Alexandre Marques Carrer

Machine Learning aplicada a segurança em Internet das Coisas: detectando anomalias na comunicação e operação

São Paulo, SP

2023

Alexandre Marques Carrer

Machine Learning aplicada a segurança em Internet das Coisas: detectando anomalias na comunicação e operação

Trabalho de conclusão de curso apresentado ao Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro de Computação.

Universidade de São Paulo – USP

Escola Politécnica

Departamento de Engenharia de Computação e Sistemas Digitais (PCS)

Orientador: Profa. Dra. Cíntia Borges Margi

Coorientador: Prof. Dr. Artur Jordão Lima Correia

São Paulo, SP

2023

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

Catalogação-na-publicação

Carrer, Alexandre
Machine Learning aplicada a segurança em Internet das Coisas: detectando anomalias na comunicação e operação / A. Carrer, C. Margi, A. Correia -- São Paulo, 2023.
54 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Sistemas de detecção de intrusão 2.Internet das Coisas 3.Aprendizado de máquina I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t. III.Margi, Cíntia IV.Correia, Artur

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001, projeto FAPESP #2022/07523-8 e Itaú Unibanco S.A. pelo programa PBI do Centro de Ciência de Dados (C2D) da Escola Politécnica da Universidade de São Paulo. Cintia B. Margi é bolsista de produtividade do CNPq #311687/2022-9.

Se eu vi mais longe, foi por estar sobre ombros de gigantes.
Isaac Newton

Resumo

Segurança é um requisito crítico em IoT, a detecção de ataques a rede é importante para garantir sua integridade, confidencialidade e disponibilidade da rede. A literatura apresenta soluções de sistemas de detecção de intrusão (IDS) não alinhados ao paradigma IoT, utilizando protocolos que não seguem padrões internacionais e que utilizam de dados de outros paradigmas de redes de computadores incompatíveis com IoT. Motivado por desenvolver abordagens de segurança em IoT para suprir esta lacuna de pesquisa, este trabalho apresenta a proposta, implementação e avaliação um sistema de detecção de intrusão para o paradigma IoT. O sistema utiliza dados os dados operação dos dispositivos e das métricas da rede que ele participa e a detecção de ataque é baseada em aprendizado de máquina utilizando o *XGBoost Classifier*. Os dados da rede IoT são coletados em simulações próprias no *Contiki Cooja*, com dados de redes sob ataques do tipo Blackhole, Greyhole e Flooding. Realiza-se um tratamento estatístico sob os dados coletados para torná-los utilizáveis pelo modelo de aprendizado supervisionado. Os dados coletados durante a simulação são disponibilizados em formato de um dataset para detecção de intrusão com 15 *features*. Por fim, cria-se o modelo e este tem uma acurácia geral e uma taxa de *recall* acima de 80% para todos os cenários de simulação implementados.

Palavras-chave: Sistemas de detecção de intrusão. Internet das Coisas. Aprendizado de máquina.

Abstract

Security is a critical requirement in IoT; detecting network attacks is important to ensure network integrity, confidentiality, and availability. The literature presents intrusion detection system (IDS) solutions that are not aligned with the IoT paradigm, using protocols that do not follow international standards and leverage data from other computer network paradigms incompatible with IoT. Motivated to develop security approaches in IoT to address this research gap, this work presents the proposal, implementation, and evaluation of an intrusion detection system for the IoT paradigm. The system uses device operation data and network metrics in which it participates, basing attack detection on machine learning using the XGBoost Classifier. IoT network data is collected in proprietary simulations using Contiki Cooja, with data from networks under Blackhole, Greyhole, and Flooding attacks. Statistical treatment is applied to the collected data to make it usable by the supervised learning model. The data collected during the simulation is provided in a dataset format for intrusion detection with 15 features. Finally, the model created demonstrates an overall accuracy and recall rate above 80% for all implemented simulation scenarios.

