam seguros, mas como saber se a segurança está presente em uma comunicação de dados? Para isto são considerados três propriedades fundamentais para a segurança de dados: a **Confidencialidade**, **Integridade** e **Disponibilidade** (Kim e Solomon, 2014).

Resumidamente, implementar a *CIA triad*[5] garante: 1º – Que o conteúdo de uma mensagem seja acessado apenas pelas partes endereçadas e autorizadas; 2º – Que a mensagem não sofra nenhum tipo de alteração proposital, que chegue ao destino exatamente como saiu da

[1] csa-iot.org
[2] lora-alliance.org
[3] threadgroup.org
[4] openconnectivity.org
[5] *Confidentiality, Integrity and Availability*

origem; 3º – Que o sistema ou mensagem estará acessível quando (ou enquanto) o usuário precisar (Moraes, 2010; Moraes e Hayashi, 2021).

De forma geral, devem ser realizadas três tipos de ação no combate aos possíveis problemas (Fontes, 2012). São elas:

- **Ações preventivas**: Medidas implementadas para evitar violações de segurança, focando no fortalecimento dos sistemas e na redução da probabilidade de ataques bem-sucedidos. Exemplos incluem criptografia, autenticação robusta, *Firewalls,* atualizações de segurança, *backups* periódicos, treinamento e conscientização em segurança, e testes de penetração (Bilal, 2017; Cheruvu *et al.*, 2020; Bang *et al.*, 2022; Araya e Rifà-Pous, 2023).
- **Ações detectivas**: Medidas adotadas para detectar violações de segurança que já ocorreram ou estão em andamento. O objetivo é identificar atividades suspeitas e fornecer alertas para uma resposta rápida. Ferramentas como Sistemas de Detecção de Intrusão (IDS), análise de logs, monitoramento da integridade de arquivos e *honeypots*[6] são exemplos (Bijone, 2016; Araya e Rifà-Pous, 2023).
- **Ações corretivas**: Medidas tomadas para mitigar o impacto de uma violação de segurança já ocorrida, visando restaurar sistemas e dados ao seu estado normal de operação e prevenir recorrências. Exemplos incluem isolamento e reparação de sistemas, redefinição de senhas e credenciais, análise de causa raiz e relato de incidentes para ações preventivas futuras (Bijone, 2016; Krishna *et al.*, 2021; Bang *et al.*, 2022).

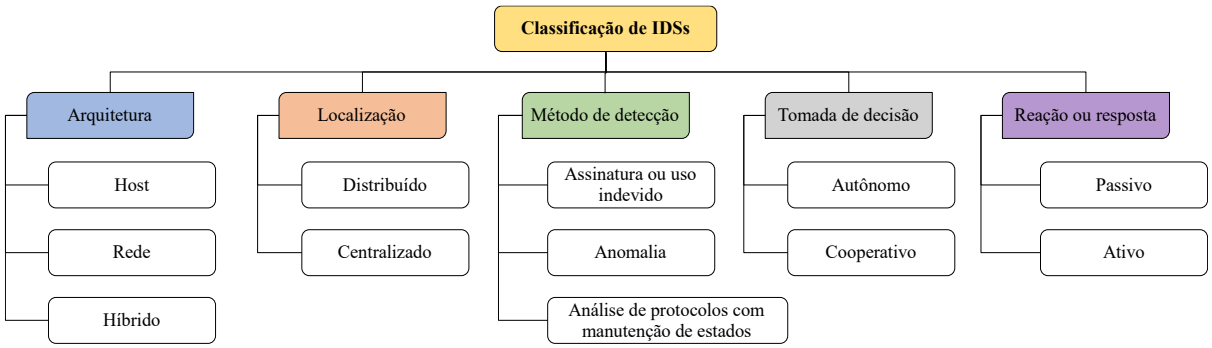
Para garantir a segurança das redes de computadores, diversas camadas de proteção podem ser implementadas. Uma das primeiras linhas de defesa é o uso de firewalls, que funcionam como filtros examinando o tráfego de entrada e saída com base em regras de segurança pré-definidas, restringindo o acesso apenas ao tráfego autorizado. Essa filtragem pode ser realizada de várias maneiras, incluindo endereços IP, portas de origem e destino, protocolos de comunicação e estado da conexão. No entanto, firewalls operam apenas no nível de rede do modelo TCP/IP. (Kurose e Ross, 2021).

[6] Armadilhas projetadas para atrair atacantes e coletar informações sobre suas táticas, técnicas e procedimentos. (Bijone, 2016)

No entanto, para detectar muitos tipos de ataques, é necessária uma inspeção profunda de pacote, no nível dos dados das aplicações e isso é realizado por um IDS. Este tipo de sistema realiza uma análise mais profunda, inspecionando o conteúdo dos pacotes para identificar atividades suspeitas. Essa análise se baseia em assinaturas de ataques conhecidos ou em anomalias estatísticas que diferem do comportamento padrão da rede. Uma vez que uma atividade suspeita é detectada, o IDS pode gerar alertas ou até mesmo tomar medidas para bloquear o tráfego malicioso (Butun, Osterberg e Song, 2020; Chauhan, Singh e Jain, 2020; Araya e Rifà-Pous, 2023).

Os IDSs podem ser classificados considerando aspectos como arquitetura, localização, método de detecção, tomada de decisão e reação (Bijone, 2016). Na Figura 5 esta classificação é exposta com mais detalhes.

Figura 5 Classificação dos IDSs.



Fonte: Adaptado de Bijone (2016).

No que diz respeito à arquitetura, os sistemas de detecção de intrusões podem ser baseados em *host* (HIDS), que geralmente consistem em um *software* instalado em um dispositivo e monitoram a atividade apenas nesse sistema específico, ou baseados em rede (NIDS), que acompanham os pacotes que trafegam em um *link* específico da rede, podendo ser um *hardware* dedicado ou um *software* executado em um computador conectado à rede. Existe também a arquitetura híbrida, que proporciona versatilidade e aprimora a segurança ao integrar sensores de IDS em diversas localizações, reportando ataques direcionados a segmentos específicos da rede ou à sua totalidade (Bijone, 2016).

Quanto a localização um IDS pode ser Centralizado ou Distribuído. O IDS Distribuído (DIDS) realiza a análise dos dados em múltiplos locais, adaptando-se ao número de hosts monitorados. Por outro lado, o IDS Centralizado realiza a análise dos dados em um número fixo de locais, independentemente da quantidade de hosts monitorados (Bijone, 2016).

Os IDS distribuídos apresentam vantagens em comparação aos sistemas centralizados, incluindo escalabilidade, resiliência, detecção de anomalias localizadas e menor latência. Eles lidam melhor com redes em expansão, distribuindo a carga de trabalho entre sensores ou agentes, e são menos vulneráveis a falhas e ataques direcionados. Além disso, os IDS distribuídos detectam atividades anômalas em áreas específicas e oferecem menor latência na detecção de atividades maliciosas graças à distribuição de sensores ou agentes pela rede (Snapp *et al.*, 1991; Debar, Dacier e Wespi, 1999; Janakiraman, Waldvogel, e Qi Zhang, 2003; García-Teodoro *et al.*, 2009).

Os principais desafios na implementação de um IDS distribuído incluem a coordenação e comunicação eficientes entre os sensores ou agentes, a gestão da carga de trabalho e a necessidade de algoritmos de detecção e correlação avançados para analisar os dados coletados de várias fontes (Debar, Dacier e Wespi, 1999; García-Teodoro *et al.*, 2009).

Já em relação à abordagem ou método de detecção, a classificação dos IDSs quanto ao método de detecção inclui três abordagens principais: Detecção de Assinatura, Detecção de Anomalias e Análise de Protocolo com Estado. Na Detecção de Assinatura, o IDS utiliza regras ou listas estáticas para comparar informações coletadas com bancos de dados de ataques conhecidos, identificando assim apenas intrusões previamente catalogadas. Em contraste, a Detecção de Anomalias se baseia nas estatísticas do tráfego normal da rede monitorada, usando frequentemente modelos de aprendizado de máquina para identificar padrões de comportamento incomuns que possam indicar tentativas de ataque. Essa abordagem, no entanto, pode falhar ao detectar ataques conhecidos se estes não se desviarem significativamente do comportamento normal estabelecido. Por fim, a Análise de Protocolo com Estado avalia a sequência de eventos e o estado atual das conexões, identificando atividades suspeitas ao verificar desvios em relação a perfis de comportamento benigno aceitos e especificações de protocolo. Essas abordagens oferecem diferentes níveis de eficácia e precisão ao lidar com diversos tipos de ameaças (Bijone, 2016; Eriza e Survadi, 2021).

Com base na reação ou resposta, os IDS podem ser classificados em Resposta Ativa e Resposta Passiva. Na Resposta Passiva, o IDS registra informações sobre possíveis violações de segurança e emite alertas, sem interromper ativamente as intrusões. Vários métodos de notificação, como e-mail, podem ser utilizados. Já na Resposta Ativa, o IDS toma medidas em resposta às intrusões detectadas, como aumentar a sensibilidade para obter mais informações

sobre o ataque e o invasor, ou realizar mudanças na configuração de sistemas e dispositivos de rede para bloquear a intrusão (Bijone, 2016).

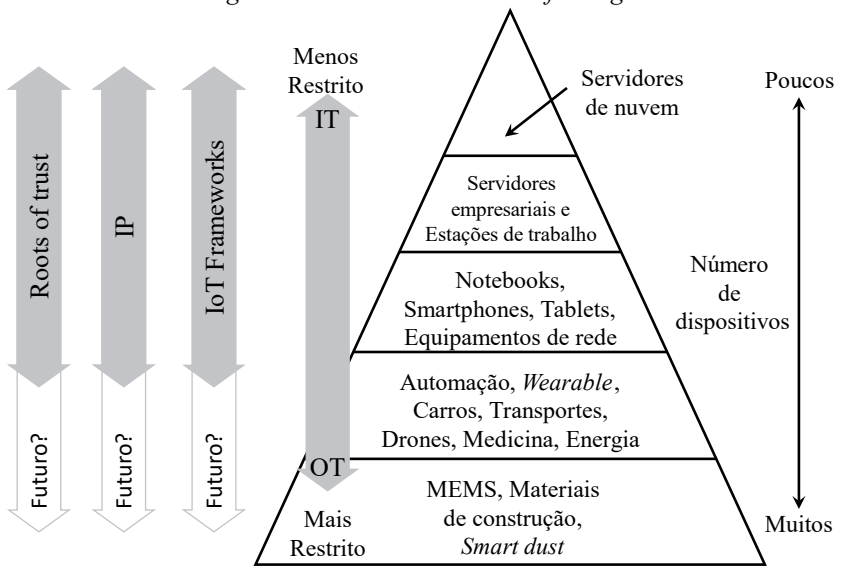
O IDS ideal deve ser eficiente, simples, preciso e abrangente, detectando todos os ataques com mínima penalidade de desempenho. As abordagens de design para um IDS podem incluir filtragem de cabeçalhos de pacotes, filtragem de conteúdo de pacotes, manutenção do estado da conexão, utilização de assinaturas complexas de múltiplos pacotes e emprego de um número mínimo de assinaturas com impacto máximo. Além disso, deve ser capaz de filtrar em tempo real e de forma online, ocultar sua presença e ajustar o tamanho ideal da janela de tempo variável para corresponder às assinaturas (Bijone, 2016).

2.2.1. Segurança da informação em IoT

Cheruvu (2020) apresenta um relato de um ataque bem-sucedido do *malware "Stuxnet"* em uma instalação nuclear no Irã em 2010. O processo de enriquecimento de urânio dependia de sistemas SCADA (*Supervisory Control and Data Acquisition*) que incorporam protocolos de mensagens (MQTT, AMQP e DDS) e PLC (*Programmable Logic Controllers*) com interfaces USB e Modbus (um protocolo de comunicação utilizado em automação industrial). O *Stuxnet* foi propagado por sites da web sobre futebol e conseguiu programar unidades *flash* USB para atualizar os PLCs das centrífugas de enriquecimento. O ataque, que controlava os equipamentos de forma sutil, causou danos físicos às centrífugas fazendo com que elas funcionassem em velocidades superiores aos limites. O caso monstra como sistemas de informação podem afetar sistemas operacionais de forma a prejudicar sistemas físicos, infraestrutura, meio ambiente e até a vida humana utilizando eletrônicos e *software* disponíveis. Com isso sigla a IoT pode assumir um significado adicional unindo *Information Technology (IT)* com *Operational Technology (OT)* em *Informational and Operational Technology (IOT)*.

Como visto, as projeções da (IoT Analytics, 2022, 2023), indicam que em três anos o número de dispositivos conectados poderá alcançar próximo 30 bilhões em todo o mundo. O conceito fundamental do design de segurança consiste em reduzir a superfície de ataque para minimizar o risco de interações não previstas (Cheruvu *et al.*, 2020). Na Figura 6, é ilustrado que a superfície de ataque se amplia significativamente conforme a quantidade de dispositivos conectados aumenta nos níveis inferiores da pirâmide.

Figura 6 Pirâmide da Internet of Things.



Fonte: Adaptado de Cheruvu et al.(2020)

Arquiteturas de segurança tradicionais não existem na arquitetura IoT. É desafiador garantir a implementação dos serviços de segurança, como autorização, autenticação, controle de acesso e privacidade, reduzindo os efeitos na usabilidade do usuário. Quanto mais seguro é um sistema, mais penalizada é a performance e vice-versa (Moraes e Hayashi, 2021).

Morais e Hayashi (2021) destacam que a segurança em IoT não deve ser tratada de forma diferente de outras tecnologias, e que os princípios fundamentais de ainda são aplicáveis. Para garantir a segurança em IoT, é necessário abordar aspectos tecnológicos, processos e pessoas, adaptando esses princípios aos desafios apresentados pelo uso dessas tecnologias.

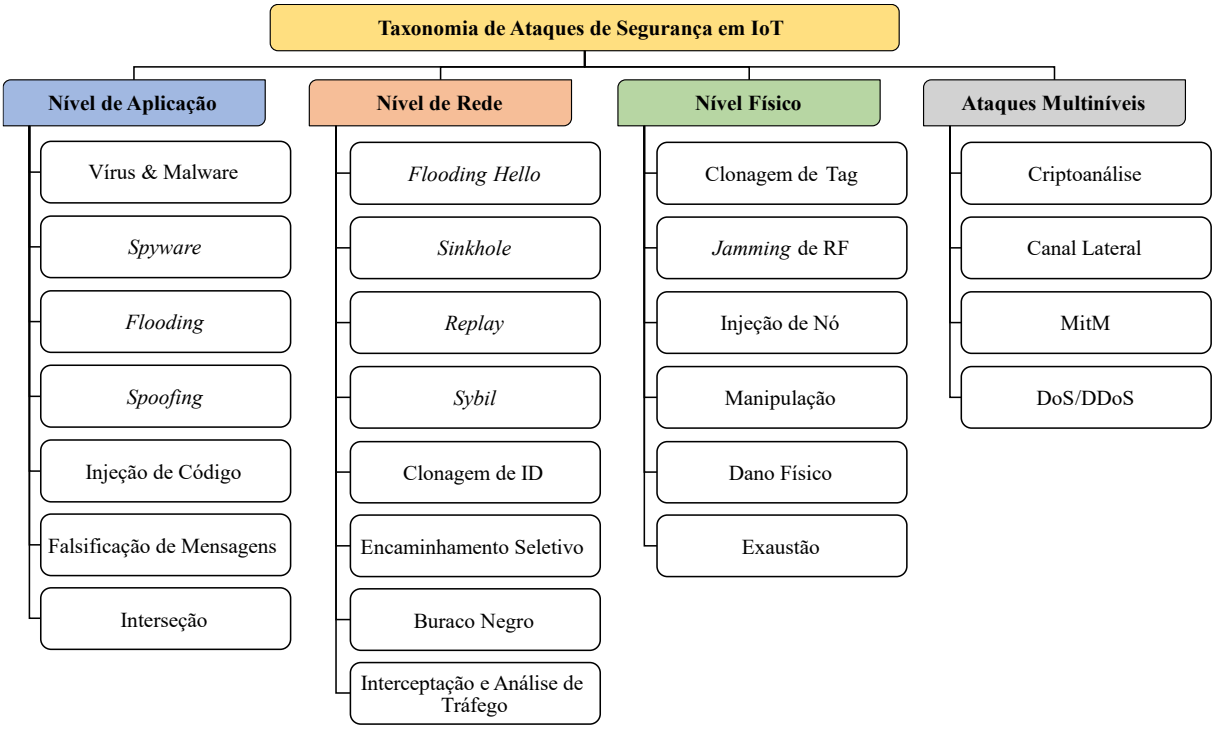
Diversos estudos têm proposto soluções baseadas em aprendizado de máquina e aprendizado profundo para detecção e prevenção de intrusões em redes IoT. Essas soluções incluem o uso de algoritmos como *K-means*, *Decision Tree*, *autoencoders*, *Support Vector Machines* (SVM), *Random Forest*, *Deep Neural Networks* (DNN), *Extreme Gradient Boosting* (XGBoost) e *Long Short-Term Memory* (LSTM) (Curvello, 2015).

2.2.2. IoT e os ciberataques

Para proteger os sistemas IoT é preciso conhecer os ataques e seus objetivos. Assim, o primeiro passo para construir uma estrutura de defesa é identificar todos os ataques que possam comprometer a segurança ou a privacidade dos ativos do IoT (Akram, Konstantas e Mahyoub, 2018).

Khanam et al., (2020) propõem uma taxonomia que classifica os ataques a sistemas IoT com base nos níveis de aplicação, rede e física da pilha de protocolos da Internet. Alguns desses ataques são categorizados como ataques multicamadas, uma vez que exploram mais de uma camada da comunicação de rede. Essa classificação é ilustrada na Figura 7, onde os ataques são listados de acordo com as respectivas categorias.

Figura 7 Taxonomia de ataques de segurança em IoT baseada em camadas.



Fonte: Adaptado de Khanam et al., (2020)

No Quadro 2 são apresentadas a descrição e objetivo dos ataques. Como não é objetivo deste trabalho detalhar todos os tipos de ataques, serão considerados apenas os que ocorrem na camada de aplicação.

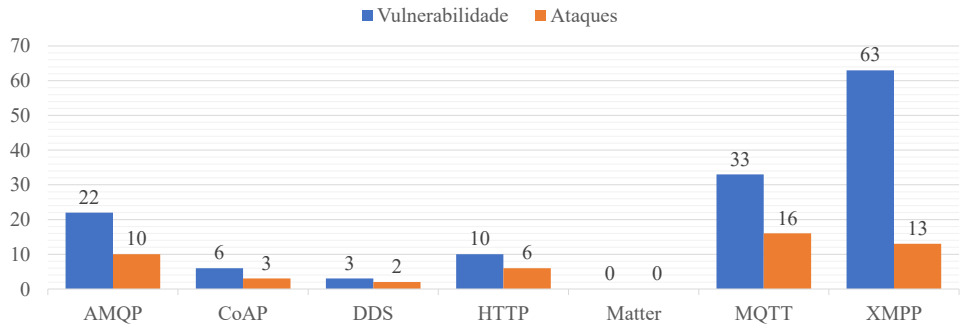
Quadro 2 Descrição e objetivos de ciberataques

Ataque	Descrição	Objetivo
Vírus e Malware	São ataques que têm como alvo o sistema e geralmente ocorrem na forma de aplicativos, como *Trojans*, *Spams*, *Worms* e outros tipos. Representam um risco significativo para dispositivos iot de alta tecnologia (que excutam aplicativos)	Violar a privacidade e obter vantagens indevidas
Spyware	É um programa instalado nos dispositivos dos usuários sem o consentimento deles. Não causa danos diretos aos dispositivos ou usuários.	Espionar ou monitorar o comportamento dos usuários e coletar informações sensíveis e enviá-las de volta ao distribuidor
Spoofing	Ocorre quando um atacante se passa por um nó válido na rede iot, enganando outros dispositivos ou nós para pensar que estão se comunicando com um nó legítimo. Isso pode levar a acessos não autorizados, manipulação de dados ou execução de atividades maliciosas	Obter acesso não autorizado e realizar ações maliciosas se passando por um nó legítimo.
Injeção de Código	Inserir código malicioso em um aplicativo ou sistema inteligente aproveitando programas com falhas.	Ganhar acesso não autorizado, roubar dados sensíveis dos usuários, assumir o controle do sistema ou transmitir *worms*
Falsificação de Mensagens	Ocorre quando um nó malicioso modifica ou cria uma mensagem para transmitir conteúdo diferente do original.	Enganar os destinatários ao fornecer informações falsas ou manipuladas.
Interseção	Tem como alvo a privacidade do sistema, obtendo informações secundárias a partir dele. Os atacantes coletam essas informações de fontes de terceiros ou registros públicos a fim de vinculá-las às informações de privacidade anonimizadas do sistema.	Identificação de informações confidenciais e a vinculação de dados aparentemente não relacionados.
Criptoanálise	O criptoanalista ou atacante tenta acessar uma mensagem criptografada sem possuir a chave de criptografia. Um exemplo desse tipo de ataque é o ataque de Força Bruta.	Quebrar a criptografia de uma mensagem e obter acesso não autorizado ao seu conteúdo.
Canal Lateral	Durante o processo de operação de criptografia, o atacante obtém informações e realiza um processo de engenharia reversa para coletar as credenciais criptográficas de um dispositivo IoT. Essas informações podem ser obtidas por meio de diferentes canais laterais, como análise de consumo de energia, análise temporal ou análise de emissão eletromagnética	Obter informações confidenciais, como as chaves criptográficas e acesso não autorizado a um sistema.
MitM (Homem no Meio)	O adversário se posiciona entre dois dispositivos para obter acesso ao texto cifrado e quebrar o sistema de criptografia a fim de encontrar a chave de criptografia. Uma vez obtido acesso ao texto em claro, o criptoanalista pode alterar a mensagem entre as duas partes sem o consentimento delas.	comprometer a integridade e a confidencialidade da comunicação entre os dispositivos e poder interceptar e monitorar o tráfego de dados, obter informações privadas e até mesmo manipular as mensagens transmitidas.
DoS/DDoS	Esses ataques podem ocorrer de várias formas, sendo uma delas a geração de um grande tráfego de rede e o envio de uma enorme quantidade de solicitações para a vítima tornando o serviço inacessível aos usuários.	Tornar os dispositivos, softwares, serviços de rede e recursos indisponíveis para os consumidores-alvo.

Fonte: (Khanam *et al.*, 2020)

Outra abordagem para entender e tratar da segurança de sistemas IoT é identificar as fragilidades dos protocolos envolvidos. Bang et al. (2022) discutem as vulnerabilidades de maneira sistemática e utilizaram o sistema *Common Vulnerabilities and Exposures (CVE)*[7] como referência para analisar e discutir a causa e os efeitos dos ataques possíveis nos protocolos da IoT. O sistema CVE lista e categoriza as vulnerabilidades para facilitar a análise das causas e efeitos dos possíveis ataques. Esses ataques, no contexto da IoT, referem-se a ações maliciosas que exploram essas vulnerabilidades para causar danos ou interrupções. Alguns exemplos incluem *bypass* de controle de acesso em XMPP e MQTT, execução arbitrária de código em HTTP REST e Spring AMQP, e ataques de injeção de solicitação HTTP usando informações de token de autenticação de API RESTful, entre outros. (Bang *et al.*, 2022).

Figura 8 Quantidade de ataques e vulnerabilidades conhecidas dos protocolos



Fonte: Adaptado de Bang et al (2022).

O gráfico da Figura 8 é uma adaptação dos dados apresentados por Bang et al. (2022) para os protocolos em nível de aplicação. O protocolo Matter aparece com zero vulnerabilidades e ataques por ser um protocolo relativamente novo, apresentado em novembro de 2022, até o momento de a redação deste trabalho não existirem estudos disponíveis que revelem estes dados.

2.2.3. Medidas de Mitigação de Ataques em IoT

Várias abordagens de medidas de mitigação de ataques foram desenvolvidas, visando proteger a integridade, confidencialidade e disponibilidade dos sistemas IoT. Essas contramedidas podem ser categorizadas em soluções baseadas em Machine Learning,

[7] O CVE é parte do *National Vulnerability Database (NVD)*, que é um banco de dados mantido pelo *National Institute of Standards and Technology* (NIST) dos Estados Unidos.

abordagens autônomas e métodos de criptografia, cada uma oferecendo vantagens e enfrentando desafios específicos (Butun, Osterberg e Song, 2020; Khanam *et al.*, 2020).

Soluções de *Machine Learning (ML)* são eficazes para detectar atividades maliciosas em tempo real, aproveitando a capacidade de resolver problemas complexos (Khanam *et al.*, 2020). Abordagens autônomas, também conhecidas como *self-secure*, envolvem mecanismos de autoproteção e autorrecuperação, permitindo que sistemas IoT se defendam automaticamente contra ataques sem intervenção humana, o que é crucial devido à natureza desassistida desses dispositivos (Khanam *et al.*, 2020). Métodos de criptografia, como criptografia simétrica e assimétrica, garantem a confidencialidade e integridade dos dados em trânsito e em repouso, protegendo contra acessos não autorizados (Khanam *et al.*, 2020).

A eficácia de cada categoria de contramedida depende do contexto da aplicação e das limitações dos dispositivos IoT, como poder de processamento, memória e energia, que dificultam a implementação de protocolos de segurança robustos (Butun, Osterberg e Song, 2020; Khanam *et al.*, 2020).

2.2.3.1. Soluções baseadas em *Machine Learning*

O aprendizado de máquina é um subcampo da inteligência artificial que permite aos computadores aprenderem sem programação explícita, por meio da descoberta sistemática de padrões em conjuntos de dados ou experiências anteriores. Em vez de depender de instruções passo a passo, os algoritmos de ML aprendem com os dados, identificando padrões e fazendo previsões (Cai *et al.*, 2018).

ML e *Deep Learning* (DL) são eficazes na detecção de intrusões por meio da análise de dados e previsão de comportamento de um sistema. Algoritmos como *Decision Tree (DT), Random Forest (RF), Deep Belief Network (DBN) e Convolutional Neural Networks (CNN)* são exemplos de técnicas de aprendizado usadas para segurança em IoT. Esses algoritmos aprendem com dados históricos, detectando padrões e tendências que podem indicar atividades maliciosas. Embora eficazes, as soluções baseadas em aprendizado dependem da qualidade e disponibilidade de dados do mundo real para treinamento, o que pode ser um desafio em ambientes IoT (Khanam *et al.*, 2020).

2.2.3.2. Abordagens autônomas

Também conhecidas como autoproteção ou autocura, visam minimizar a intervenção humana na segurança de IoT. Essas abordagens utilizam um ciclo de controle MAPE (**M**onitoramento, **A**nálise, **P**lanejamento e **E**xecução) para gerenciar recursos de segurança. Por exemplo, o sistema MAPE pode detectar e mitigar ataques DoS (Negação de Serviço) estabelecendo barreiras de conexão ou controlando remotamente nós suspeitos. O desafio na implementação de soluções autônomas reside na necessidade de estruturas cognitivas complexas e compatibilidade com protocolos de comunicação dinâmicos em ambientes heterogêneos, o que pode ser difícil devido às limitações de recursos dos dispositivos IoT.(Khanam *et al.*, 2020).

2.2.3.3. Criptografia

As soluções utilizam métodos matemáticos para proteger dados e comunicações em IoT, com criptografia simétrica (chave secreta) e assimétrica (chave pública) sendo os dois tipos principais. Criptografia simétrica, como AES e DES, usa uma única chave para criptografar e descriptografar dados, tornando-a mais rápida e eficiente para dispositivos IoT com recursos limitados. Criptografia assimétrica, como RSA e ECC, usa um par de chaves (pública e privada) para maior segurança, mas exige mais recursos computacionais. A escolha da criptografia depende do caso de uso específico e das restrições de recursos (Khanam *et al.*, 2020).

De forma resumida e simplificada o Quadro 3 apresenta vantagens e desvantagens das abordagens apresentadas.

Quadro 3 Vantagens e desvantagens das abordagens de contramedidas à ataques.

Abordagem	Vantagens	Desvantagens
IDS baseado em ML	**Detecção de intrusões**: Classificação de atividades normais e anormais em tempo real mesmo que desconhecidas. **Adaptabilidade**: Capacidade de aprender com novos dados e aprimorar a detecção de ataques com o tempo (dependendo do algoritmo). **Variedade de algoritmos**: Disponibilidade de algoritmos como DT, RNN, SVM etc.	**Recursos computacionais**: Algoritmos complexos exigem alto poder de processamento e memória. **Dependência de dados**: A eficácia depende da qualidade e quantidade de dados de treinamento. **Falsos positivos**: Possibilidade de classificar erroneamente atividades legítimas.
Abordagens Autônomas	**Autogestão**: Capacidade de detectar e responder a ataques sem intervenção humana, agilizando a recuperação. **Gerenciamento de recursos**: Otimização do uso de recursos como energia e memória em dispositivos IoT. **Atualizações de segurança**: Implementação automatizada de patches e atualizações para corrigir vulnerabilidades.	**Complexidade de implementação**: Exige desenvolvimento e integração complexos de software e hardware. **Limitações em ambientes dinâmicos**: Podem ter dificuldades em se adaptar a ambientes com mudanças rápidas e imprevistas. **Custos:** Implementação e manutenção podem ser dispendiosas, especialmente em soluções complexas.
Métodos de Criptografia	**Confidencialidade**: Proteção de dados em trânsito e em repouso contra acesso não	**Consumo de recursos:** Algoritmos de criptografia, principalmente os assimétricos,

Abordagem	Vantagens	Desvantagens
	autorizado, garantindo a privacidade das informações. **Integridade:** Verificação da autenticidade dos dados e garantia de que não foram adulterados. **Autenticação:** Verificação da identidade de dispositivos e usuários para evitar acessos indevidos.	podem exigir alto poder de processamento e energia. **Gerenciamento de chaves**: A segurança depende da gestão eficiente e segura das chaves criptográficas. **Vulnerabilidades:** Falhas na implementação ou algoritmos desatualizados podem comprometer a segurança.

Fontes: (Butun, Osterberg e Song, 2020; Khanam *et al.*, 2020)

2.3. Considerações Finais

É importante ressaltar que a segurança em IoT não se limita a soluções tecnológicas e requer uma abordagem holística que englobe aspectos físicos, de software e de gerenciamento (Cheruvu *et al.*, 2020). A proteção física de dispositivos, o desenvolvimento seguro de software e a implementação de políticas de segurança robustas são cruciais para a segurança geral do ecossistema IoT.(Bilal, 2017; Cheruvu *et al.*, 2020).

A questão da segurança e privacidade de dados em dispositivos de IoT tem-se destacado como um campo de pesquisa em expansão. Tem-se observado um enfoque particular no desenvolvimento de soluções baseadas em *Machine Learning* (ML) as quais podem ser empregadas como IDSs (Khanam *et al.*, 2020). Priya et al., (2021) destacam a eficácia dos métodos baseados em aprendizado de máquina na detecção de eventos anômalos no fluxo de tráfego de rede. Para identificar e classificar o tráfego malicioso, Mahajan et al. (2022) reforçam a importância de incorporar estratégias de *Machine Learning* e *Deep Learning*. Essas abordagens corroboram a necessidade de utilizar técnicas avançadas de ML para lidar com ameaças e garantir uma detecção mais precisa e eficiente de atividades maliciosas na rede.

Sistemas de Detecção de Intrusão (IDS - *Intrusion Detection System*) baseados em anomalias utilizando modelos de ML, apresentam vantagens. Entre elas, destacam-se o tamanho fixo do modelo após a implementação e a flexibilidade de execução, que pode ocorrer tanto em servidores, em conjunto com *middlewares* operando em *hardware* com menos restrições, quanto em alguns dispositivos baseados em microcontroladores, mesmo com as limitações mencionadas.

Na seção 3.4, serão discutidos trabalhos relacionados que exploram abordagens distintas e variadas perspectivas sobre o tema. Esses estudos propõem soluções baseadas no desenvolvimento e uso de modelos de ML aplicados a IDS.

No Quadro 4 é apresentado um resumo das principais características dos protocolos listados anteriormente, permitindo uma comparação entre eles. É importante notar que cada protocolo foi originalmente desenvolvido para propósitos específicos, mas podem ser utilizados em aplicações além de suas funções primárias. Por exemplo, o HTTP, que foi concebido para transporte de hipertextos na web, é usado na troca de mensagens entre um servidor e um dispositivo IoT. Assim, a escolha do protocolo mais adequado para uma determinada aplicação fica a critério do desenvolvedor.

Todos os protocolos, conforme ilustrado no Quadro 4, empregam mecanismos de criptografia, como o TLS e DTLS, em uma subcamada de segurança na camada de aplicação para assegurar a privacidade dos dados. No entanto, esses mecanismos não conseguem prevenir ataques que não têm como objetivo a obtenção dos dados, tais como os ataques *Deny of Service* (DoS) ou *Distributed Denial of Service* (DDoS). Adicionalmente, as versões desses protocolos de segurança podem conter vulnerabilidades ainda não descobertas ou falhas em suas implementações.

Vale mencionar também que os protocolos TLS e DTLS foram amplamente adotados pelos protocolos IoT porém, é importante ressaltar que esses protocolos não foram especificamente projetados para atender às demandas das aplicações de IoT (Tiburski *et al.*, 2017).

Quadro 4 Principais características dos protocolos de aplicação IoT.

Protocolo	**Padrão**	**Modelo**			**Requer servidor**	**Comunicação Síncrona**	**QoS**	**Serviço de descoberta**	**Cabeçalho (bytes)**	**Segurança**	**Protocol de transporte**			**Escopo**			**Propósito ou tipo de aplicação**
		Pub/Sub	**Req/Res**	**Outros**							**TCP**	**UDP**	**Outros**	**D2D**[8]	**D2C**[9]	**C2C**[10]	
AMQP	OASIS	✓	✓	P2P	✓	✓	✓	✗	8	TLS e SASL	✓	✗		✓	✓	✓	Aplicações comerciais e bancárias

[8] D2D – *Device to device*
[9] D2C – *Device to computer*
[10] C2C – *Computer to computer*

Protocolo	Padrão	Modelo			Requer servidor	Comunicação Síncrona	QoS	Serviço de descoberta	Cabeçalho (bytes)	Segurança	Protocol de transporte			Escopo			Propósito ou tipo de aplicação
		Pub/Sub	Req/Res	Outros							TCP	UDP	Outros	D2D[8]	D2C[9]	C2C[10]	
CoAP	RFC 7252	✗	✓	REST	✓	✓	✓	✓	4	DTLS	✗	✓		✓	✗	✗	Aplicações baseadas em dispositivos com restrições
DDS	OMG	✓	✓		✗	✓	✓	✓	20	TLS, DTLS e DDS Sec	✓	✓		✓	✓	✓	Automação Industrial
HTTP/ 1.1	RFC 2616	✗	✓		✓	✓	✓	✗	40	TLS	✓	✗			✓	✓	Navegação na web
HTTP/ 2.0	RFC 7540	✗	✓		✓	✓	✓	✗	-	TLS	✓	✗		✓	✓	✓	Navegação na web
HTTP/ 3.0	RFC 9114	✗	✓		✓	✓	✓	✗	-	QUIC	✗	✓		✓	✓	✓	Navegação na web
Matter	CSA	✓	✓	Mesh	✗	✓	✓	✓	100	AEAD	✓	✓	BTP	✓	✓	✗	Casa Inteligente
MQTT	OASIS	✓	✗		✓	✗	✓	✗	2	TLS	✓	✗		✗	✓	✗	Comunicação M2M[11] leve
XMPP	RFC 6120	✗	✓		✓	✗	✓	✓	100	TLS/ SASL	✓	✗	BOSH	✗	✓	✓	Troca de mensagens, jogos, VoIP, videochamadas e arquivos.

Fontes: (Object Management Group, 2015, 2022; Dhas e Jeyanthi, 2019; Swamy e Kota, 2020; Connectivity Standards Alliance, 2022)

[11] M2M – *Machine to machine*

3. IoT-IDS

Os sistemas IoT, como discutido na Seção 2.2, enfrentam desafios de segurança devido à ausência de estruturas tradicionais de proteção, como softwares antivírus e firewalls, ou mesmo impossibilidade de implementá-los por restrições computacionais. Esses desafios são agravados pelas limitações inerentes aos dispositivos IoT, que possuem restrições em termos de memória, capacidade de processamento e, em alguns casos, fornecimento de energia, limitando o tipo e volume de processamento que podem realizar. Para enfrentar esses desafios, qualquer abordagem de segurança destinada a dispositivos IoT precisa consumir poucos recursos computacionais, além de ser adaptável e escalável, considerando a ampla diversidade de dispositivos e protocolos de comunicação para a IoT. Além disso, dada a sensibilidade dos dados envolvidos, é fundamental que a resposta a atividades maliciosas ocorra em tempo real ou com baixa latência (Khanam *et al.*, 2020; Araya e Rifà-Pous, 2023).

Entre as principais medidas de mitigação de ataques, a utilização de modelos de *Machine Learning* (ML) para detecção de anomalias pode oferecer vantagens significativas em relação aos demais tipos de solução considerando as restrições dos dispositivos IoT. Esses modelos, uma vez implementados, têm um tamanho fixo e podem ser executados tanto em servidores robustos quanto em dispositivos baseados em microcontroladores com recursos limitados, proporcionando flexibilidade e eficácia na proteção de sistemas IoT, quando aplicada técnicas de otimização dos modelos ML para dispositivos restritos (Bilal, 2017; Butun, Osterberg e Song, 2020; Chauhan, Singh e Jain, 2020; Khanam *et al.*, 2020; Krishna *et al.*, 2021; Bhandari *et al.*, 2023).

3.1. Abordagem

A abordagem proposta como solução IDS para ambientes IoT é para desenvolvimento de um NDIS (*Network IDS*) centralizado baseado em modelos de aprendizado de máquina. Nesta abordagem é utilizado um conjunto de processos interligados para monitorar, analisar e classificar o tráfego de rede em sistemas de Internet das Coisas (IoT), com o objetivo de detectar intrusões e anomalias. A abordagem é estruturada em quatro etapas principais: captura de dados, extração de características, classificação de tráfego e ação de mitigação.

Na captura de dados o IDS deve iniciar a captura contínua de pacotes de dados que trafegam na rede. Esses dados brutos são coletados em tempo real, fornecendo todo o tráfego

de rede que passa pelo adaptador de rede do *host* onde o IDS é executado. Por isso o *host* deve funcionar como um *gateway* para o fluxo de comunicação dos dispositivos IoT.

Na extração de características o sistema processa características relevantes que descrevem o estado atual da rede. Essas características incluem métricas e estatísticas que ajudam a representar o comportamento do tráfego de rede através de diversos registros que relacionam nós de origem e destino. A extração acontece de forma que as estatísticas possam ser usadas para identificar padrões, independente dos nós participantes da rede.

Utilizando as características processadas, o sistema emprega um modelo de aprendizado de máquina para classificar o tráfego de rede. As características extraídas devem formar um conjunto de dados que permitam o modelo de ML identificar padrões de comportamento normal e detectar anomalias ou intrusões. A quantidade de classes detectadas, relativas aos tipos de ataque, vão depender do conjunto de dados fornecido para o modelo de ML.

As saídas do IDS podem ser de natureza passiva, ou seja, todos os resultados da classificação do tráfego são completamente registrados em log e quando uma anomalia é identificada, o pacote classificado como anômalo tem informações, como tipo, origem e destino, registradas e exibidas em console. Além disso, essas informações podem ser utilizadas para a execução de uma ação com saída de natureza ativa específica para o tipo anomalia, como o bloqueio de um host ou porta, dependendo da infraestrutura disponível e da escolha do desenvolvedor.

Desta forma o IDS deve integrar todas as etapas de forma coordenada, garantindo que a captura de dados, a extração de características e a classificação aconteçam de maneira contínua e em tempo real. Isso permite uma detecção eficiente e imediata de possíveis ameaças, reduzindo o tempo de resposta a incidentes de segurança, desde que o IDS esteja em execução junto ao nó central da rede, pois trata-se de um NDIS.