Keywords: Intrusion Detection System. Internet of Things. Machine Learning.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CoAP	Constrained Application Protocol
CSMA	Carrier Sense Multiple Access
DAO	Destination Advertisement Object
DIO	DODAG Information Object
DODAG	Destination Oriented Directed Acyclic Graph
IA	Inteligência Artificial
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoT	Internet of Things
IPv6	Internet Protocol Version 6
LEACH	Low-energy adaptive clustering hierarchy
LLNs	Low-Power and Lossy Networks
LoRa™	Long Range
MAC	Medium Access Control
ML	Machine Learning
RFCs	Request For Comments
RPL	Routing Protocol for Low-Power and Lossy Networks
RSSF	Redes de Sensores sem Fio
TCP	Transmission Control Protocol
TSCH	Time-Slotted Channel Hopping
UDP	User Datagram Protocol
WSN	Wireless Sensor Networks

Sumário

1 Introdução

1.1 Motivação

No século XXI, temos um mundo gradativamente mais conectado, com tecnologias de automação inteligentes mudando como interagimos com o mundo e como ele interage conosco. Sistemas ciber-físicos são utilizados para controlar mecanismos utilizando algoritmos de computador, a partir da comunicação com sensores físicos como parâmetros. Na indústria 4.0, por exemplo, o controle de uma planta industrial utiliza sensores de monitoramento físico do maquinário e assim ajusta autonomamente sua operação para controlar a produção. Já na cidade, temos *smart grids*, que com milhares de sensores é possível prever e suprir futuras demandas de energia no nível municipal, estadual e federal, além de realizar uma auto-checagem da condição física da infraestrutura (AL-TURJMAN; ABUJUBBEH, 2019)(FIGUEROA; WANG; GIAKOS, 2022).

Para dar suporte a estas estruturas inteligentes, temos uma grande quantidade de sensores e dispositivos IoT que se interligam para gerar informação que é utilizada para controlar os sistemas. Tomar decisões com base em dados maliciosos podem acarretar em grandes perdas, linhas de produção interrompidas, cidades no escuro e prejuízo material de equipamentos. Garantir a integridade e segurança fim-a-fim do sistema contra ataques maliciosos que visam interromper seu funcionamento normal é um requisito crítico para sistemas ciber-físicos.

Autores comumente definem IoT como uma rede de objetos físicos (PATEL; PATEL; SCHOLAR, 2016). A arquitetura, ilustrada pela Figura 1, construída neste paradigma compõe a tradução destes objetos para aplicações utilizáveis. Esta tradução possibilitada pela comunicação entre dispositivos, agregando e transmitindo informação por redes definidas por protocolos e padrões estabelecidos. Esta etapa da arquitetura IoT é sujeita a maior parte de ataques e intrusões ao sistema como todo, com atacantes tentando ferir a integridade, confidencialidade e disponibilidade de seus serviços.

Uma das principais dificuldades de se garantir integridade na totalidade do sistema está na utilização de sensores para captação dos dados que serão utilizados pelo sistema para tomar suas decisões. O uso de dispositivos IoT para o sensoriamento traz consigo problemas de segurança em sua aplicação, onde grande parte dos sensores possuem recursos limitados de memória, processamento e alimentação. Essa limitação é um dos complicadores para a aplicação de algoritmos de autenticação, para a proteção dos dados coletados, e para a existência de uma arquitetura que supre a necessidade de segurança requerida pelo sistema crítico a qual pertence. Em um estudo feito pela *International Society of*

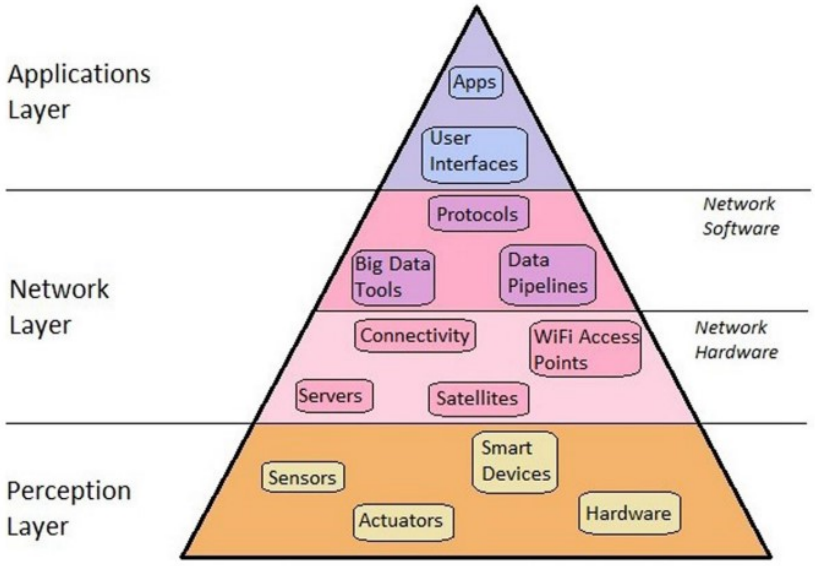


Figura 1 – Diagrama de arquitetura IoT.
Fonte: Tsimenidis, Lagkas e Rantos (2022)