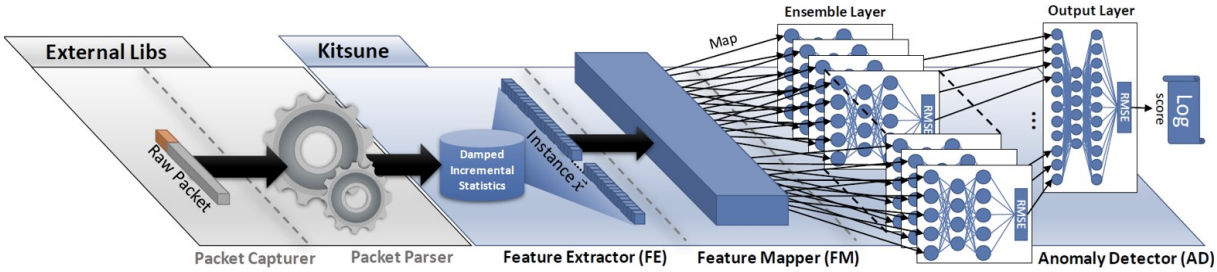
Com esta abordagem é possível construir um IDS voltado para desenvolvedores que precisam de uma solução para agregar um nível de segurança a uma infraestrutura de IoT, e por isso tem características flexíveis que podem ser ajustadas de acordo com as necessidades ou limitações do ambiente. A implementação do IoT-IDS, seguindo esta abordagem, será apresentada em detalhes nas próximas seções.

3.2. Arquitetura

A principal referência e inspiração para este trabalho é o ***Kitsune,*** descrito por Mirsky et al. (2018) como "um sistema de detecção de intrusões de rede (NIDS) *plug-and-play* que pode aprender a detectar ataques na rede local de forma não supervisionada e eficiente em tempo real", porém na versão disponível no repositório mencionado no artigo, limita-se a análise de arquivos de captura. Seu método de extração de características rastreia padrões em todos os canais da rede de forma eficiente. Porém o Kitsune limita-se a indicar se há ou não alguma anomalia em pacotes de rede, não realizando a classificação do tipo de ataque.

Na Figura 9 é ilustrada a sequência de processamento dos pacotes de rede na arquitetura do Kitsune onde o principal ponto de interesse desta dissertação é o *Feature Extrator (FE).* Ele mantém estatísticas incrementais e atualiza as características de forma dinâmica, permitindo que o sistema detecte anomalias sem a necessidade de uma análise manual extensiva dos dados.

Figura 9 Ilustração da Arquitetura do Kitsune.



Fonte: (Mirsky *et al.*, 2018)

A classe *netStat* (usada no FE) é responsável por manter as Estatísticas Incrementais Atenuadas (*Damped Incremental Statistics)*, ilustradas no cilindro da Figura 9. Estas estatísticas permitem que o sistema use pesos para atenuar os valores mais antigos, em vez de usar uma janela deslizante, que tem alta complexidade de memória e tempo de execução. Assim são mantidos os dados calculados dos últimos cinco pacotes e com os seus respectivos pesos (Mirsky *et al.*, 2018).

A saída do *FeatureExtractor* é uma lista com cem valores numéricos descritos e listados no Quadro 5. Na primeira coluna do Quadro 5, além do nome da estatística, são listadas as quantidades de valores relacionados a elas conforme os níveis de granularidade temporal dos pacotes capturados. As estatísticas são extraídas em cinco janelas de tempo diferentes: 100ms, 500ms, 1.5 segundos, 10 segundos e 1 minuto com atenuação de λ = 5, 3, 1, 0.1, 0.01,

respectivamente. Para *MIstat* e *HHstat*_jit, em cada janela de tempo, são calculados três tipos de estatísticas: peso (w), média (μ) e desvio padrão (σ), ou seja, 15 valores de cada. Já para *HHstat* e *HpHpstat*, além das três estatísticas básicas (w, μ, σ) calculadas para cada uma das cinco janelas de tempo, também são incluídas estatísticas adicionais para capturar relações bidimensionais (em pares) entre os fluxos, como magnitude, raio, covariância e coeficiente de correlação, ou seja, 35valores de cada (Mirsky *et al.*, 2018).

Quadro 5 Descrição das Estatísticas Incrementais Atenuadas calculadas.

Nome (Quant.)	Relação	Dados	Descrição
MIstat (15)	MAC-IP	**SrcMAC-IP:** • Tamanho do Pacote: Média, Desvio Padrão • Taxa de Pacotes: Peso	Estatísticas temporais do tráfego de rede que se originam de um endereço MAC e IP específicos (SrcMAC-IP)
HHstat (35)	Host-Host	**SrcIP:** • Tamanho do Pacote: Média, Desvio Padrão • Taxa de Pacotes: Peso	Estatísticas incrementais associadas ao tráfego entre dois endereços IP específicos. "HH" indica a relação "Host-to-Host", ou seja, as estatísticas são calculadas para capturar o comportamento do tráfego entre dois hosts na rede.
HHstat_jit (15)	Jitter Host-Host	**Canal (origem e destino)**: • Tamanho do Pacote: Média, Desvio Padrão • Taxa de Pacotes: Peso • Jitter: Peso, Média, Desvio Padrão	Estatísticas incrementais associadas ao "*jitter*" (variação no atraso de pacotes) no tráfego entre dois endereços IP específicos. Esse tipo de estatística é importante para detectar anomalias relacionadas à consistência e qualidade da comunicação entre hosts.
HpHpstat (35)	Host-Host por Porta	**Socket**: • Tamanho do Pacote: Média, Desvio Padrão • Taxa de Pacotes: Peso	Estatísticas incrementais associadas ao tráfego entre pares de sockets de rede específicos. "HpHp" indica a relação "Host-port to Host-port", ou seja, as estatísticas são calculadas para capturar o comportamento do tráfego entre sockets de rede específicos nos hosts.

Fonte: (Mirsky *et al.*, 2018)

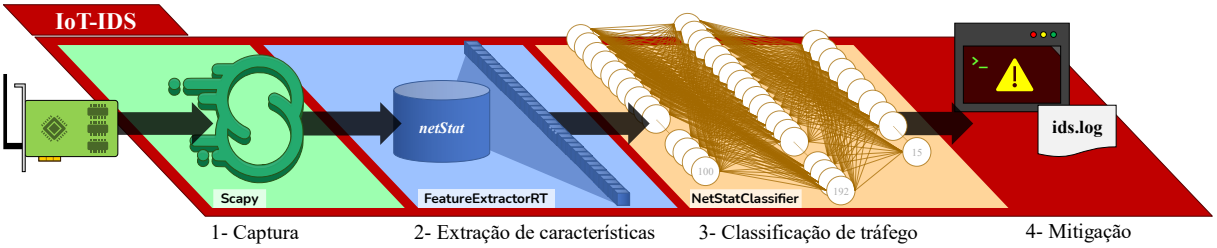
A abordagem do Kitsune só permite a verificação de anomalias por meio de arquivos nos formatos '.pcap' ou '.pcapng', os quais podem ser gerados por meio da biblioteca *Scapy* ou através da aplicação *Tshark*. Dessa forma, o Kitsune não consegue realizar detecção em tempo real, uma vez que apresenta uma grande latência e necessidade de gravar arquivos de captura em meios de armazenamento.

Assim, a arquitetura proposta neste trabalho combina a extração e manutenção de estatísticas incrementais atenuadas, baseada no *FeatureExtractor* (FE) do Kitsune, para a implementação de um IDS que além de detectar anormalidades ou ataques em tempo real, mas também identifica a natureza específica do ataque, ou seja, faz sua classificação. Para detecção

de intrusão em tempo real, a arquitetura proposta adaptou o FE para lidar com pacotes capturados online, além da verificação em arquivos, o qual foi nomeado como *FeatureExtractorRT*.

Na Figura 10 é ilustrada a arquitetura do IoT-IDS implementado com as ações dividida em 4 passos, conforme a abordagem proposta. Esta implementação utiliza-se de duas threads com finalidade da execução da captura e o processamento dos pacotes para garantir que nenhum pacote deixe de ser analisado pelo IDS e que os intervalos de tempo necessários para os cálculos das estatísticas sejam cumpridos.

Figura 10 Ilustração da Arquitetura do IoT-IDS.



Fonte: Adaptado de Mirsky et al. (2018).

No passo 1 a função *sniff()*[12], da biblioteca *Scapy,* é utilizada na primeira *thread* para capturar os pacotes de rede em tempo real e retorna-os como objetos *Scapy*, que são representações estruturadas de pacotes de rede. Esses objetos contêm todas as informações dos pacotes capturados, como cabeçalhos, *payload* e outros detalhes de protocolo. Esse formato é adequado à entrada do *FeatureExtractorRT*.

No passo 2, são extraídos diversos dados do pacote de rede e são encaminhados ao *netStat* que calcula as estatísticas incrementais atenuadas - explicadas anteriormente no Quadro 5 - e mantém em memória os dados dos últimos cinco pacotes e com os seus respectivos pesos. No Quadro 6 são listados e descritos os dados extraídos de cada pacote de rede.

Quadro 6 Dados extraídos de um pacote de rede pelo FeatureExtractorRT

Campo de origem	**Descrição**
packet.time	Momento em que o pacote foi capturado.
len(packet) - do script	Tamanho total do pacote em bytes.
IPtype - do script	Indica se o pacote usa IPv4 (0) ou IPv6 (1).

[12] BIONDI, Philippe. Usage - Scapy 2.6.0 documentation. 2024. Disponível em: https://scapy.readthedocs.io/en/latest/usage.html#sniffing. Acesso em: 10 nov. 2023.

Campo de origem	Descrição
packet[IP].src ou packet[IPv6].src	Endereço IP da fonte do pacote.
packet[IP].dst ou packet[IPv6].dst	Endereço IP de destino do pacote.
packet[TCP].sport ou packet[UDP].sport	Número da porta de origem se o pacote for TCP ou UDP.
packet[TCP].dport ou packet[UDP].dport	Número da porta de destino se o pacote for TCP ou UDP.
packet.src	Endereço MAC de origem do pacote.
packet.dst	Endereço MAC de destino do pacote.
packet[ARP].psrc ou packet[ARP].pdst ou 'icmp'	Se o pacote for ARP ou ICMP, extrai os respectivos campos.
packet.src e packet.dst	Se não houver informações de protocolo e IP, retorna os endereços MAC.

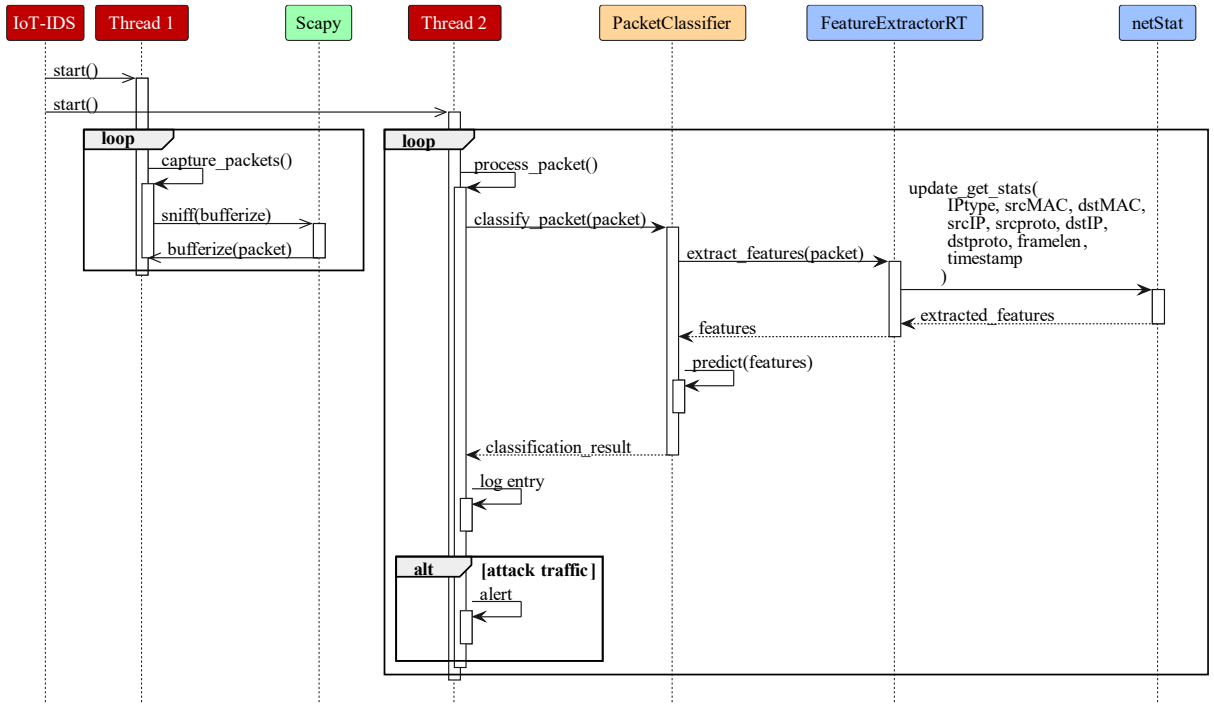
Fonte: Autoria própria.

No passo 3 o *NetStatClassifier* é o modelo de ML que recebe um vetor com cem valores de dados estatísticos como entrada e os classifica. O modelo foi treinado com quinze classes sendo "*Normal traffic*" ou como um dos quatorze tipos de ataque detalhados na subseção 3.3.

Por último, no passo 4, o resultado do classificador sempre é registrado em um arquivo de log e, em caso de detecção de ataques, gera alertas no terminal. Os mesmos dados que são exibidos nos alertas podem ser utilizados para realizar ações de mitigação ou tratamento de escolha do desenvolvedor da aplicação que estiver usando o IoT-IDS.

O diagrama de sequência da Figura 11 registra a ordem das funções e métodos executados e seus retornos para realizar os 4 passos mencionados. Vale notar que o IoT-IDS usa duas *threads* que compartilham as variáveis e objetos globais e funcionam de forma contínua. Na *Thread 1*, o IoT-IDS utiliza a função *sniff()*, que chama *bufferize()* para cada pacote capturado, colocando-os em uma fila. A *Thread 2* remove um pacote da fila e o fornece como argumento a uma instância da classe *PacketClassifier* por meio do método *classify_packet()*. Este objeto inclui encapsulados o modelo *NetStatClassifier*, o *FeatureExtractorRT* e o *netStat*, responsáveis pelo cálculo e manutenção das estatísticas de tráfego. Com as estatísticas atualizadas o método *predict()* do modelo é chamado e sua saída é registrada em log e, caso seja classificada como algum tipo de ataque, um aleta é gerado.

Figura 11 Diagrama de Sequência representando a interação entre os principais componentes do IoT-IDS



Fonte: Autoria própria.

O IoT-IDS pode ser ajustado em dois parâmetros para obter maior estabilidade. O valor do parâmetro *window* define a quantidade de pacotes que será utilizada para calcular as estatísticas de rede antes de iniciar a classificação dos pacotes. Um maior valor permite que a formação das estatísticas seja estabilizada, pois alguns dos valores calculados podem ser discrepantes com poucos pacotes. O segundo parâmetro é o tamanho máximo do *buffer* que define um limite de pacotes que podem ser enfileirados. Um limite maior permite que mais pacotes sejam armazenados temporariamente enquanto aguardam processamento, o que pode ser útil em cenários de alto tráfego para evitar a perda de pacotes, porém, um valor muito grande pode aumentar o uso de memória.

3.3. NetStatClassifier

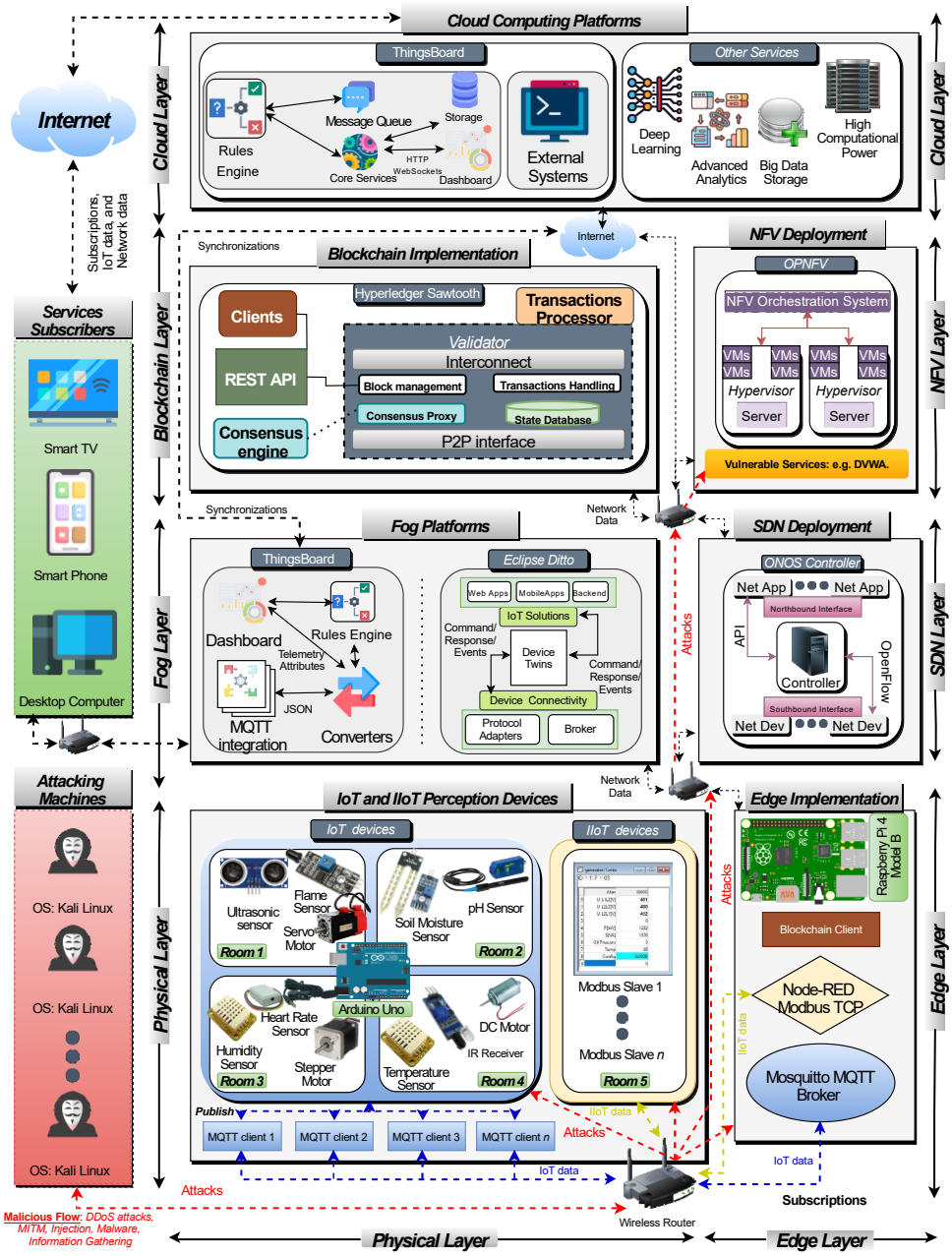
As subseções a seguir descrevem o processo de desenvolvimento do *NetStatClassifier*, que é um modelo de ML capaz de classificar pacotes de rede em até quinze classes diferentes.

3.3.1. Conjunto de dados

Para a criação do modelo foi utilizado o *Edge-IIoTset* que é um conjunto de dados abrangente e realista de segurança cibernética para aplicações de IoT e IIoT, proposto por Ferrag et al. (2022). O *dataset* foi gerado usando um *testbed* IoT/IIoT que inclui uma vasta gama de dispositivos, sensores, protocolos e configurações de nuvem e borda. O *testbed* usado

inclui mais de dez tipos de dispositivos IoT além da infraestrutura de rede incluído servidores e serviços virtualizados, como ilustrados na Figura 12.

Figura 12 Arquitetura do ambiente de testes do EdgeIIoTset.

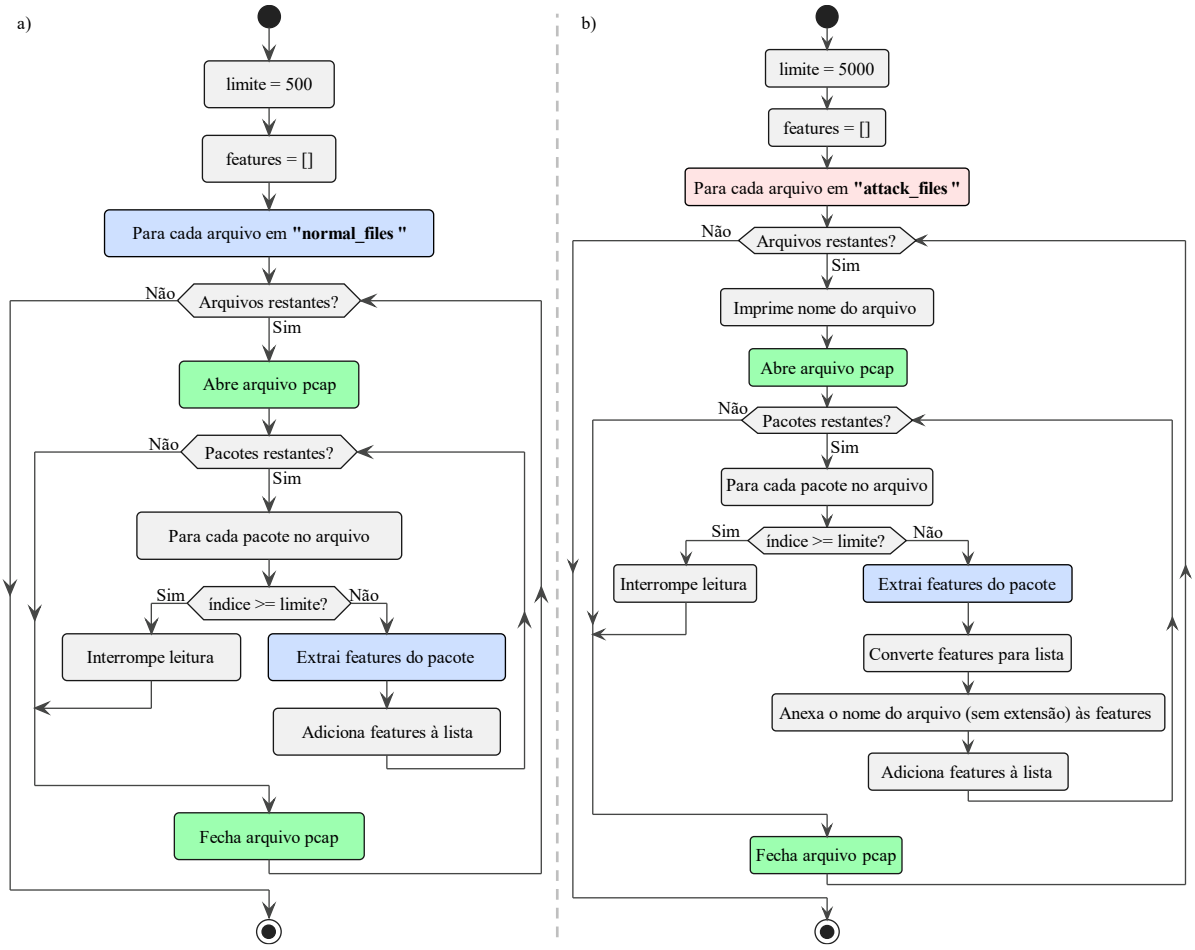


Fonte: (Ferrag et al., 2022)

Dentre outros conjuntos de dados o IoT o Edge-IIoTset se destaca pela infraestrutura empregada e pela quantidade de ataques realizados. O *dataset* possui quinze classes: *Normal traffic, Backdoor Attack, DDoS HTTP Flood, DDoS ICMP Flood, DDoS TCP SYN Flood, DDoS UDP Flood, Password Attack, Port Scanning, Ransomware, SQL Injection, Uploading Attack, Vulnerability Scanner, XSS Attack, MITM ARP Spoofing DNS* e *OS Fingerprinting.*

Porém, não foi possível utilizar os dados rotulados do *dataset* Edge-IIoTset, pois os campos não estavam de acordo com a entrada do *netStat*, impossibilitando a geração das estatísticas de tráfego de rede. Contudo, este *dataset* também fornece os dados brutos, exatamente como foram capturados, em formato *pcap* e organizados de forma a identificar as classes a que pertencem. Assim, foi possível aproveitar os dados para a criação de um novo *dataset*, utilizando o *netStat* para gerar as estatísticas de tráfego de rede, e por isso foi chamado de *EdgeIIoTset_netStat*. A Figura 13 apresenta os algoritmos utilizados para gerar o novo *dataset*. Nesta figura, é possível observar a geração da classe "tráfego normal" (a), a qual obtém informações dos quinhentos primeiros pacotes a partir de dez arquivos diferentes do *dataset* original. Já para as demais classes o algoritmo foi limitado a cinco mil pacotes por arquivo, uma vez que cada arquiva representa um tipo de ataque. Com isso, cada classe apresenta informações de, no máximo, cinco mil pacotes. Após a adição dos rótulos e junção arquivos obtém-se o novo *dataset*.

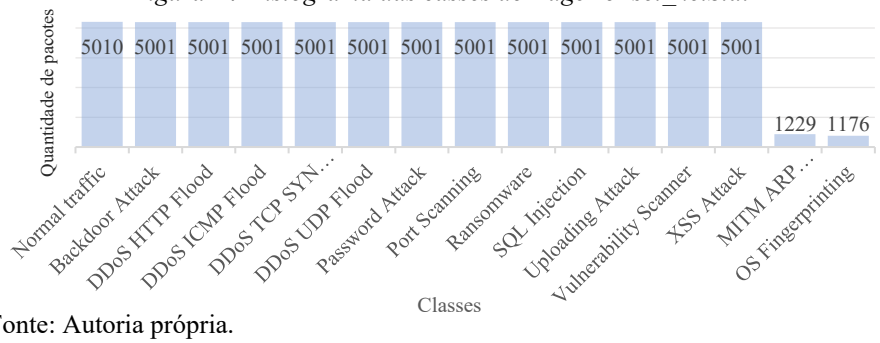
Figura 13 Algoritmos para gerar estatísticas de trafego de rede a partir de arquivo '.pcap'.



Fonte: Autoria própria.

O novo *dataset* apresenta o balanceamento de dados relativamente equilibrado, com exceção das classes *MitM Arp Spoofing* e *OS Fingerprinting*. No histograma da Figura 14 são revelados os quantitativos de dados para cada classe.

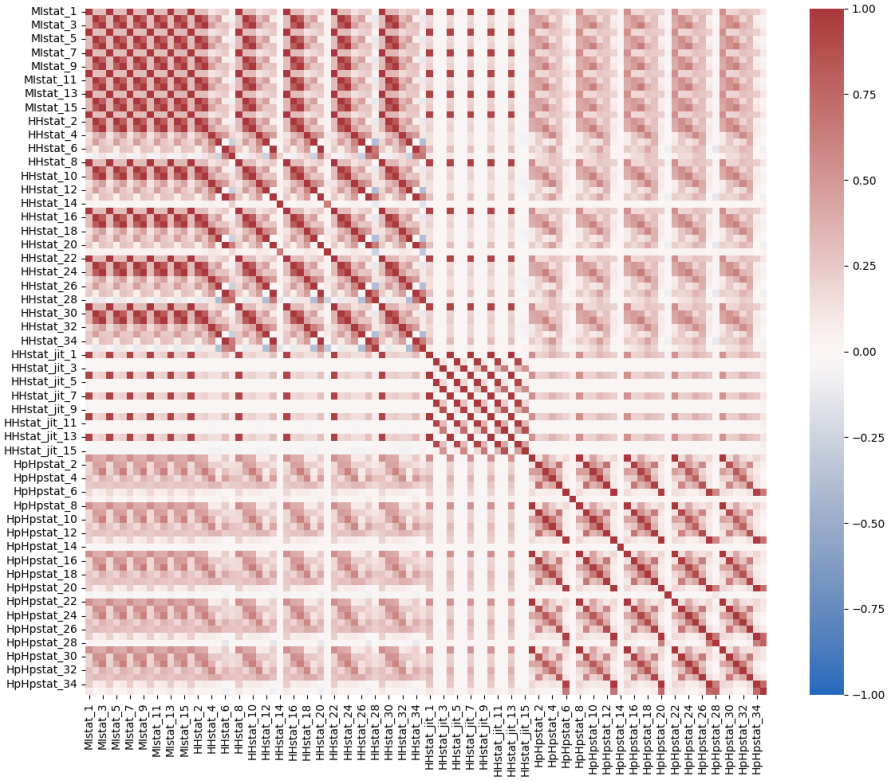
Figura 14 Histograma das casses do EdgeIIoTset_netStat



Fonte: Autoria própria.

No contexto do aprendizado de máquina, a correlação de Pearson é essencial para a análise inicial dos dados de treinamento, auxiliando na identificação de relações lineares entre variáveis independentes e a variável alvo, ou entre as próprias variáveis independentes. Este coeficiente, que varia entre -1 e 1, quantifica a força e a direção de uma relação linear entre duas variáveis (Cai *et al.*, 2018). A Figura 15 apresenta uma matriz de correlação calculada para o conjunto de dados *EdgeIIoT_netStat*. Os atributos estão nomeados como no Quadro 5.

Figura 15 Matriz de correlação dos atributos do EdgeIIoT_netStat.



Fonte: Autoria própria.

Na matriz de correlação da Figura 15, observa-se que as variáveis de um mesmo grupo, como MIstat e HHstat, tendem a ter uma correlação muito forte entre si, o que é visível pelos valores elevados ao longo da diagonal e nas proximidades desta. Isso sugere que estatísticas como média, desvio padrão e peso são bastante interrelacionadas quando calculadas para diferentes janelas de tempo do mesmo tipo de relação, seja entre MAC e IP ou entre hosts. Além disso, é possível observar que as estatísticas de *jitter* (como em HHstat_jit) apresentam correlações importantes que indicam consistência no comportamento do atraso na comunicação. Blocos separados na matriz refletem grupos que estão mais correlacionados internamente, sugerindo que cada conjunto captura uma dimensão específica do comportamento do tráfego.

Pode-se concluir que as correlações altas observadas entre as variáveis do mesmo grupo sugerem uma possível redundância, indicando que algumas das variáveis podem ser removidas sem comprometer significativamente a informação total dos dados. Isso é útil em um contexto de redução de dimensionalidade para facilitar a interpretação e o treinamento de modelos de aprendizado de máquina. Além disso, análises futuras devem focar na variação das correlações ao longo do tempo para identificar comportamentos anômalos, como ataques de negação de serviço (DDoS) que alteram significativamente o jitter ou a taxa de pacotes. As estatísticas bidimensionais, como covariância e coeficiente de correlação entre pares de sockets (HpHpstat), têm papel crucial na captura de relações que são essenciais para detectar comportamentos anômalos de maneira mais eficiente.

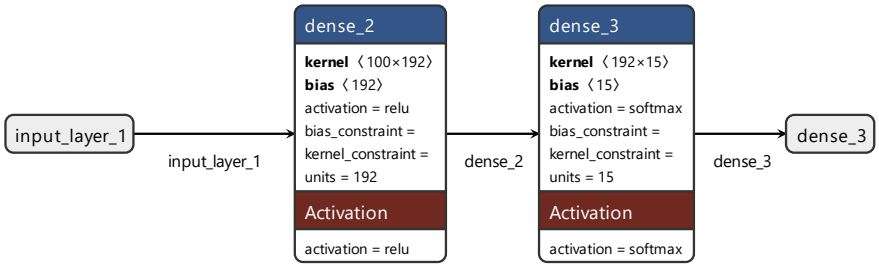
3.3.2. Preparação dos Dados

Nesta etapa, os dados do *EdgeIIoT_netStat* foram divididos em três conjuntos: treino, validação e teste, com 80%, 10% e 10% dos dados, respectivamente. No processo de treinamento o modelo é ajustado com o conjunto de treino, avaliado e ajustado continuamente com o conjunto de validação. Isso garante que o modelo nunca tenha acesso aos dados de teste durante o treinamento, preservando a integridade da avaliação final. A divisão foi realizada utilizando a função *train_test_split()* do *Scikit-learn*, com estratificação para manter as proporções das classes em cada subconjunto. Além disso, foram separadas as características dos dados (*X_train*, *X_val* e *X_test*) dos rótulos (*y_train*, *y_val* e *y_test*), que identificam a classe do tráfego. Um *LabelEncoder* foi usado para converter as classes de texto em valores numéricos, facilitando o processamento pelos modelos de aprendizado de máquina.

3.3.3. Otimização de Hiperparâmetros

Para determinar a configuração ideal do modelo, foi utilizada a biblioteca

Figura 16 Diagrama de arquitetura do modelo netStat_classifier



Fonte: Gerado a partir do modelo com a ferramenta *Netron*.

KerasTuner[13] para realizar uma busca em uma ampla gama de hiperparâmetros, incluindo o número de camadas ocultas, o número de unidades em cada camada e a taxa de aprendizado. Especificamente, foi definido um intervalo para o número de camadas entre 1 e 5, com unidades variando entre 32 e 512, e testadas diferentes taxas de aprendizado. Com o objetivo de maximizar a acurácia de validação, utilizando *"val_accuracy"* como métrica principal. Após cinco tentativas, o melhor modelo foi identificado com uma única camada oculta composta por 192 unidades e uma taxa de aprendizado de 0,001, alcançando uma acurácia de validação de 94,69%. Esta configuração resultou em um modelo com 22.287 parâmetros treináveis.

3.3.4. Construção e Treinamento

O diagrama da Figura 16 representa a rede neural sequencial *netStat_classifier* composta por três camadas principais, sendo as duas últimas completamente conectadas. Este modelo consiste em uma rede neural com três camada, sendo uma camada de entrada, seguida por uma camada densa oculta e mais uma camada densa de saída formadas, respectivamente de 100, 192 e 15 neurônios. A primeira camada densa usa a função de ativação *ReLU* e a segunda usa a *Softmax*, que são popularmente associadas aos modelos de classificação (Cai *et al.*, 2018). O que é exatamente o caso, pois modelo foi projetado para um problema de classificação com 15 classes. O modelo foi construído com o uso de bibliotecas do *TensorFlow*.

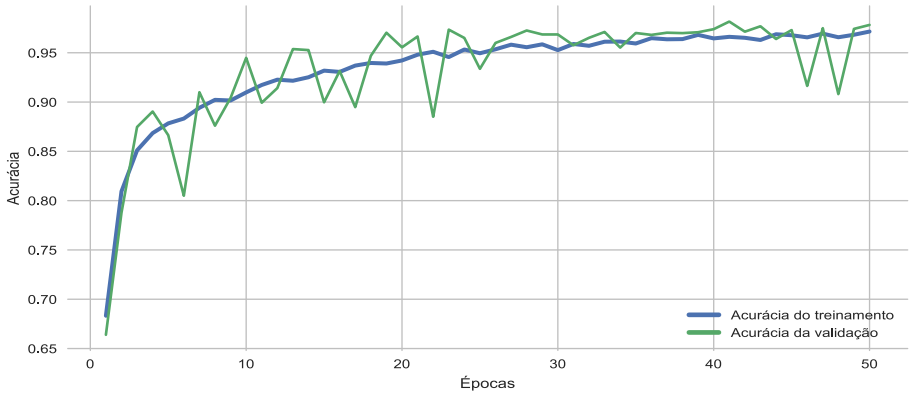
[13] https://keras.io/keras_tuner/

O treinamento foi monitorado por uma *callback* do tipo *EarlyStopping* usada para interromper o processo em caso de piora na acurácia após cinco de épocas de treinamento consecutivas sem melhoria. Uma época refere-se a uma iteração completa sobre todo o conjunto de dados de treinamento. Na Figura 17 são apresentados os resultados do treinamento com os melhores resultados obtidos. Foram realizadas 50 épocas de treinamento resultando em 97,27% de acurácia com os dados de treinamento e 97,82% com os dados de validação.

3.3.5. Avaliação do Modelo

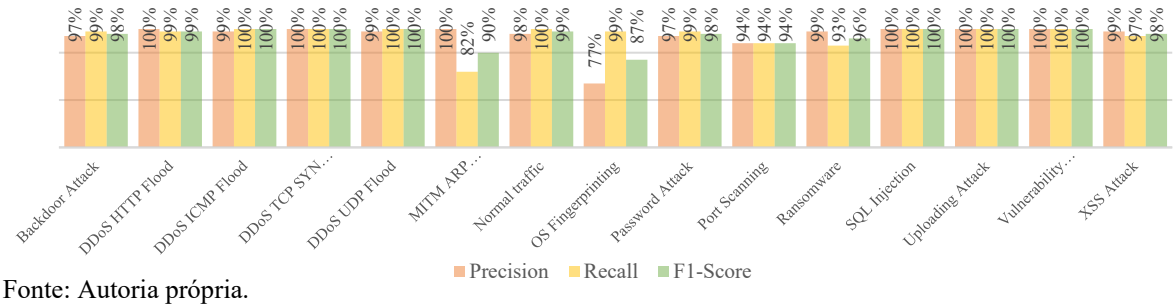
Para realizar a avaliação do modelo, a terceira fração dos dados (*X_test*) foi utilizada com o método *predict()* do modelo, e os resultados previstos foram comparados com os rótulos originais do conjunto de dados (*y_test*) utilizando as funções *classification_report* e *confusion_matrix* da biblioteca *Scikit-learn*[14].

Figura 17 Acurácia com os dados de treinamento e validação



Fonte: Autoria própria.

Figura 18 Relatório de classificação



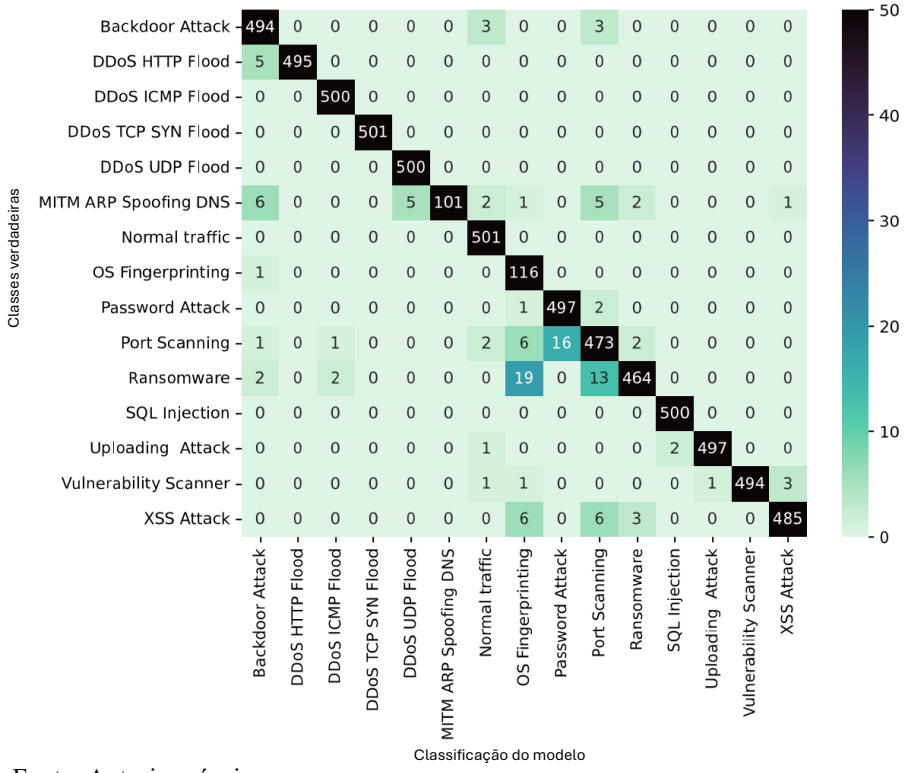
Fonte: Autoria própria.

[14] https://scikit-learn.org/stable/api/sklearn.metrics.html

Os resultados do *classification_report* estão apresentados no gráfico da Figura 18. A média macro obtida foi de 97%, enquanto a média ponderada atingiu 98%, ambas referentes às métricas de Precision, Recall e F1-Score. A acurácia alcançada pelo modelo foi de 98%. No entanto, o modelo tem alguma dificuldade com o "OS Fingerprinting", com uma precisão relativamente baixa, exatamente as duas classes com menor quantidade de elementos no *dataset*.

Na matriz de confusão, da Figura 19, é registrada a distribuição dos erros e acertos do modelo na classificação de diferentes tipos de tráfego e ataques de rede. Cada linha representa as instâncias verdadeiras de uma classe, enquanto cada coluna representa as instâncias que o

Figura 19 Matriz de confusão gerada com os dados de teste



Fonte: Autoria própria.

modelo previu como pertencentes àquela classe. A maioria das instâncias estão concentradas na diagonal principal, indicando que o modelo classifica corretamente a maioria dos exemplos em suas respectivas classes.