Automation (ISA) com sensores classificados como estado-da-arte utilizados na indústria, os dispositivos analisados não possuíam metade dos requisitos de cibersegurança proposto pela instituição (WEISS et al., 2022).

Grande parte da literatura da área é focada em aplicar estas soluções de IA em Sistemas de detecção de intrusão em redes. Weiss et al. (2022) utiliza um modelo de aprendizado de máquina para contornar os problemas de segurança provindos da falta de recursos de computação em sensores IoT utilizados para controlar uma linha de produção de uma planta hidrometalúrgica. Utilizando escutas direta para a coleta de dados, foi criado sistema *offline* utilizando ML para caracterizar o funcionamento da linha com os dados obtidos antes de serem enviados como pacotes, levantando suspeitas em casos anormais ou de divergência dos dados do sistema principal de controle. O trabalho desenvolvido pôde detectar falhas e ataques do tipo *Man-in-the-middle* em dispositivos IoT que fugiam do seu funcionamento normal. A proposta de aplicação de Aprendizado de Máquina foi eficiente em detectar anomalias que ocorreram entre a borda (sensor) e centro (Interface do operador) da rede. O sistema de detecção de intrusão criado foi construído para trabalhar utilizando métricas da rede dos dispositivos IoT, por isso, falha em detectar outros tipos de intrusão que vão além das métricas de rede. Ataques físicos, injeção de código e sabotagem de sensor trazem consigo consequências nos padrões de funcionamento do dispositivo, como aumento na utilização de processador, maior consumo de energia e maior quantidade de pacotes transmitidos por minuto, podem ser úteis na detecção de intrusão em IoT não são tratados na literatura em uma abordagem centralizada.

Na pesquisa da área existem revisões de literatura que abrangem detecção de intrusão em Internet das Coisas para relatar soluções, definir paradigmas e taxonomia,

analisar arquiteturas de sistemas e elencar tendências na área. Albulayhi et al. (2021) e Lohiya e Thakkar (2020) focam em analisar as soluções de IDS para IoT baseadas em Inteligência artificial, para o primeiro, construídos com datasets empíricos e, para o segundo, discutindo as vulnerabilidades das soluções e desafios na área de segurança em IoT. Ambas revisões apresentam a abordagem de utilização de técnicas de IA como promissoras para utilização em modelos de inferência de IDS. Zarpelão et al. (2017) elencam os principais problemas de soluções analisadas: (i) Não abrangem tipos de ataques e tecnologias IoT implementadas o suficiente; (ii) Definir *tradeoffs* das soluções escolhidas; (iii) Apresentar localização ideal do IDS para maximizar o alcance de detecção; (iv) Consolidar estratégias de validação do IDS.

A literatura aponta a viabilidade de técnicas de aprendizado de máquina para detectar rapidamente anomalias e que estas aplicações aumentam a segurança da aplicação, mas não abrange ou discute as brechas para outros tipos de falhas de segurança que a utilização de Inteligência Artificial pode acarretar na integridade geral da rede. Invasores podem adicionar amostras maliciosas para manipular as predições do modelo ou modificar amostras de dados fazendo com que o modelo erre em sua classificação de integridade do sistema (OPREA; SINGHAL; VASSILEV, 2022), fazendo com que o ataque seja interpretado como o funcionamento correto da rede.

1.2 Objetivo

O objetivo da pesquisa é de propor, implementar e avaliar um sistema capaz de detectar invasões de terceiros em redes de sensores sem fio por predições geradas por um modelo de Machine Learning. Com base em métricas de rede e padrões de utilização da arquitetura do dispositivo, será definido o comportamento padrão da rede para detectar a ocorrência de intrusões até do tipo *day-zero*.

Após a criação do IDS, a pesquisa será pautada em investigar e discutir os tradeoffs da utilização de técnicas de Inteligência Artificial para aplicações de segurança, sobretudo em sistemas automatizados voltados para a detecção de intrusão.

Para a pesquisa atual na área de detecção de intrusão em IoT, normas e padronizações são imprescindíveis para a pesquisa no domínio de redes de dispositivos IoT computacionalmente restritivos. Estes padrões são bem definidos em RFCs criadas pela Internet Engeneering Task Force (IETF) e vão de protocolos a serem utilizados como boas práticas ao desenvolver um sistema IoT. Em artigos recentes, autores propõe redes neurais artificiais complexas para reconhecer ataques à rede para serem executados por dispositivos computacionalmente e energeticamente restritos.

Assim, as duas questões de pesquisa que serão pautadas no trabalho são:

Q1: "A utilização de métricas de operação do dispositivo aliadas às métricas de rede auxiliam na detecção de intrusão em IDS centralizados baseados em anomalia?"

Q2: "Em segurança de IoT, qual a viabilidade das técnicas de Inteligência artificial para detecção de intrusão em redes de dispositivos computacionalmente e energeticamente restritivos?"

A primeira questão de pesquisa é explorativa e está pautada na hipótese de que a coleta, transmissão e análise das métricas de operação dos dispositivos não tem impacto negativo suficiente no desempenho da mesma para que seja desaconselhada de se usar para detectar intrusão. A segunda questão é pautada na hipótese de que técnicas de Inteligência Artificial podem ser úteis na detecção de intrusão, mas todo o sistema deve ser colocado dentro do contexto das restrições dos dispositivos que compõe a rede. Em uma abordagem centralizada, a transmissão das informações necessárias para a inferência não deve afetar significativamente o desempenho de rede. Em abordagens distribuídas, o modelo de inferência deve ser capaz de ser executado nos dispositivos e não deve comprometer o tempo de vida do aparelho. Além de que deve ser garantida a integridade da rede durante todo o processo de treinamento.

1.3 Justificativa

Prevendo e identificando invasões e ataques maliciosos à uma rede de sensores sem fio atrelados a um sistema autônomo, a pesquisa proposta tem o potencial de rapidamente notificar operadores e responsáveis para iniciar um plano de contingência e reparação de danos, reduzindo significativamente os prejuízos. Além disso, identificando os dispositivos comprometidos com o modelo de *Machine Learning* é possível realizar atualizações em software ou hardware nos sensores invadidos, indetectados por outros sistemas de ML.

Outro tópico de relevância é de alavancar a discussão sobre a aplicação de técnicas de Inteligência Artificial e *Machine Learning* em partes críticas de sistemas autônomos e as vulnerabilidades aderidas à utilização destas técnicas. Fomentando a discussão de consequências de se adotar modelos a partir dos resultados quantitativos e qualitativos do sistema de ML proposto.

1.4 Contribuições

Parte das análises e dos resultados obtidos durante a pesquisa foram publicados em um artigo apresentado no Workshop de Trabalhos de Iniciação Científica e de Graduação do XXIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (CARRER; MARGI, 2023). Os anais serão publicados na Biblioteca Digital da Sociedade Brasileira de Computação <https://sol.sbc.org.br/> no ano que vem.

Os dados e códigos produzidos neste trabalho estão disponíveis no *github*: <https://github.com/AleMarquis/ids4lln>

1.5 Organização do Trabalho

Além desta breve introdução, o trabalho está organizado em outros 4 capítulos voltados à explicação das atividades realizadas durante o trabalho de graduação. O capítulo 2 apresenta os fundamentos conceituais importantes ao trabalho e o estado da arte da pesquisa na área. O capítulo 3 apresenta o método da pesquisa, indicando as tarefas, requisitos e tecnologias adotadas no trabalho. O capítulo 4 apresenta o desenvolvimento do trabalho e a arquitetura da solução proposta. O capítulo 5 apresenta as conclusões e considerações finais do documento.

2 Fundamentos e Estado da Arte

Neste capítulo explica-se os fundamentos e aspectos conceituais utilizados e explorados durante o desenvolvimento do trabalho de conclusão de curso. Também será elencado outros trabalhos relacionados que apresentam o estado da arte de Sistemas detectores de intrusão baseados em inteligência artificial aplicados à redes sem fio.