Com relação a classe "*MITM ARP Spoofing DNS*" que possui 123 pacotes verdadeiros ao todo, 101 foram classificados corretamente, mas 22 foram classificados de forma errada como falsos positivos para outras classes. Exatamente como apontado pelo relatório de classificação, onde o *recall* dessa classe foi relativamente baixo, indicando que o modelo tem

dificuldade em identificar corretamente todos os exemplos dessa classe. De forma semelhante, a classe "*OS Fingerprinting*", porém agora classificando 35 pacotes como falsos positivos. Novamente alinhado com a *precision* mais baixa observada para esta classe, indicando uma confusão entre "*OS Fingerprinting*" e outras classes de tráfego malicioso ou normal. Algumas instâncias de "*Ransomware*" foram classificadas como "*Port Scanning*" (16 instâncias), e vice-versa (13 instâncias). Isso pode sugerir que esses dois tipos de tráfego apresentam características semelhantes, confundindo o modelo.

A matriz de confusão complementa a análise do relatório de classificação, destacando onde o modelo tem dificuldades e onde ele é mais eficaz. A principal fraqueza do modelo é na identificação de "*MITM ARP Spoofing DNS*", "*OS Fingerprinting*", e na distinção entre "*Port Scanning"* e "*Ransomware*". No caso dos dois primeiros, é bastante forte a suspeita de que o desbalanceamento dos dados pode ter provocado o problema. A maior parte das outras classes é identificada, com poucos erros, confirmando desempenho médio do modelo com 97% de acurácia.

3.4. Trabalhos relacionados

Nesta seção, serão apresentados, em ordem cronológica de publicação, alguns trabalhos que oferecem abordagens distintas e perspectivas variadas. Esses trabalhos propõem soluções que envolvem a desenvolvimento de modelos de ML e seu uso em IDS, os quais puderam ser estudados, adaptados e/ou utilizados no desenvolvimento do IoT-IDS.

Shukla, (2017) apresentou três Sistemas de Detecção de Intrusão (IDS) baseados em aprendizado de máquina para redes IoT utilizando o protocolo RPL: *K-means* (KM-IDS), *Decision Tree* (DT-IDS), e um modelo híbrido. O KM-IDS, não supervisionado, agrupa nós em "zonas seguras" e sinaliza conexões entre zonas diferentes como ataques, enquanto o DT-IDS, supervisionado, aprende um "limiar de distância segura" e sinaliza conexões que excedem esse limiar como ataques. O *Hybrid-IDS* combina ambos, com o KM-IDS identificando candidatos a ataques e o DT-IDS refinando a detecção. Experimentos em topologias de rede diversas mostraram que o KM-IDS tem uma taxa de detecção de 70 a 93%, mas com mais falsos positivos, já o DT-IDS detecta de 71 a 80% dos ataques com menos falsos positivos. O Hybrid-IDS, apesar de uma ligeira redução na taxa de detecção (71 a 75%), reduz significativamente os falsos positivos. A tolerância a falsos positivos e da precisão necessária para detectar ataques *Wormhole*, com a limitação de que o estudo foca apenas nesse tipo de ataque.

Mirsky et al. (2018) apresentam Kitsune, um sistema de detecção de intrusões de rede (NIDS) eficiente, baseado em um conjunto de *autoencoders*, para dispositivos de rede. O Kitsune é um sistema *plug-and-play* que não requer supervisão para detectar ataques na rede local. Seu algoritmo central utiliza um conjunto de *autoencoders* que coletivamente diferenciam padrões de tráfego normal e anômalo. Segundo os autores, uma vantagem significativa do uso de *autoencoders* é que eles não necessitam de treinamento com dados rotulados, permitindo que o NIDS se adapte automaticamente a diferentes redes sem a necessidade de intervenção do usuário. As avaliações apresentadas demonstram que o Kitsune pode detectar diversos ataques com desempenho comparável a detectores de anomalias offline, mesmo em dispositivos de *hardware* simples.

Holbrook e Alamaniotis (2019) compararam a eficácia de três algoritmos de aprendizado de máquina - *Support Vector Machines* (SVM), *Random Forest*, e *Deep Neural Network* (DNN) - na detecção de anomalias em redes de dispositivos IoT, utilizando dados reais de tráfego de rede, focando em ataques das famílias de *malware* Mirai e Gafgyt. A DNN demonstrou o maior coeficiente de determinação e as maiores taxas de verdadeiros negativos, indicando uma superior capacidade de modelar e prever as características do tráfego de rede. A DNN, sendo um algoritmo não supervisionado, opera com autonomia sem necessidade de intervenção humana constante, tornando-a mais adequada para ambientes reais com restrições de largura de banda e alta necessidade de autonomia. Embora todos os algoritmos tenham alcançado alta precisão (99%) na detecção de ataques, a DNN se destacou por seu aprendizado mais abrangente e adaptável.

Eskandari *et al.*, (2020) apresentam o Passban IDS, um sistema de detecção de intrusão projetado para dispositivos na borda da IoT. O IDS utiliza algoritmos de aprendizado de máquina de classificação não supervisionada para aprender o comportamento normal do tráfego de rede e identificar anomalias que indicam ataques. Foram utilizados dois algoritmos: *Isolation Forest (iForest)* e *Local Outlier Factor (LOF)*. O artigo descreve a implementação e avaliação do IDS em um ambiente de teste real usando um *gateway* IoT em dois cenários: um *gateway* dedicado e uma configuração "segurança na caixa". Os resultados demonstram a eficácia do IDS na detecção de vários tipos de ataques cibernéticos, incluindo varredura de portas, força bruta HTTP e SSH e ataques de inundação SYN. O algoritmo o *iForest* obteve melhor desempenho, com pontuações F1 superiores a 90% para a maioria dos ataques em ambos os cenários. O artigo destaca a capacidade do IDS de funcionar em dispositivos de baixo custo e com recursos limitados, como o Raspberry Pi 3 B, tornando-o adequado para implantação em

cenários de IoT do mundo real. No entanto, o desempenho do IDS é afetado por taxas de dados de rede muito altas, acima de 40-50 Mb/s, o que pode levar à perda de pacotes e menor precisão na detecção.

Wang e Lu (2020) propõem um HIDS para IoT utilizando o empilhamento de aprendizado de máquina *Extreme Gradient Boosting* e modelos de aprendizado profundo *Long Short-Term Memory*(LSTM). Através das sequências de chamadas do sistema, o modelo identifica anomalias. Segundo os autores o modelo superou soluções existentes, com uma pontuação AUC (*Area Under the Curve*) de 0,983, que mede capacidade do modelo de distinguir entre as classes positivas e negativas. O HIDS está restrito a sistemas com capacidade para incorporar um sistema Linux, como um *Raspberry Pi* ou PC, o que exclui grande parte dos dispositivos IoT.

Attota et al. (2021) propõem o MV-FLID, uma abordagem de detecção de intrusão baseada em aprendizado federado que utiliza múltiplas visões de dados de rede IoT para detectar ataques. Ao segmentar os dados de tráfego de rede em recursos de Fluxo Bidirecional, Fluxo Unidirecional e Pacotes, o MV-FLID visa capturar diferentes aspectos do comportamento da rede para melhorar a precisão da detecção. O processo de aprendizado federado permite que *gateways* de segurança distribuídos treinem modelos localmente e agreguem parâmetros com um servidor central, preservando a privacidade dos dados. Os autores relatam resultados promissores, demonstrando que o MV-FLID atinge alta precisão na detecção de ataques, mas não revelam valores numéricos. No entanto, eles reconhecem as limitações inerentes à sua pesquisa. A dependência do protocolo MQTT limita a aplicabilidade geral da abordagem a redes que empregam diferentes protocolos.

Priya et al. (2021) desenvolveram um modelo de detecção de intrusão de duas fases baseado em ML para melhorar a confiabilidade das redes IIoT, integrando e combinando SVM, *Naïve Bayes* e redes neurais artificiais. O modelo foi testado em três *datasets* padrão de ataque IoT (WUSTL_IIOT-2018, N_BaIoT e Bot_IoT), alcançando uma precisão máxima de 99%. A pesquisa conclui que a combinação de técnicas de ML melhora a confiabilidade das redes IIoT. O artigo não especifica os protocolos de comunicação presentes nos *datasets*. No entanto, a pesquisa contribui para o campo da segurança de IIoT principalmente ao demonstrar a eficácia da combinação de técnicas de aprendizado de máquina para detecção de anomalias. O estudo conclui que a combinação de técnicas de aprendizado de máquina, como *ensemble blending* e ANN, resulta em um método robusto e preciso para detectar intrusões em redes IIoT.

Dutta e Kant (2021) propõem a integração de uma plataforma de inteligência de ameaças cibernéticas (*Cyber Threat Intelligence* - CTI) com abordagens de ML para prever e mitigar vulnerabilidades em dispositivos IoT. Para superar as limitações do tamanho dos dispositivos IoT, sugerem a implementação do modelo em uma plataforma *TinyML* baseada em *TensorFlow*. Segundo os autores, a integração proposta pode ajudar a proteger dispositivos IoT de forma automatizada e prevenir vulnerabilidades de "*zero day*". Esta é a abordagem mais distinta de todas as demais, uma vez que utiliza de fontes de dados[15] - e não apenas os pacotes capturados - para o treinamento do modelo, além de incorporá-lo em microcontroladores. Os autores utilizaram as métricas de acurácia, precisão e *f-score* para avaliar o modelo, que alcançou uma acurácia de 96,8% no conjunto de dados de treinamento e 96,3% no conjunto de dados de teste. Além disso, o modelo demonstrou uma alta precisão de 93% na identificação correta de palavras-chave não relacionadas a ameaças. Os autores enfatizam que o foco principal está na acurácia como indicador de desempenho que se alinha com o objetivo de criar um sistema que possa fornecer previsões confiáveis sobre potenciais ameaças.

Zolotukhin, Kotilainen e Hamalainen (2021) desenvolveram uma pesquisa voltada para operadoras de redes móveis que empregam SDN e o encadeamento de funções de serviço (SFC). O foco da investigação foi a utilização de um sistema de detecção de intrusão em rede (NIDS) como um serviço para virtualização e SDN. Para isso, os autores empregaram um modelo de aprendizado por reforço (RL). Entretanto, para os fins de “prova de conceito”, os dados de treinamento foram limitados a ataques de negação de serviço (DoS), *botnet* e força bruta. Os resultados experimentais, utilizando dados do conjunto de dados CICIDS2018, mostraram que os agentes de RL conseguiram aprender a identificar e bloquear o tráfego malicioso, superando uma política de segurança estática predefinida. Os algoritmos ACER e PPO, em particular, demonstraram um bom desempenho, segundo os autores, na detecção e mitigação de ataques DDoS, botnet e de força bruta em aplicativos da web, adaptando dinamicamente as políticas de segurança e minimizando o impacto em conexões legítimas. O estudo destaca a capacidade dos agentes RL de selecionar o modelo de detecção, o tamanho da janela de tempo, o limite de classificação e as ações de redirecionamento de tráfego mais

[15] *Structured Threat Information Expression (STIX), Common Vulnerabilities and Exposures (CVE), Common Weakness Enumeration (CWE), Cyber Observable eXpression (CybOX), Trusted Automated eXchange of Indicator Information (TAXII)* e outras.

adequados com base no cenário de ataque, levando a taxas mais altas de detecção de tráfego malicioso e taxas mais baixas de falsos positivos em comparação com uma abordagem estática.

Vieira e Oliveira, Carneiro (2022) realizaram uma comparação entre seis algoritmos de *Machine Learning* (LR, KNN, GNB, DT, RF e SVM) com o objetivo de detectar intrusões em redes IoT, contemplando protocolos como MQTT e CoAP. Experimentos foram conduzidos utilizando tráfego sintético e dois conjuntos distintos de dados, levando em conta três categorias de ataques. Os achados indicaram que o tráfego bidirecional foi identificado com maior precisão, e os ataques MQTT obtiveram melhor detecção pelos algoritmos. Os pesquisadores sugerem evitar o algoritmo SVM devido ao seu elevado consumo energético e menor eficiência. O protocolo MQTT é aconselhado para garantir uma rede segura, especialmente quando empregados os algoritmos KNN e DT. Caso seja imprescindível desenvolver uma rede CoAP ou uma solução que contemple ambos os protocolos, o algoritmo DT é sugerido como opção adequada. Embora o artigo não apresente soluções de IDS e apesar da limitação de protocolos, ele forneceu informações valiosas, como os resultados dos modelos testados, que foram de grande utilidade para o desenvolvimento do trabalho proposto.

Os dez estudos mencionados nesta seção trazem diferentes técnicas, abordagens e aplicações de *Machine Learning/Deep Learning,* todos com a finalidade de identificar ataques e anomalias nas comunicações de uma rede ou em um nó.

No Quadro 7 são relacionados os artigos apresentados e esta proposta, de forma a identificar os tipos de modelos de ML empregados e o tipo de IDS quando for o caso.

Quadro 7 Trabalhos Relacionados

Trabalho	**ML Model**				**IDS**	
	Supervisionado	**Não Supervisonado**	**Ensemble Learning**	**Deep Learning**	**Host IDS**	**Network IDS**
Shukla (2017)	✓	✓	✓	✗	✗	✓
Mirsky et al. (2018)	✗	✓	✓	✗	✗	✓
Holbrook e Alamaniotis (2019)	✓	✓	✗	✓	✗	✓
Eskandari *et al.*, (2020)	✗	✓	✗	✗	✗	✓
Wang e Lu (2020)	✓	✗	✓	✓	✓	✗
Attota et al. (2021)	✓	✗	✓	✗	✗	✓
Priya et al. (2021)	✓	✗	✓	✓	✗	✗
Dutta e kant (2021)	✓	✗	✗	✓	✓	✗

Trabalho	ML Model				IDS	
	Supervisionado	**Não Supervisonado**	**Ensemble Learning**	**Deep Learning**	**Host IDS**	**Network IDS**
Zolotukhin, Kotilainen e Hamalainen (2021)	✗	✗	✗	✗	✗	✓
Vieira, Oliveira e Carneiro (2022)	✓	✓	✓	✓	✗	✗
IoT-IDS	✓	✗	✗	✗	✗	✓

A maioria dos trabalhos relacionados se concentram em métodos específicos ou restritos a tipos limitados de ataques ou protocolos, como o trabalho de Shukla (2017), que aborda apenas o ataque *Wormhole* usando dados do protocolo RPL, e o estudo de Zolotukhin et al. (2021), que foca em ataques DoS, *botnets* e força bruta dentro de um ambiente de SDN. Já o *Kitsune* (Mirsky *et al.*, 2018), que, embora eficiente, se baseia principalmente na detecção de anomalias com o uso de *autoencoders*, não tem a capacidade de classificação explícita dos tipos de ataques, assim como o *Passban* (Eskandari *et al.*, 2020), que usa um classificador de classe única.

O IoT-IDS, proposto neste texto diferencia-se significativamente de outros sistemas de detecção de intrusões (IDS) descritos, pois oferece uma abordagem mais abrangente e adaptativa. Ele não apenas detecta, mas também classifica uma ampla variedade de ataques em tempo real, emitindo alertas quando necessário. Alertas estes que podem ser usados como gatilhos para execução de rotinas de tratamento dos ataques específicos, o que é crucial para a segurança em redes IoT complexas e diversificadas. Ele foi desenvolvido utilizando o *TensorFlow* para a criação de um modelo supervisionado robusto, que pode ser treinado com dados rotulados para alcançar um alto desempenho, como demonstrado em seus resultados de precisão e *recall*. Isso o torna uma solução mais poderosa e aplicável em ambientes onde a identificação rápida e correta do tipo de ataque é essencial para a mitigação e resposta adequada.

O IoT-IDS utiliza uma rede neural para a classificação de ataques, mas não se enquadra na categoria de *deep learning*, pois emprega apenas uma camada oculta, conforme descrito na arquitetura do modelo (subseção 3.3.4). Essa abordagem, além de realizar a classificação de forma eficaz, se destaca por sua eficiência em termos de tempo de resposta e uso de memória. Isso torna a solução altamente adaptável para execução em dispositivos com limitações computacionais, incluindo aqueles que operam sem a necessidade de sistemas operacionais.

No entanto, vale ressaltar algumas limitações. O modelo de aprendizado de máquina (ML) utilizado no IoT-IDS foi treinado com uma quantidade limitada de dados para dois dos tipos de ataque, o que pode ter resultado em erros na classificação desses ataques. Além disso, o modelo enfrenta dificuldades ao lidar com ataques que não pertencem às quatorze classes com as quais foi treinado, já que é incerto como esses ataques serão classificados, com a possibilidade de serem erroneamente identificados como *"Normal Traffic"*.

3.5. Considerações finais

Este capítulo apresentou o IoT-IDS, um sistema de detecção e classificação de intrusões para redes IoT baseado em rede neural. O capítulo descreve a abordagem, arquitetura do sistema e o desenvolvimento do modelo de ML que faz parte do IDS. O capítulo também discute as vantagens do IoT-IDS em relação a outros trabalhos relacionados, destacando sua capacidade de detectar e classificar diversos tipos de ataques em tempo real, usando um modelo de machine learning e finaliza apontando suas limitações.

4. ESTUDO EMPIRICO

A fim de validar a eficácia da abordagem proposta para o IoT-IDS, foi realizado um estudo empírico, envolvendo a implementação e a avaliação do sistema em um ambiente experimental controlado. Este capítulo descreve detalhadamente o protocolo experimental, os resultados obtidos e as análises realizadas para compreender o desempenho do sistema proposto. Portanto os objetivos são:

- Validar a Capacidade de Detecção: Verificar se o IoT-IDS é capaz de identificar corretamente diferentes tipos de ataques em um ambiente IoT, incluindo ataques *Port Scanning*, *DoS* e *APR Spoofing*;
- Avaliar o Impacto no Desempenho: Analisar o impacto do IoT-IDS no desempenho geral do sistema IoT, medindo fatores como consumo de recursos (CPU, memória) e possíveis degradações no tempo de resposta dos dispositivos IoT monitorados;
- Comparar com outros Existentes: Comparar os resultados obtidos com Kitsune avaliando o diferencial da solução proposta.

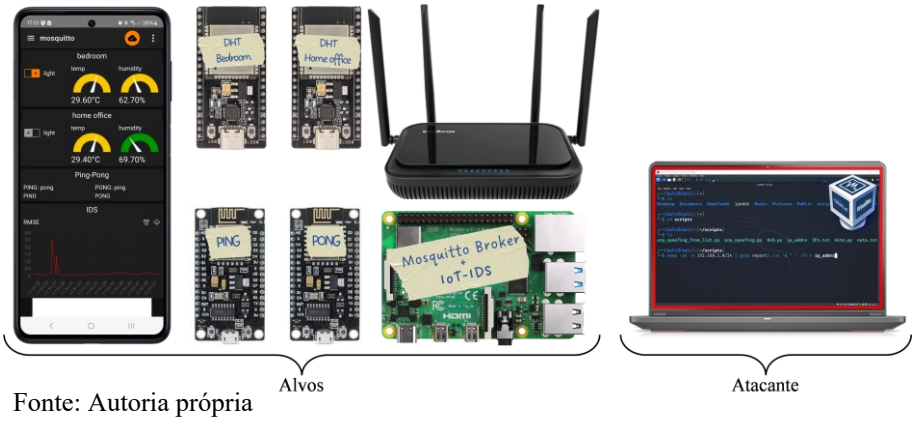
4.1. Protocolo

Para atingir os objetivos foi desenvolvido um protocolo que segue os seguintes passos: Configuração do ambiente de testes, Testes de Penetração e Resultados.

4.1.1. Configuração do ambiente de testes

Como o propósito é ter um cenário com dispositivos IoT se comunicando normalmente e com IoT-IDS monitorando toda a atividade de rede do conjunto, foi configurado um ambiente de testes controlados com os seguintes dispositivos: 1 *smartphone* Android, 2 ESP32_DHT com sensores DHT22 (umidade relativa do ar e temperatura), 2 ESP8266, 1 *Raspberry Pi 4* modelo B e um roteador WiFi. Na Figura 20 estão os dispositivos listados e um computador de onde foram executados os ataques. No contexto do estudo, presume-se que o atacante já conseguiu acesso à rede WiFi, pois escopo da pesquisa não se estende ao controle de acesso à rede em si. Essa abordagem visa concentrar-se especificamente na detecção de possíveis ameaças e ataques aos dispositivos IoT.

Figura 20 Dispositivos que compõem o ambiente de testes.



Fonte: Autoria própria

O *Raspberry Pi 4* modelo B[16] é um *single board computer* (SBC), ou computador de placa única, com um processador *ARM v8 Quad core Cortex-A72 64-bit de 1.5GHz na versão de 8GB de memória RAM*. Ele foi configurado com o sistema operacional *Raspberry Pi OS Lite*[17] de 64bits, sem interface gráfica. Neste foram configurados o *Eclipse Mosquitto*[18], o *sys_mon (System Monitor)* e os IDS.