2.1 Aspectos Conceituais

Para o entendimento do projeto, são explicados os aspectos conceituais e fundamentos dos tópicos vitais do trabalho. A seguir, sãpo explicados definições, conceitos, requisitos e desafios dos domínios de Redes de Sensores sem Fio, sistemas de detecção de intrusão e inteligência artificial.

2.1.1 Redes de Dispositivos IoT

O enfoque do trabalho é projetar um IDS voltado para redes de dispositivos IoT com recursos limitados. Dentro da pesquisa na área, a nomenclatura utilizada para se tratar desta rede em questão é Low Power and Lossy Networks (LLNs), ou Redes de Sensores sem Fio (RSSF), ambas semelhantes à pesquisa em IoT (ATZORI; IERA; MORABITO, 2017). Para a rede ser classificada como tal, os dispositivos que a compõe devem ter a capacidade de sentir o ambiente com sensores de estímulos físicos, executar processamento local com um microcontrolador *on-board* e realizar a comunicação entre dispositivos. Estas redes normalmente possuem centenas à milhares de nós (sensores) e um ou mais nó sorvedouro (nós na beirada da rede e que possui acesso à rede externa). É comum que este nó coletor não tenha limitação de energia e processamento e recebe mais responsabilidades dos outros nós da rede.

A taxonomia de redes de sensores sem fio é definida com base nos aspectos da aplicação, a mobilidade dos nós na rede, os hardwares dos sensores que compõe a rede, a hierarquia entre nós, o número e localização dos sorvedouros e o meio físico que será utilizado para a comunicação entre eles.

Pode-se realizar a definição de sua categorização para garantir que a construção do projeto, a escolha dos protocolo e padrões utilizados atenderão aos requisitos. Assim, podemos levantar aspectos de projetos da rede a serem analisados quanto a:

- Mobilidade: Uma rede pode ser composta por nós estáticos, que não se movimentam em relação à rede, ou móveis, que permitem esta movimentação. Também deve ser

levado em consideração se existe um padrão conhecido na mobilidade dos nós da rede;

- Hardware dos nós da rede: Uma rede pode ser classificadas como homogênea, quando todos os nós da rede apresentam hardware e funções semelhantes, ou heterogênea, quando isto não ocorre;
- Hierarquia da rede: Uma rede pode ser classificada como hierárquica, quando a rede é organizada com uma hierarquia definida e existe a diferença de responsabilidade entre os nós, ou plana, quando todos possuem as mesmas funções e permissões na rede;
- Número e localização dos sorvedouros da rede: é necessário realizar o levantamento do número de sorvedouros e sua localização na rede para definir a dinâmica de coleta de dados e ou sua transmissão para fora da rede.

Desafios de implementação

Redes de Sensores sem fio se diferenciam das redes cabeadas de computador não só no meio físico a qual é realizada a comunicação entre arquiteturas, mas também nos desafios para a sua implementação e limitações impostas na rede.

Os sensores que compõe a rede normalmente devem possuir uma vida longa de funcionamento e não devem requerer atenção ou manutenção. Isto faz com que a economia de energia seja um requisito vital no desenvolvimento dos nós da rede e na escolha dos protocolos para manter seu funcionamento. A rede deve ser adaptável e auto configurável para se adaptar às situações onde deve-se lidar com mobilidade ou o desligamento de outros nós. Os protocolos tradicionais de acesso ao meio e de roteamento de pacotes de redes Internet não se aplicam a estes desafios de LLNs. Protocolos atuais de MAC e de roteamento são pensadas em redes cabeadas onde o meio físico está menos sujeito à instabilidade em suas conexões. O TCP, por exemplo estabelece conexões entre os dispositivos para garantir qualidade de serviço (QoS) para suas aplicações, esta aplicação se torna inviável pensando no *overhead* que o protocolo apresenta para a transmissão de pacotes.

Camadas da Aplicação e Pilha de protocolos

A *Internet Engineering Task Force* (IETF) descreve a pilha de protocolos padronizada para *Low-Power and Lossy Networks* (LLNs). Adaptada da pilha de protocolos Internet, a pilha *IETF IoT* apresenta a camada de rede, de transporte e de aplicação e se apoia no padrão IEEE802.15.4 para as demais camadas. A nova descrição da camada de rede apresenta uma camada de adaptação a protocolos de roteamento entre sensores em uma LLN e ao padrão já utilizado de endereços IPv6 do padrão internet.