O *Mosquitto* é um servidor de código aberto que implementa o protocolo MQTT (*Message Queuing Telemetry Transport*) que é um protocolo de comunicação projetado para a troca de mensagens entre dispositivos em redes IoT com o uso do modelo *publisher/subscribe***,** onde as mensagens são enfileiradas em tópicos.

O *script sys_mon* monitora e registra periodicamente dados de uso de recursos do sistema, incluindo CPU, memória, disco e rede, de cada processo ativo no sistema. Para cada processo, são coletados dados como o uso percentual de CPU e memória, contagem de threads, e operações de leitura e escrita em disco. Essas informações são registradas em um arquivo de log em formato JSON, com *timestamp* para cada entrada, possibilitando uma análise detalhada do comportamento dos processos ao longo do tempo. Esses dados de log servirão como base para a análise final de desempenho e uso de recursos do sistema.

Os dois dispositivos ESP32_DHT foram montados com o *NodeMCU 32s* e um sensor DHT22. O dispositivo é capaz de medir a temperatura e a umidade relativa do ar. Ele também

[16] https://www.raspberrypi.com/products/raspberry-pi-4-model-b/
[17] https://www.raspberrypi.com/software/operating-systems/
[18] https://mosquitto.org/

publica esses dados, juntamente com o estado do LED interno em um tópico específico. É possível controlar o estado do LED publicando uma mensagem no tópico "*light*" do dispositivo.

Os dois dispositivos PING e PONG, montados com *NodeMCU ESP8266*, são empregados com a finalidade de se comunicarem mutuamente através do protocolo MQTT, alternadamente ativando e desativando um LED interno em cada placa. A finalidade é apenas de criar um fluxo de comunicação contínuo e bidirecional.

O *smartphone* Android tem a finalidade de executar o aplicativo *IoT MQTT Panel*[19] que permite gerenciar e visualizar projetos de IoT, baseados no protocolo MQTT, através de uma *dashboard* personalizável.

O computador foi utilizado para a execução de uma máquina virtual (VM) com o sistema operacional *Kali Linux* que é uma distribuição Linux de código aberto que permite executar testes avançados de penetração e auditoria de segurança (OffSec Services Limited, 2024). A VM foi configurada no modo *Bridge*, assim ela se comporta como um dispositivo independente na mesma rede que a máquina hospedeira, se comunicando diretamente com todos os outros dispositivos na rede e vice-versa.

4.1.2. Testes de Penetração

Os testes de penetração são fundamentais para avaliar a robustez e a resiliência de sistemas de segurança, especialmente em ambientes IoT, onde a diversidade de dispositivos e protocolos cria um cenário propício para uma ampla gama de ataques. Foram realizados testes de penetração que simulam ataques reais, incluindo *Port Scanning*, *ARP Spoofing*, um ataque DoS direcionado ao protocolo MQTT e *OS Fingerprinting*. As descrições de execução que seguirão nos próximos itens foram executadas manualmente para testes e registro, porém um *script Bash* foi desenvolvido para automatizar a execução dos testes de penetração, garantindo a repetição dos testes de forma padronizada, permitindo uma avaliação consistente e confiável do sistema. Esses testes têm como objetivo não apenas identificar vulnerabilidades, mas também validar a capacidade do IoT-IDS em detectar essas ameaças.

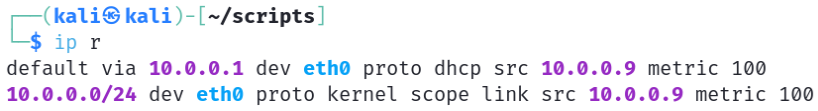
[19] https://play.google.com/store/apps/details?id=snr.lab.iotmqttpanel.prod

4.1.2.1. Port Scanning

O escaneamento de portas é uma técnica sistemática usada para explorar sistemas de computação em rede tentando se conectar a várias portas de rede em um dispositivo. O objetivo principal é identificar quais portas estão abertas e quais serviços estão em execução, o que pode revelar possíveis vulnerabilidades que os invasores podem explorar. Embora frequentemente associada a atividades maliciosas, a verificação de portas também é uma prática legítima empregada por profissionais de TI para solução de problemas e avaliações de segurança (Pittman, 2023);

Antes de executar o escaneamento de portas propriamente dito é preciso identificar o endereço de rede, para isso o comando *ip r* já é suficiente. Na Figura 21 consta a saída do *ip r* no terminal do Kali Linux.

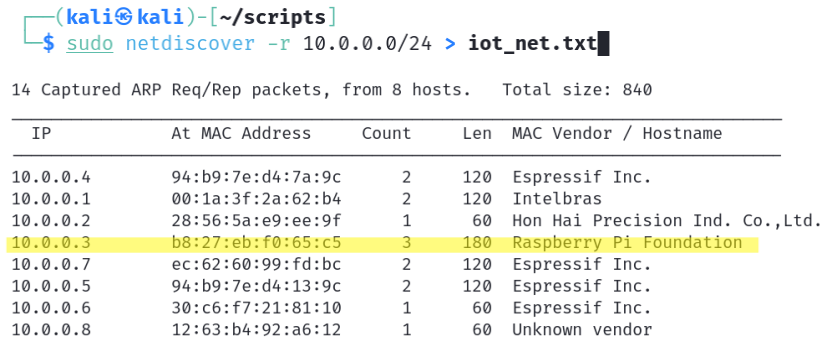
Figura 21 Comando ip r e resultados.



Fonte: Autoria própria

Para a execução do teste foi utilizada a ferramenta *netdiscover* via terminal no Kali Linux. Na Figura 22 consta a linha de comando chamando o *netdiscover* e passando alguns argumentos para salvar os resultados encontrados pela ferramenta. Na figura estão os resultados que o *netdiscover* salvou no arquivo iot_net.txt.

Figura 22 Aplicação Netdiscover e resultados



Fonte: Autoria própria

Com as informações do passo anterior o próximo passo é verificar as portas dos serviços em execução. No caso o alvo é o Raspberry Pi (10.0.0.3), que geralmente é usado para execução de serviços como um broker, servidor web, acesso remoto etc. Na Figura 23 o resultado da execução do *Nmap* direcionado para o endereço IP do servidor mostra que as portas 22 e 1883 estão abertas e ainda identifica os respectivos serviços como SSH e MQTT. De posse desta informação é possível explorar as vulnerabilidades dos serviços ou descuidos por parte de quem os configurou, como, por exemplo a ausência de criptografia, senhas fracas etc.

Figura 23 Aplicação Nmap e resultados.

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -p- 10.0.0.3

Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-18 22:13 EDT
Nmap scan report for 10.0.0.3
Host is up (0.0040s latency).
Not shown: 65533 closed tcp ports (reset)
PORT     STATE SERVICE
22/tcp   open  ssh
1883/tcp open  mqtt
MAC Address: B8:27:EB:F0:65:C5 (Raspberry Pi Foundation)

Nmap done: 1 IP address (1 host up) scanned in 26.15 seconds
```

Fonte: Autoria própria

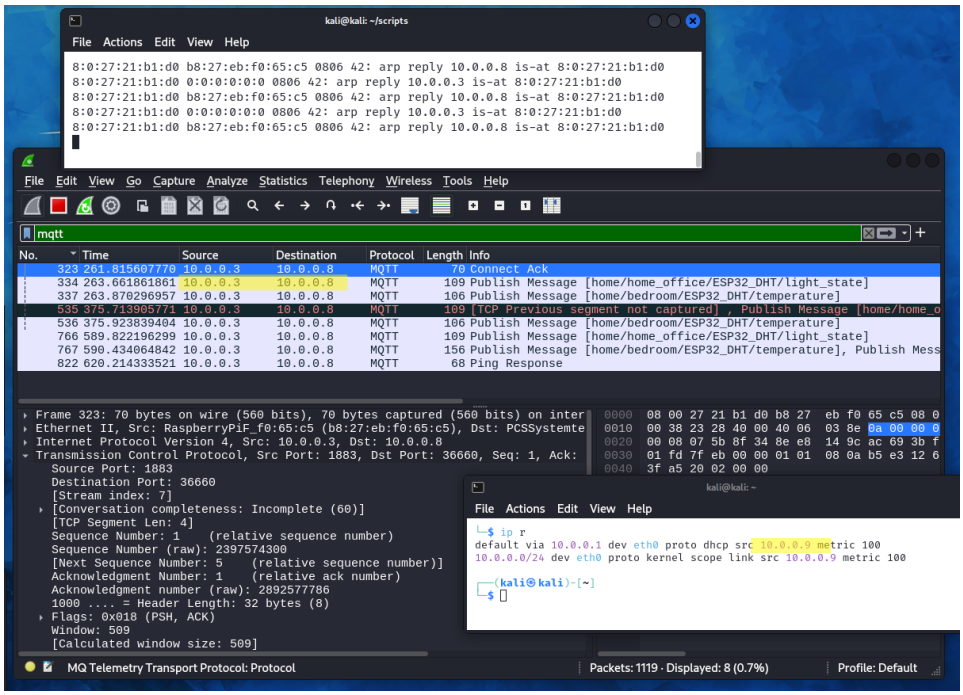
4.1.2.2. ARP Spoofing

O Protocolo de resolução de endereços (*Address Resolution Protocol* - ARP) é um protocolo utilizado para mapear endereços IP a endereços MAC (físicos) em uma rede local. O *ARP Spoofing,* ou falsificação de ARP, é uma técnica de ataque em redes de computadores que envolve a falsificação de endereços de ARP. Um atacante envia mensagens ARP falsas na rede, associando seu próprio endereço MAC ao endereço IP de um dispositivo legítimo. Isso pode levar a várias consequências, como interceptação de dados, ataques de *Man-in-the-middle* e *DoS* (Ferrag *et al.*, 2022);

O ataque foi executado com o uso da ferramenta *arpspoof* presente no Kali Linux de forma a executá-lo de forma bidirecional. Os argumentos passados ao *arpspoof* são a interface de rede do atacante o endereço IP do alvo e o endereço IP do dispositivo a ser enganado (`arpspoof -i eth0 -t 10.0.0.3 -r 10.0.0.8`.).

Na Figura 24 podem ser observadas três janelas, onde a da porção superior é o terminal onde o *arpspoof* está em execução, a maior em segundo plano é da apliacação *WireShark,* exibindo uma lista de pacotes interceptados e o terminal mais abaixo apenas mostra o endereço

Figura 24 Execução de um ataque com a ferramenta arpspoof.



Fonte: Autoria própria.

IP da VM (10.0.0.9) comprovando o funcionamento do *arpspoof* permitindo a execução de um ataque *Men in the Middle* interceptando mensagens trocadas dispositivos na rede.

4.1.2.3. Denial of Service (DoS)

Ou negação de serviço, é uma tentativa maliciosa de interromper o funcionamento normal de um serviço, geralmente um servidor, um serviço ou uma rede, tornando-o indisponível para os usuários. Os ataques DoS são realizados por meio do envio de uma quantidade excessiva de solicitações ao alvo, sobrecarregando-o e impedindo que ele responda a requisições legítimas (Silva *et al.*, 2020).

Para realizar o DoS foi desenvolvido um *script* em Python que realiza *N* conexões com o *broker* e para cada conexão publica initerruptamente valores aleatórios, como pode ser observado na Figura 25. O script pode ser executado mais de uma vez ao mesmo tempo em terminais diferentes, causando uma inundação no *broker.* Na Figura 25 está o trecho principal do script.

Figura 25 Núcleo do script de ataques DoS.

```
def create_connection(n):
    global running
    n = str(n)
    topic = "TryDoS_"+n
    client = mqtt.Client()
    client.username_pw_set(username=mqtt_username, password=mqtt_password)
    client.connect(broker_address, broker_port)
    print(f"Thread {n}")
    while running:
        payload = generate_payload()
        result, mid = client.publish(topic, payload)
        if result != 0:
            print("Publication failed")
        client.loop()
threads = []
for i in range(N):
    thread = threading.Thread(target=create_connection, args=(i,))
    threads.append(thread)
    thread.start()
```

Fonte: Autoria própria

4.1.2.4. OS Fingerprinting

O ataque *OS fingerprinting*, no contexto da IoT, visa identificar o sistema operacional de um dispositivo através da análise de pacotes de dados. É uma técnica que explora as sutis diferenças na forma como os diferentes sistemas operacionais implementam os protocolos de rede. Estas diferenças, embora mínimas, geram assinaturas únicas que podem ser usadas para determinar o SO do dispositivo, de posse destes dados um atacante pode explorar as vulnerabilidades conhecidas do SO em questão (Ferrag *et al.*, 2022). Nos testes foi utilizado o *Nmap* com a opção '-O' conforme apresentado na Figura 26.

Figura 26 OS Fingerprinting executado com a indicação do SO.

```
┌──(kali㉿kali)-[~/scripts]
└─$ sudo nmap -O 10.0.0.3
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-02 10:51 -03
Nmap scan report for 10.0.0.3
Host is up (0.0089s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: B8:27:EB:A5:30:90 (Raspberry Pi Foundation)
Aggressive OS guesses: Linux 2.6.32 (96%), Linux 3.2 - 4.9 (96%), Linu
x 4.15 - 5.8 (96%), Linux 2.6.32 - 3.10 (96%), Linux 5.0 - 5.5 (96%),
Linux 3.4 - 3.10 (95%), Linux 3.1 (95%), Linux 3.2 (95%), AXIS 210A or
 211 Network Camera (Linux 2.6.17) (95%), Synology DiskStation Manager
 5.2-5644 (94%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https:/
/nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.41 seconds
```

Fonte: Autoria própria

4.2. Resultados

Nos testes de penetração, cada ataque foi executado de forma automatizada por aproximadamente cinco minutos em três cenários distintos: inicialmente, com o ambiente de testes configurado apenas com a configuração padrão do broker e dispositivos, conforme descrito na subseção 4.1.1; em seguida, com o IoT-IDS; e, por fim, com o Kitsune adaptado para análise em tempo real (Kitsune_IDS). Em cada cenário, o sistema foi reiniciado três vezes para assegurar a consistência dos dados obtidos.

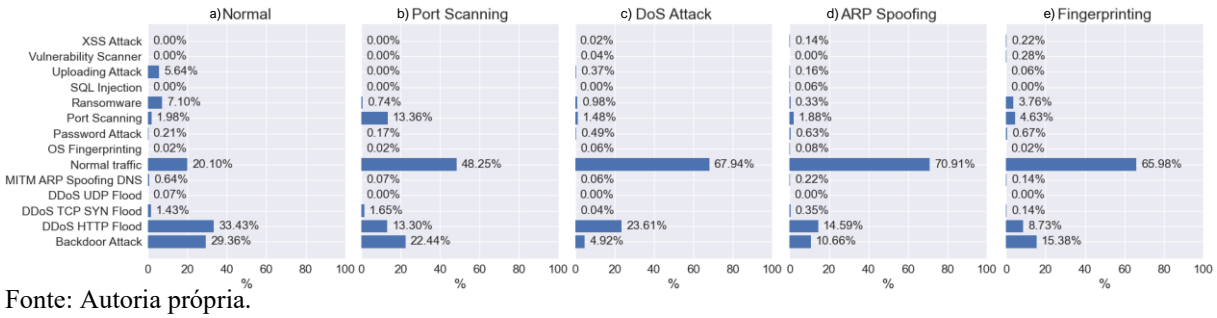
Na Figura 27 estão ilustrados os resultados proporcionais para cada simulação. Na simulação *Normal,* Figura 27(a), onde os dispositivos IoT se comunicavam sem qualquer tentativa de invasão, apenas 20,1% dos pacotes foram classificados corretamente, enquanto 79,9% correspondem a falsos positivos. Na simulação de *Port Scanning,* Figura 27(b), o NetStatClassifier obteve uma taxa de acerto de apenas 13,36%.

Na simulação de *DoS,* Figura 27(c), 2,18% dos pacotes foram classificados como *DDoS HTTP Flood*. Contudo, essa classificação não reflete precisamente o ataque realizado, pois existem diferenças significativas entre os ataques. No caso do *DDoS*, os pacotes são distribuídos entre múltiplas origens, enquanto na simulação realizada os pacotes se originaram de um único atacante, utilizando o protocolo MQTT.

Já na simulação de *ARP Spoofing,* Figura 27(d), a taxa de classificação correta foi de apenas 0,22%. No caso do ataque *OS Fingerprinting,* Figura 27(e), a precisão foi ainda menor, com apenas 0,02% de acerto.

Destaca-se que, em todos os cenários, houve uma alta incidência de falsos positivos para as categorias *Backdoor Attack* e *DDoS HTTP Flood*. Além disso, a categoria Normal apresentou percentuais elevados, que podem representar tanto falsos positivos quanto

Figura 27 Resultados das classificações do IoT-IDS durante pentest.

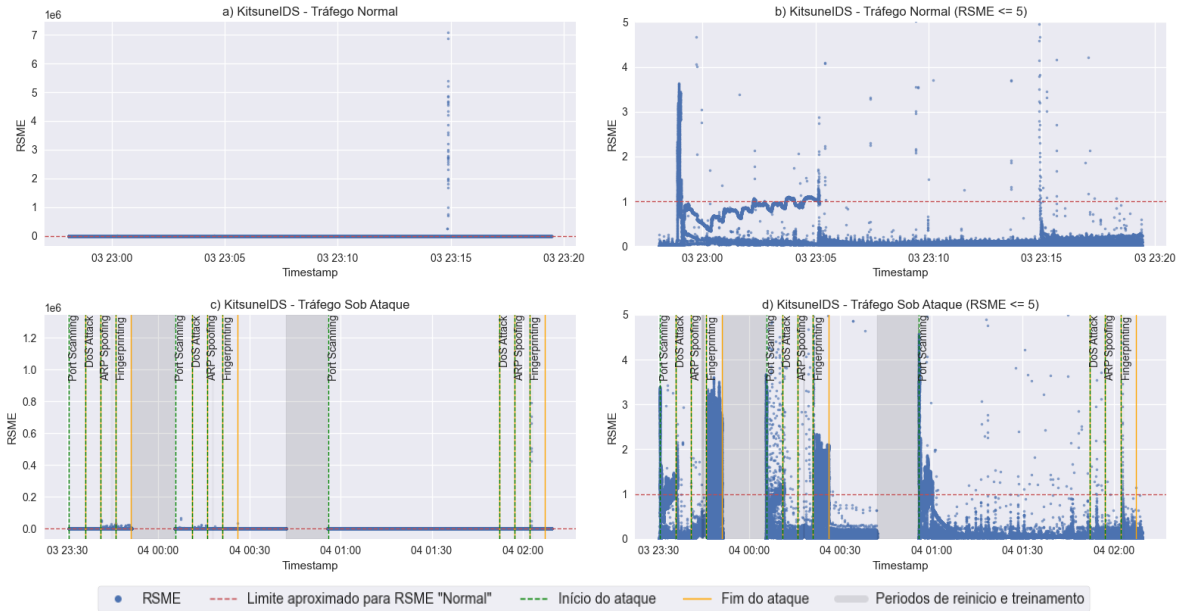


Fonte: Autoria própria.

classificações corretas, considerando a presença de tráfego legítimo concomitante ao tráfego de ataque.

Para fins de comparação, as mesmas simulações foram realizadas utilizando o Kitsune_IDS. Os gráficos apresentados na Figura 28 representam a saída do Kitsune_IDS, medida pela raiz do erro quadrático médio (RMSE). Esse valor reflete o grau de diferença entre os pacotes utilizados no treinamento e aqueles processados após o treinamento. Valores elevados de RMSE indicam a presença de anomalias, enquanto valores próximos ao intervalo de 0 a 1 correspondem a tráfego normal. Nas Figuras 28(a) e 28(c), os pontos representam os valores de RMSE em toda a sua escala. Por outro lado, nas Figuras 28(b) e 28(d), os valores de RMSE são limitados a um máximo de 5, permitindo observar com mais clareza a variação neste intervalo. Na Figura 28(b), mesmo em cenários de tráfego normal, o Kitsune_IDS indica a presença de 12.73% de anomalia (pontos acima de 1). Já nas Figuras 28(c) e 28(d), para os ataques Port Scanning, DoS, ARP Spoofing e OS Fingerprinting, as anomalias representam 15.49%, 5,22%, 1,07% e 22,62% respectivamente. As alterações para ARP Spoofing e OS Fingerprinting são mais perceptíveis na Figura 28(c), especialmente na segunda rodada de simulações. Essa variação nos resultados sugere uma possível instabilidade no desempenho do Kitsune_IDS, uma vez que a detecção de anomalias ocorre de forma inconsistente, sendo identificada em determinados momentos e ausente em outros.

Figura 28 Gráficos das saídas RSME do Kitsune_IDS durante pentest. As figuras a e c representam os dados em sua escala total. As figuras b e d limitam o RSME até 5.

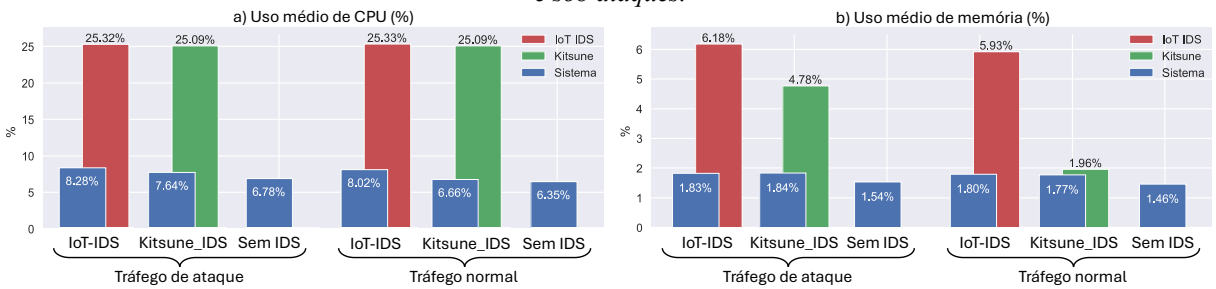


Fonte: Autoria própria.

A despeito dos resultados de classificação e considerando um *Raspberry Pi 4B* como *hardware* do servidor, equipado com 8GB de memória LPDDR4, CPU quad-core Cortex-A72 com clock de 1.8GHz e um cartão MicroSD de 32GB classe 10, executando um sistema operacional Raspberry Pi OS Lite 64-bit (sem interface gráfica), a análise do consumo de recursos computacionais demonstra que a implementação é viável em contextos de *hardware* semelhantes ou superiores. A solução não compromete significativamente o desempenho do servidor testado, mantendo-se eficiente em um ambiente de recursos limitados. Essa viabilidade sugere que, com um classificador aprimorado, a abordagem pode atingir melhores resultados sem impactos negativos no sistema. Durante a execução do IDS, os dispositivos IoT mantiveram uma comunicação estável com o servidor, sem indícios de falhas ou atrasos significativos, mesmo sob condições de ataque simuladas.

Nos gráficos da Figura 29 são apresentadas as médias de ocupação do processador e memória e para o sistema e destacando os IDS. Os dados foram obtidos com um *script Python* desenvolvido para monitorar processos do sistema, usando a biblioteca *psutil*[20] para coletar informações detalhadas, como ID, uso de CPU, memória, I/O e número de threads, que foram registradas em um arquivo de log. Na Figura 29(a), observa-se que o uso de CPU para ambos os IDS é similar, mantendo-se abaixo de 26%. Já na Figura 29(b), nota-se que com o IoT-IDS a média de uso de memória aumentou em 0,25% ao processar tráfego de ataque, enquanto o Kitsune_IDS apresentou um aumento de 2,82%, indicando maior variação no consumo de memória, apesar de a média sob ataque ser 1,4% menor. Em ambos os IDS, o consumo de CPU e memória pelo que foi classificado como 'Sistema' aumentou. Isso ocorre porque os IDS, por

Figura 29 Consumo médio de memória e CPU sem IDS, com o IoT-IDS e com o KitsuneIDS com tráfego normal e sob ataques.



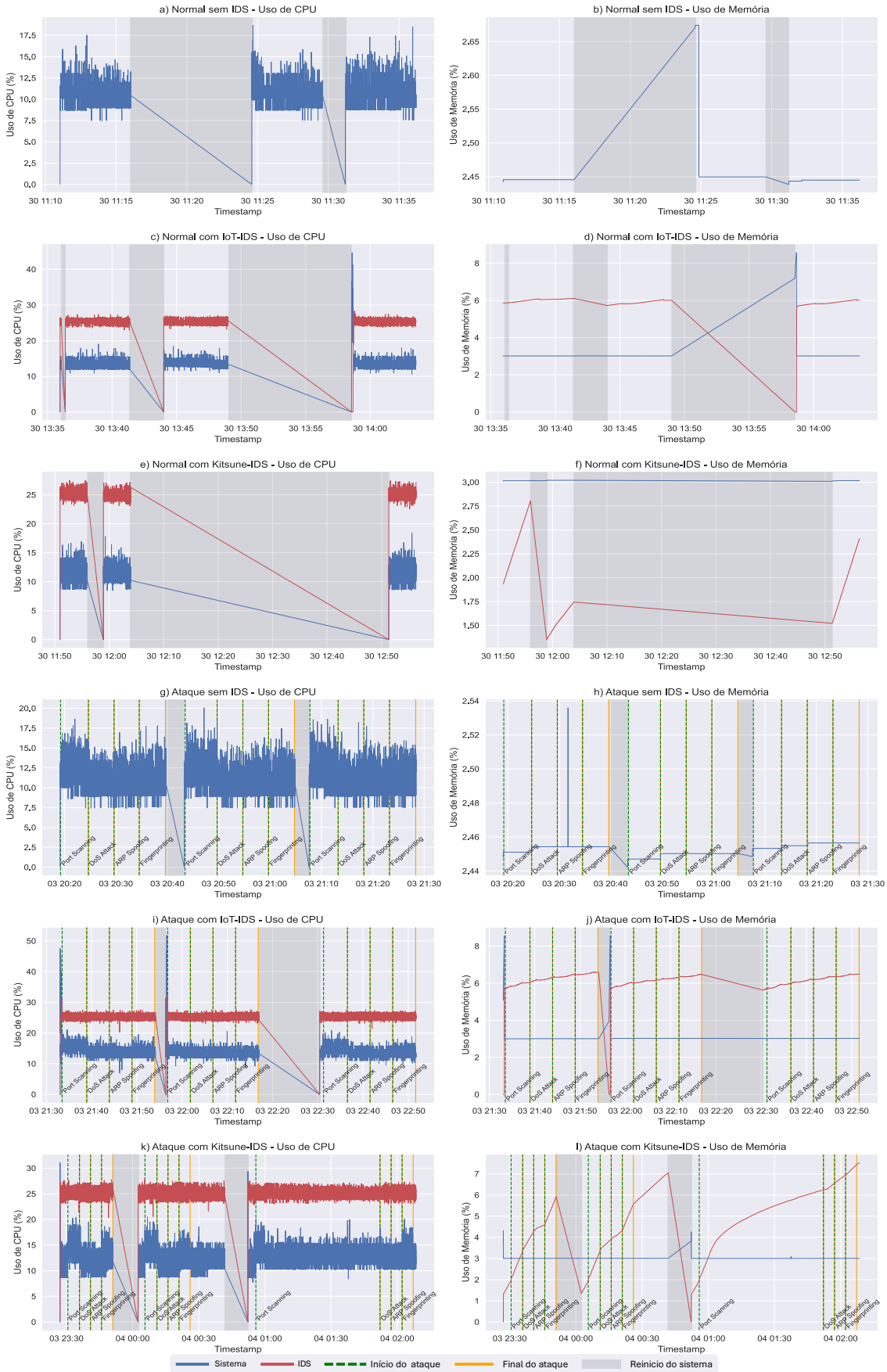
Fonte: Autoria própria.

[20] https://pypi.org/project/psutil

serem escritos em Python, introduzem processos adicionais do interpretador durante sua execução, o que se soma à carga geral do sistema.

Na Figura 30 são apresentados os mesmos dados, mas agora em função do tempo de execução das simulações. A Figura 30(a) mostra o consumo de CPU em percentual durante o *pentest* sem IDS, enquanto a Figura 30(b) apresenta o consumo de memória no mesmo cenário. As Figuras 30(c) e 30(d) exibem o consumo de CPU e memória, respectivamente, com o IoT-IDS ativo, e as Figuras 30(e) e 30(f) mostram esses dados para o Kitsune_IDS. Nas Figuras 30(g) e 30(h) pode ser observados o consumo de CPU e memória durante os ataques sem IDS, enquanto as Figuras 30(i) e 30(j) ilustram o consumo com o IoT-IDS e as Figuras 30(k) e 30(l), com o Kitsune_IDS.

Figura 30 Consumo de CPU e memória ao longo do tempo sem IDS, com o IoT-IDS e com o KitsuneIDS com tráfego normal e sob ataques.



Fonte: Autoria própria.

4.3. Discussão

Os resultados obtidos nos testes de penetração indicam que tanto o IoT-IDS quanto o Kitsune_IDS apresentaram dificuldades em detectar e classificar corretamente os ataques simulados. Em particular, o IoT-IDS demonstrou uma alta taxa de falsos positivos, alcançando quase 80% durante o tráfego normal, e falhou na identificação de ataques específicos, como *ARP Spoofing* e *OS Fingerprinting*, com menos de 1% de acerto para ambos. Esses resultados sugerem que o modelo de aprendizado de máquina implementado no IoT-IDS pode não estar suficientemente ajustado para lidar com a diversidade de tráfego e tipos de ataques presentes nas simulações.

Apesar dessas limitações, é importante destacar que o consumo de recursos computacionais pelo IoT-IDS não impacta significativamente o desempenho dos dispositivos IoT monitorados. O uso de CPU se manteve abaixo de 26%, enquanto a memória RAM utilizada foi em torno de 500MB. Esses indicadores sugerem que, embora a eficácia na detecção precise ser aprimorada, a abordagem proposta é viável do ponto de vista operacional e pode ser aplicada em cenários reais de IoT com hardwares similares.

Ao comparar o desempenho do IoT-IDS com o Kitsune_IDS, verifica-se que ambos os sistemas apresentam limitações semelhantes, em parte devido ao uso compartilhado do componente NetStat. Essas limitações também podem estar relacionadas às diferenças entre os conjuntos de dados utilizados para treinamento e os dados do ambiente de testes. Em testes realizados durante a configuração do ambiente, pequenos ajustes, como a adaptação da faixa de endereços IP, pareceram influenciar modestamente os resultados, embora a melhora observada tenha sido sutil e não conclusiva. Ainda assim, o IoT-IDS, assim como o Kitsune_IDS, não foi capaz de identificar todas as anomalias presentes.

É importante também observar que o Kitsune_IDS apresentou um consumo de CPU semelhante ao IoT-IDS, mas demonstrou uma variação considerável no uso de memória, que oscilou entre 160 e 391MB, com uma tendência de aumento ao longo do tempo. Essa variação pode indicar a possibilidade de um aumento gradual do uso de memória conforme o sistema continua operando, o que deve ser levado em consideração em implementações de longo prazo.

4.4. Considerações finais

Neste capítulo foram apresentados os resultados de um estudo empírico que avaliou o desempenho do IoT-IDS em um ambiente de testes simulado, demonstrando que, apesar do baixo consumo de recursos e da viabilidade operacional, o sistema ainda enfrenta desafios em termos de precisão e capacidade de classificação de ataques. Os testes revelaram uma taxa elevada de falsos positivos e dificuldades na detecção de alguns tipos de intrusões, indicando a necessidade de aprimoramento no modelo de aprendizado de máquina utilizado. Embora o IoT-IDS tenha demonstrado desempenho ao Kitsune_IDS, os resultados ressaltam a importância de continuar desenvolvendo e refinando o modelo de classificação do sistema.

5. CONCLUSÃO

Este trabalho apresentou a proposta de abordagem e a implementação de um para um Sistema de Detecção de Intrusão de Rede (NDIS) voltado para ambientes de Internet das Coisas, denominado IoT-IDS, utilizando técnicas de aprendizado de máquina para melhorar a segurança em redes IoT. A motivação para esta pesquisa surgiu da crescente preocupação com a segurança e privacidade dos dados em IoT, devido à proliferação de dispositivos conectados e às suas limitações em termos de recursos computacionais, que dificultam a aplicação de métodos de segurança tradicionais.

A pesquisa envolveu a construção de um conjunto de dados a partir do *Edge-IIoTset* e do *netStat*, adaptando o modelo de extração de características utilizado no Kitsune para operar em tempo real e classificando diversos tipos de ataques específicos ao ambiente IoT.

O modelo de ML *NetStatClassifier*, desenvolvido e treinado com o conjunto de dados *EdgeIIoTset_netStat*, nos resultados de suas métricas, demonstrou ser uma ferramenta eficaz na classificação de diferentes tipos de tráfego, incluindo ataques e tráfego normal. No entanto, os testes revelaram que, apesar da eficácia operacional, o sistema ainda enfrenta desafios relacionados à precisão na detecção de intrusões apontando para a necessidade de melhorias futuras no modelo de aprendizado de máquina empregado.

Os resultados gerais do estudo demonstraram a viabilidade do IoT-IDS em ambientes reais, sendo capaz de operar com baixo consumo de recursos computacionais, o que é crucial para dispositivos IoT com restrições em termos de processamento e energia.

A avaliação do sistema foi realizada em um ambiente de testes realista, utilizando um conjunto de dispositivos IoT e simulando uma série de ataques. Os resultados indicaram que, embora o IoT-IDS seja eficaz em operar com baixo consumo de recursos e tenha demonstrado viabilidade operacional, ele ainda apresenta desafios significativos em termos de precisão na detecção e classificação de ataques, evidenciados pela alta taxa de falsos positivos e dificuldades em identificar intrusões.

5.1. Ameaças à validade do estudo

Uma das principais ameaças à validade deste estudo é a limitação no conjunto de dados utilizado para treinar e testar o modelo IoT-IDS. Embora o *Edge-IIoTset* seja um *dataset* abrangente os dados selecionados a partir dele podem ter sido insuficientes.

Outra ameaça à validade do está na limitação da capacidade de generalizar os resultados para outros ambientes de IoT, visto que o estudo foi conduzido em um ambiente controlado e com um conjunto específico de dispositivos e ataques. Diferentes configurações de hardware, topologias de rede e padrões de tráfego podem levar a resultados significativamente diferentes. Ademais, a diversidade de dispositivos IoT e protocolos de comunicação em ambientes reais pode introduzir variabilidades não consideradas neste estudo, impactando a aplicabilidade dos resultados em outros contextos.

5.2. Contribuições

A principal contribuição deste trabalho reside na proposta de uma abordagem para a detecção de intrusões em sistemas de Internet das Coisas (IoT). A abordagem desenvolvida combina técnicas de aprendizado de máquina com um Sistema de Detecção de Intrusão especificamente adaptado para o ambiente IoT, caracterizado por suas limitações em termos de recursos computacionais e pela heterogeneidade dos dispositivos. O IoT-IDS foi projetado para operar de forma centralizada, realizando a captura, extração de características, classificação de tráfego e ação de mitigação de forma eficiente e em tempo real. A implementação dessa abordagem demonstrou a capacidade de integrar um nível adicional de segurança aos sistemas IoT, utilizando modelos de aprendizado de máquina.

Além disso, o trabalho também contribui ao demonstrar a viabilidade operacional do IDS proposto em ambientes IoT reais. Os testes realizados em um ambiente de simulação mostraram que o IoT-IDS é capaz de funcionar com baixo consumo de recursos, mantendo a integridade do desempenho dos dispositivos monitorados. Essa característica é crucial para a aplicação prática em sistemas IoT, onde a economia de recursos, como energia e capacidade de processamento, é essencial.

Estas contribuições, embora apresentem limitações, fornecem uma base inicial para o desenvolvimento de futuras soluções de segurança em redes IoT. Apesar dos desafios encontrados nos resultados de classificação, o IoT-IDS ainda oferece elementos que podem ser explorados e aprimorados em estudos futuros, buscando ampliar a proteção e a robustez desses sistemas em um cenário de ameaças cibernéticas em constante evolução.

5.3. Perspectivas de Continuidade

Dada a natureza dinâmica e em rápida evolução das tecnologias de IoT, é essencial que futuras investigações continuem a aprimorar os mecanismos de detecção, expandam as capacidades do sistema, e explorem novas abordagens que possam superar as limitações identificadas neste estudo. A seguir, são sugeridas algumas direções que podem ser exploradas a partir deste trabalho, visando obter melhor desempenho ou mesmo mudando de abordagem.

- Verificar o quanto a modificação do ambiente, como configurações de rede e dispositivos, pode afetar o comportamento do IoT-IDS quanto a deteção de invasões;

- Implementação do IoT-IDS em um roteador WiFi com o uso de sistemas como OpenWrt[21] com a finalidade de usá-lo como NDIS não só dos dispositivos IoT, mas da rede como um todo. Junto a isso converter o modelo com o *TensorFlow Lite* como forma de reduzir o consumo de recursos computacionais e ampliar a eficiência do conjunto *software* e *hardware*.

- Usar o IoT-IDS nos *hosts* (HIDS) fazendo uma adaptação ou conversão do modelo para execução em dispositivos IoT baseados em microcontroladores com o uso de tecnologias como *TinyML*[22] ou *AIfES*[23].

- Implementar um modelo IoT-IDS com aprendizado federado de modo a facilitar o treinamento de novas ameaças.

[21] Open Wireless Router - openwrt.org
[22] TinyML - tinyml.org
[23] Artificial Intelligence for Embedded Systems - aifes.ai

REFERÊNCIAS

AKRAM, Hezam; KONSTANTAS, Dimitri; e MAHYOUB, Mohammed. A Comprehensive IoT Attacks Survey based on a Building-blocked Reference Model. **International Journal of Advanced Computer Science and Applications**, [*s. l.*], v. 9, n. 3, 2018. ISSN 21565570, 2158107X. DOI 10.14569/IJACSA.2018.090349. Disponível em: http://thesai.org/Publications/ViewPaper?Volume=9&Issue=3&Code=ijacsa&SerialNo=49. Acesso em: 11 maio 2022.

ALGARNI, Mona; ALKHELAIWI, Munirah; e KARRAR, Abdelrahman. Internet of Things Security: A Review of Enabled Application Challenges and Solutions. **International Journal of Advanced Computer Science and Applications**, [*s. l.*], v. 12, n. 3, 2021. ISSN 21565570, 2158107X. DOI 10.14569/IJACSA.2021.0120325. Disponível em: http://thesai.org/Publications/ViewPaper?Volume=12&Issue=3&Code=IJACSA&SerialNo=25. Acesso em: 11 maio 2022.

ARAYA, Juan Ignacio Iturbe; e RIFÀ-POUS, Helena. Anomaly-based cyberattacks detection for smart homes: A systematic literature review. **Internet of Things**, [*s. l.*], v. 22, p. 100792, jul. 2023. ISSN 25426605. DOI 10.1016/j.iot.2023.100792.

ATTOTA, Dinesh Chowdary; MOTHUKURI, Viraaji; PARIZI, Reza M.; e POURIYEH, Seyedamin. An Ensemble Multi-View Federated Learning Intrusion Detection for IoT. **IEEE Access**, [*s. l.*], v. 9, p. 117734–117745, 2021. ISSN 2169-3536. DOI 10.1109/ACCESS.2021.3107337.

BAKHSHI, Zeinab; BALADOR, Ali; e MUSTAFA, Jawad. Industrial IoT security threats and concerns by considering Cisco and Microsoft IoT reference models. *In*: 2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), 2018, Barcelona. **2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)**. Barcelona: IEEE, abr. 2018. p. 173–178. ISBN 978-1-5386-1154-8. DOI 10.1109/WCNCW.2018.8368997. Disponível em: https://ieeexplore.ieee.org/document/8368997/. Acesso em: 31 jul. 2024.

BANG, Ankur O. et al. An IoT Inventory Before Deployment: A Survey on IoT Protocols, Communication Technologies, Vulnerabilities, Attacks, and Future Research Directions. **Computers & Security**, [*s. l.*], p. 102914–102914, 1 set. 2022. DOI: 10.1016/j.cose.2022.102914MAG ID: 4295872857S2ID: b5e9043258c2584526763a53cd50785d4e8faded. DOI 10.1016/j.cose.2022.102914.

BELLINI, Pierfrancesco; NESI, Paolo; e PANTALEO, Gianni. IoT-Enabled Smart Cities: A Review of Concepts, Frameworks and Key Technologies. **Applied Sciences**, [*s. l.*], v. 12, n. 3, p. 1607, 3 fev. 2022. ISSN 2076-3417. DOI 10.3390/app12031607.

BHANDARI, Guru; LYTH, Andreas; SHALAGINOV, Andrii; e GRØNLI, Tor-Morten. Distributed Deep Neural-Network-Based Middleware for Cyber-Attacks Detection in Smart IoT Ecosystem: A Novel Framework and Performance Evaluation Approach. **Electronics**, [*s. l.*], v. 12, n. 2, p. 298, 6 jan. 2023. ISSN 2079-9292. DOI 10.3390/electronics12020298.

BIJONE, Manu. A Survey on Secure Network: Intrusion Detection & Prevention Approaches. **American Journal of Information Systems**, [*s. l.*], 2016. DOI 10.12691/ajis-4-3-2. Disponível em: http://pubs.sciepub.com/ajis/4/3/2/.

BILAL, Muhammad. **A Review of Internet of Things Architecture, Technologies and Analysis Smartphone-based Attacks Against 3D printers**. [*S. l.*]: arXiv, 17 jun. 2017. arXiv:1708.04560 [cs]. Disponível em: http://arxiv.org/abs/1708.04560. Acesso em: 15 jul. 2024.

BUTUN, Ismail; OSTERBERG, Patrik; e SONG, Houbing. Security of the Internet of Things: Vulnerabilities, Attacks, and Countermeasures. **IEEE Communications Surveys & Tutorials**, [*s. l.*], v. 22, n. 1, p. 616–644, 2020. ISSN 1553-877X, 2373-745X. DOI 10.1109/COMST.2019.2953364.

CAI, Jie; LUO, Jiawei; WANG, Shulin; e YANG, Sheng. Feature selection in machine learning: A new perspective. **Neurocomputing**, [*s. l.*], v. 300, p. 70–79, jul. 2018. ISSN 09252312. DOI 10.1016/j.neucom.2017.11.077.

CHAUHAN, Anamika; SINGH, Rajyavardhan; e JAIN, Pratyush. A Literature Review: Intrusion Detection Systems in Internet of Things. **Journal of Physics: Conference Series**, [*s. l.*], v. 1518, n. 1, p. 012040, 1 abr. 2020. ISSN 1742-6588, 1742-6596. DOI 10.1088/1742-6596/1518/1/012040.

CHERUVU, Sunil; KUMAR, Anil; SMITH, Ned; e WHEELER, David M. **Demystifying Internet of Things Security: Successful IoT Device/Edge and Platform Security Deployment**. Berkeley, CA: Apress, 2020. ISBN 978-1-4842-2895-1. DOI 10.1007/978-1-4842-2896-8. Disponível em: http://link.springer.com/10.1007/978-1-4842-2896-8. Acesso em: 17 mar. 2023.

CONNECTIVITY STANDARDS ALLIANCE. **Matter Specification Version 1.0**. [*S. l.*: *s. n.*], 28 set. 2022. Acesso em: 8 mar. 2023.

COOK, Jonathan; REHMAN, Sabih ur; e KHAN, M. Arif. **Security and Privacy for Low Power IoT Devices on 5G and Beyond Networks: Challenges and Future Directions**. [*S. l.*]: arXiv, 2023. DOI 10.48550/ARXIV.2304.00713. Disponível em: https://arxiv.org/abs/2304.00713. Acesso em: 24 jul. 2024.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T.; e BLAIR, G. **Sistemas Distribuídos: Conceitos e Projeto**. 5. ed. Porto Alegre: Bookman Editora, 2013. ISBN 978-85-8260-054-2. Disponível em: https://books.google.com.br/books?id=6WU3AgAAQBAJ.

CURVELLO, André. Apresentando o módulo ESP8266. **Embarcados**. [*S. l.*: *s. n.*], 2015. Disponível em: https://embarcados.com.br/modulo-esp8266/. Acesso em: 22 maio 2023.

DEBAR, Hervé; DACIER, Marc; e WESPI, Andreas. Towards a taxonomy of intrusion-detection systems. **Computer Networks**, [*s. l.*], v. 31, n. 8, p. 805–822, abr. 1999. ISSN 13891286. DOI 10.1016/S1389-1286(98)00017-6.

DHAS, Y. Justin; e JEYANTHI, P. A Review on Internet of Things Protocol and Service Oriented Middleware. *In*: 2019 International Conference on Communication and Signal Processing (ICCSP), 2019, Chennai, India. **2019 International Conference on Communication and Signal Processing (ICCSP)**. Chennai, India: IEEE, abr. 2019. p. 0104–

0108. ISBN 978-1-5386-7595-3. DOI 10.1109/ICCSP.2019.8698088. Disponível em: https://ieeexplore.ieee.org/document/8698088/. Acesso em: 24 fev. 2023.

DUTTA, Abir; e KANT, Shri. Implementation of Cyber Threat Intelligence Platform on Internet of Things (IoT) using TinyML Approach for Deceiving Cyber Invasion. *In*: 2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), 2021, Mauritius, Mauritius. **2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)**. Mauritius, Mauritius: IEEE, 7 out. 2021. p. 1–6. ISBN 978-1-66541-262-9. DOI 10.1109/ICECCME52200.2021.9590959. Disponível em: https://ieeexplore.ieee.org/document/9590959/. Acesso em: 2 abr. 2023.

ERIZA, Aminanto Achmad; e SURVADI, M. T. Literature Review of Machine Learning Models on Intrusion Detection for Internet of Things Attacks. *In*: 2021 International Conference on Electrical, Computer and Energy Technologies (ICECET), 2021, Cape Town, South Africa. **2021 International Conference on Electrical, Computer and Energy Technologies (ICECET)**. Cape Town, South Africa: IEEE, 9 dez. 2021. p. 1–5. ISBN 978-1-66544-231-2. DOI 10.1109/ICECET52533.2021.9698760. Disponível em: https://ieeexplore.ieee.org/document/9698760/. Acesso em: 2 abr. 2023.

ESKANDARI, Mojtaba; JANJUA, Zaffar Haider; VECCHIO, Massimo; e ANTONELLI, Fabio. Passban IDS: An Intelligent Anomaly-Based Intrusion Detection System for IoT Edge Devices. **IEEE Internet of Things Journal**, [*s. l.*], v. 7, n. 8, p. 6882–6897, ago. 2020. ISSN 2327-4662, 2372-2541. DOI 10.1109/JIOT.2020.2970501.

FERRAG, Mohamed Amine; FRIHA, Othmane; HAMOUDA, Djallel; MAGLARAS, Leandros; e JANICKE, Helge. **Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications: Centralized and Federated Learning**. [*S. l.*]: IEEE DataPort, 2022. DOI 10.21227/MBC1-1H68. Disponível em: https://ieee-dataport.org/documents/edge-iiotset-new-comprehensive-realistic-cyber-security-dataset-iot-and-iiot-applications. Acesso em: 14 jul. 2024.

FONTES, Edison Luiz G. **Segurança da informação**. 1ª ed. [*S. l.*]: Editora Saraiva, 2012. ISBN 978-85-02-12218-5.

FRIESKE, Benjamin; e STIELER, Sylvia. The "Semiconductor Crisis" as a Result of the COVID-19 Pandemic and Impacts on the Automotive Industry and Its Supply Chains. **World Electric Vehicle Journal**, [*s. l.*], v. 13, n. 10, p. 189, 16 out. 2022. ISSN 2032-6653. DOI 10.3390/wevj13100189.

GARCÍA-TEODORO, P.; DÍAZ-VERDEJO, J.; MACIÁ-FERNÁNDEZ, G.; e VÁZQUEZ, E. Anomaly-based network intrusion detection: Techniques, systems and challenges. **Computers & Security**, [*s. l.*], v. 28, n. 1–2, p. 18–28, fev. 2009. ISSN 01674048. DOI 10.1016/j.cose.2008.08.003.

HOLBROOK, Luke; e ALAMANIOTIS, Miltiadis. Internet of Things Security Analytics and Solutions with Deep Learning. *In*: 2019. **2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)**. [*S. l.*: *s. n.*], nov. 2019. p. 178–185. ISSN 2375-0197. DOI 10.1109/ICTAI.2019.00033.

IOT ANALYTICS. IoT market segments – Biggest opportunities in industrial manufacturing - IoT Analytics. **IoT market segments – Biggest opportunities in industrial manufacturing - IoT Analytics**. [*S. l.*: *s. n.*], 31 out. 2014. Disponível em: https://iot-analytics.com/iot-market-segments-analysis/. Acesso em: 19 mar. 2023.

IOT ANALYTICS. **Insights Release - What CEOs talked about in Q1 2022**. [*S. l.*: *s. n.*], 2022. Disponível em: https://h9e3r9w2.rocketcdn.me/wp/wp-content/uploads/2022/05/Insights-Release-State-of-IoT-Spring-2022.pdf. Acesso em: 18 mar. 2023.

IOT ANALYTICS. State of IoT 2023: Number of connected IoT devices growing 16% to 16.7 billion globally. **IoT Analytics**. [*S. l.*: *s. n.*], 24 maio 2023. Disponível em: https://iot-analytics.com/number-connected-iot-devices/. Acesso em: 24 jul. 2024.

JANAKIRAMAN, R.; WALDVOGEL, M.; e QI ZHANG. Indra: a peer-to-peer approach to network intrusion detection and prevention. *In*: Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003, Linz, Austria. **WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.** Linz, Austria: IEEE Comput. Soc, 2003. p. 226–231. ISBN 978-0-7695-1963-0. DOI 10.1109/ENABL.2003.1231412. Disponível em: http://ieeexplore.ieee.org/document/1231412/. Acesso em: 3 maio 2023.

KHANAM, Shapla; AHMEDY, Ismail Bin; IDNA IDRIS, Mohd Yamani; JAWARD, Mohamed Hisham; e BIN MD SABRI, Aznul Qalid. A Survey of Security Challenges, Attacks Taxonomy and Advanced Countermeasures in the Internet of Things. **IEEE ACCESS**, [*s. l.*], v. 8, p. 219709–219743, 2020. ISSN 2169-3536. DOI 10.1109/ACCESS.2020.3037359.

KIM, David; e SOLOMON, Michael G. **Fundamentos de Segurança de Sistemas de Informação**. 1ª ed. Rio de Janeiro: LTC, 2014. ISBN 978-85-216-3527-7.

KRISHNA, Ritika Raj; PRIYADARSHINI, Aanchal; JHA, Amitkumar V.; APPASANI, Bhargav; SRINIVASULU, Avireni; e BIZON, Nicu. State-of-the-Art Review on IoT Threats and Attacks: Taxonomy, Challenges and Solutions. **Sustainability**, [*s. l.*], v. 13, n. 16, p. 9463, 23 ago. 2021. ISSN 2071-1050. DOI 10.3390/su13169463.

KUROSE, James F.; e ROSS, Keith W. **Redes de computadores e a internet: uma abordagem top-down**. 8. ed. Porto Alegre, RS: Bookman, 15 jul. 2021. 608 p. ISBN 978-85-8260-558-5.

MAHAJAN, Nitish; CHAUHAN, Amita; KUMAR, Harish; KAUSHAL, Sakshi; e SANGAIAH, Arun Kumar. **A Deep Learning Approach to Detection and Mitigation of Distributed Denial of Service Attacks in High Availability Intelligent Transport Systems**. [*S. l.*: *s. n.*], 2022. v. 27, n. 4, p. 1423–1443. DOI 10.1007/s11036-022-01973-z.

MAZON-OLIVO, Bertha; e PAN, Alberto. Internet of Things: State-of-the-art, Computing Paradigms and Reference Architectures. **IEEE Latin America Transactions**, [*s. l.*], v. 20, n. 1, p. 49–63, jan. 2022. ISSN 1548-0992. DOI 10.1109/TLA.2022.9662173.

MIRSKY, Yisroel; DOITSHMAN, Tomer; ELOVICI, Yuval; e SHABTAI, Asaf. **Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection**. [*S. l.*]: arXiv, 27 maio

2018. arXiv:1802.09089 [cs]. DOI 10.14722/ndss.2018.23204. Disponível em: http://arxiv.org/abs/1802.09089. Acesso em: 10 nov. 2022.

MISHRA, Biswajeeban; e KERTESZ, Attila. The Use of MQTT in M2M and IoT Systems: A Survey. **IEEE Access**, [*s. l.*], v. 8, p. 201071–201086, 2020. ISSN 2169-3536. DOI 10.1109/ACCESS.2020.3035849.

MORAES, Alexandre Fernandes de. **Segurança em Redes - Fundamentos**. 1. ed. São Paulo: Editora Saraiva, 2010. ISBN 978-85-365-2208-1.

MORAES, Alexandre Fernandes de; e HAYASHI, Victor Takashi. **Segurança em IoT: entendendo os riscos e ameaças em IoT**. Rio de Janeiro: Alta Books, 2021. ISBN 978-85-508-1654-8. Disponível em: https://app.minhabiblioteca.com.br/#/books/9788550816548/. Acesso em: 16 set. 2022.

OBJECT MANAGEMENT GROUP. **Data Distribution Service (DDS)**. [*S. l.*: *s. n.*], abr. 2015. Disponível em: http://www.omg.org/spec/DDS/1.4. Acesso em: 20 mar. 2023.

OBJECT MANAGEMENT GROUP. **The Real-time Publish-Subscribe Protocol DDS Interoperability Wire Protocol (DDSI-RTPS™) Specification**. [*S. l.*: *s. n.*], 2022. Disponível em: https://www.omg.org/spec/DDSI-RTPS/2.5/PDF. Acesso em: 23 mar. 2023.

OFFSEC SERVICES LIMITED. What is Kali Linux? | Kali Linux Documentation. **Kali Linux**. [*S. l.*: *s. n.*], 2024. Disponível em: https://www.kali.org/docs/introduction/what-is-kali-linux/. Acesso em: 18 ago. 2024.

PALATTELLA, Maria Rita et al. Standardized Protocol Stack for the Internet of (Important) Things. **IEEE Communications Surveys & Tutorials**, [*s. l.*], v. 15, n. 3, p. 1389–1406, 2013. ISSN 1553-877X. DOI 10.1109/SURV.2012.111412.00158.

PITTMAN, Jason M. **A Comparative Analysis of Port Scanning Tool Efficacy**. [*S. l.*]: arXiv, 20 mar. 2023. arXiv:2303.11282 [cs]. Disponível em: http://arxiv.org/abs/2303.11282. Acesso em: 18 ago. 2024.

PRIYA, V.; THASEEN, I. Sumaiya; GADEKALLU, Thippa Reddy; ABOUDAIF, Mohamed K.; e NASR, Emad Abouel. Robust Attack Detection Approach for IIoT Using Ensemble Classifier. **Computers, Materials & Continua**, [*s. l.*], v. 66, n. 3, p. 2457–2470, 2021. ISSN 1546-2226. DOI 10.32604/cmc.2021.013852.

SETHI, Pallavi; e SARANGI, Smruti R. Internet of Things: Architectures, Protocols, and Applications. **Journal of Electrical and Computer Engineering**, [*s. l.*], v. 2017, p. 1–25, 2017. ISSN 2090-0147, 2090-0155. DOI 10.1155/2017/9324035.

SHARMA, Cheena; e GONDHI, Naveen Kumar. Communication Protocol Stack for Constrained IoT Systems. *In*: 2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU), 2018, Bhimtal. **2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)**. Bhimtal: IEEE, fev. 2018. p. 1–6. ISBN 978-1-5090-6785-5. DOI 10.1109/IoT-SIU.2018.8519904. Disponível em: https://ieeexplore.ieee.org/document/8519904/. Acesso em: 18 fev. 2023.

SHENG, Zhengguo; YANG, Shusen; YU, Yifan; VASILAKOS, Athanasios; MCCANN, Julie; e LEUNG, Kin. A survey on the ietf protocol suite for the internet of things: standards,

challenges, and opportunities. **IEEE Wireless Communications**, [*s. l.*], v. 20, n. 6, p. 91–98, dez. 2013. ISSN 1536-1284. DOI 10.1109/MWC.2013.6704479.

SHUKLA, Prachi. ML-IDS: A machine learning approach to detect wormhole attacks in Internet of Things. **2017 Intelligent Systems Conference (IntelliSys)**, [*s. l.*], p. 234–240, set. 2017. DOI 10.1109/IntelliSys.2017.8324298.

SIEBEL, Thomas M. **Transformação Digital: como sobreviver e prosperar em uma Era de extinção em massa**. 1ª ed. Transformação Digital: Alta Books, 2021. Translated from original Digital Transformation: Survive and Thrive in an Era of Mass Extinction. ISBN 978-85-508-1687-6. Disponível em: https://app.minhabiblioteca.com.br/#/books/9788550816876/. Acesso em: 5 fev. 2023.

SILVA, Henrique Cesar Ferreira; PIETRO, Luca Baron; DARIO, Luís Gustavo Beccheri; MORAES, Eduardo Alves; HERNANDES JR., Paulo R. Galego; e BAREA, Emerson Rogério Alves. Aprendizado de Máquina Aplicado na Classificação de Alertas de Ataques de DoS em Sistemas de Detecção de Intrusão. *In*: XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 2020, Brasil. **Anais Estendidos do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC Estendido 2020)**. Brasil: Sociedade Brasileira de Computação, 7 dez. 2020. p. 241–248. DOI 10.5753/sbrc_estendido.2020.12425. Disponível em: https://sol.sbc.org.br/index.php/sbrc_estendido/article/view/12425. Acesso em: 18 ago. 2024.

SNAPP, Steven R. et al. DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and An Early Prototype. **14th National Computer Security Conference**, [*s. l.*], p. 167–176, 1991.

SWAMY, S. Narasimha; e KOTA, Solomon Raju. An Empirical Study on System Level Aspects of Internet of Things (IoT). **IEEE Access**, [*s. l.*], v. 8, p. 188082–188134, 2020. ISSN 2169-3536. DOI 10.1109/ACCESS.2020.3029847.

TIBURSKI, Ramao Tiago; AMARAL, Leonardo Albernaz; MATOS, Everton DE; AZEVEDO, Dario F. G. DE; e HESSEL, Fabiano. Evaluating the use of TLS and DTLS protocols in IoT middleware systems applied to E-health. *In*: 2017. **2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)**. [*S. l.*: *s. n.*], jan. 2017. p. 480–485. ISSN 2331-9860. DOI 10.1109/CCNC.2017.7983155.

VIEIRA, Moroni N.; OLIVEIRA, Luciana P.; e CARNEIRO, Leonardo. **A Comparative Analysis of Machine Learning Algorithms for Distributed Intrusion Detection in IoT Networks**. [*S. l.*: *s. n.*], 2022. 449 LNNS, p. 249–258. DOI 10.1007/978-3-030-99584-3_22.

WANG, Xiali; e LU, Xiang. A Host-Based Anomaly Detection Framework Using XGBoost and LSTM for IoT Devices. **Wireless Communications and Mobile Computing**, [*s. l.*], v. 2020, p. 1–13, 5 out. 2020. ISSN 1530-8669, 1530-8677. DOI 10.1155/2020/8838571.

WEISER, Mark. The computer for the 21st century. **ACM SIGMOBILE Mobile Computing and Communications Review**, [*s. l.*], v. 3, n. 3, p. 3–11, jul. 1999. ISSN 1559-1662, 1931-1222. DOI 10.1145/329124.329126.

ZEGEYE, Wondimu; JEMAL, Ahamed; e KORNEGAY, Kevin. Connected Smart Home over Matter Protocol. *In*: 2023 IEEE International Conference on Consumer Electronics (ICCE), 2023, Las Vegas, NV, USA. **2023 IEEE International Conference on Consumer Electronics**

(ICCE). Las Vegas, NV, USA: IEEE, 6 jan. 2023. p. 1–7. ISBN 978-1-66549-130-3. DOI 10.1109/ICCE56470.2023.10043520. Disponível em: https://ieeexplore.ieee.org/document/10043520/. Acesso em: 18 fev. 2023.

ZOLOTUKHIN, Mikhail; KOTILAINEN, Pyry; e HAMALAINEN, Timo. Intelligent IDS Chaining for Network Attack Mitigation in SDN. *In*: 2021 17th International Conference on Mobility, Sensing and Networking (MSN), 2021, Exeter, United Kingdom. **2021 17th International Conference on Mobility, Sensing and Networking (MSN)**. Exeter, United Kingdom: IEEE, dez. 2021. p. 786–791. ISBN 978-1-66540-668-0. DOI 10.1109/MSN53354.2021.00123. Disponível em: https://ieeexplore.ieee.org/document/9751466/. Acesso em: 27 mar. 2023.