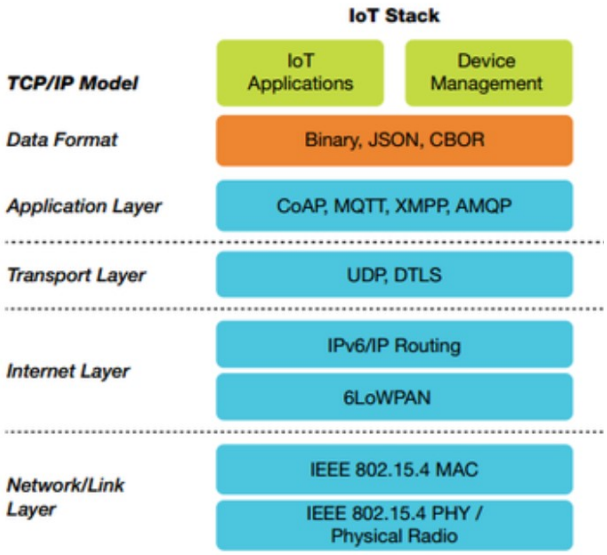


Figura 2 – Pilha de Protocolos padrão IETF para rede de dispositivos IoT em uma LLN

A Camada física é responsável por definir a interface física de conexão entre os dispositivos, seja ele em conectores elétricos em redes cabeadas ou transmissão e recepção de ondas de rádio em redes sem fio. As tecnologias de camada física mais utilizadas em LLNs são: IEEE802.15.4, Bluetooth Low-energy, LoRa. Estes protocolos tem em comum taxas de transmissão mais baixas e apresentam modulação no sinal de rádio para possuir uma eficiência energética maior que outras tecnologias semelhantes.

A Camada de Enlace é responsável por realizar o controle de acesso ao meio, o endereçamento dos quadros e a construção do quadro encaminhado ao vizinho. Estas ações são realizadas pelos nós da rede respeitando o protocolo de acesso ao meio utilizado pela rede. Para a camada de enlace de LLNs é necessário se atentar à eficiência energética utilizada para a transmissão de dados, se atentando à principalmente reduzir o tempo de rádio ativo do transmissor-receptor dos nós da rede (FEENEY; NILSSON, 2001). Dois protocolos serão destacados por seu amplo uso em LLNs:

- CSMA (Carrier Sense Multiple Access): é um *Medium Access Control* (MAC) de acesso múltiplo que sensoreia o meio, verificando se o mesmo está sendo utilizado por outro dispositivo. Se o canal estiver livre, o dispositivo pode transmitir seus dados. Este é o modo de operação base do IEEE 802.15.4. O protocolo por si só não detecta as colisões que ocorreram ou as evita, por isso normalmente é utilizado outras implementações do CSMA. Em IoT, CSMA com prevenção de colisões, CSMA/CA.
- TSCH (Time-Slotted Channel Hopping): um método que divide o tempo em intervalos, os nós na rede são sincronizados para transmitir dados durante seus espaços designados. Além dos espaços de tempo, o protocolo utiliza a mudança de canal para transmitir dados em diferentes frequências de rádio, os diferentes canais dentro do

padrão IEEE 802.15.4.

A camada de rede e seus protocolos são responsáveis pelo encaminhamento de *datagramas* desde a origem até o destino, com a determinação da rota percorrida até alcançá-lo. Nesta camada, além dos já utilizados na pilha Internet, foi necessária a aplicação de protocolos intermediários, pertencentes à camada de adaptação, que permitem o transporte de pacotes IPv6 sobre enlaces de RSSF.

O IPv6 over Low-Power Wireless Personal Area Networks (6loWPAN) é um protocolo da camada de adaptação que permite o transporte de pacotes IPv6 sobre redes que utilizam o padrão IEEE 802.15.4 como meio físico. O protocolo exerce a compressão de cabeçalhos IPv6 e UDP/ICMP e realiza a fragmentação e remontagem de pacotes IPv6. O 6loWPAN Usa enlace IEEE 802.15.4 no modo CSMA/CA

Além dos protocolos de camada de adaptação, os nós de RSSF utilizam de protocolos de roteamento para realizar a determinação de rotas dos pacotes na camada de rede. O roteamento de pacotes em RSSF evoluiu no decorrer dos anos com o desenvolvimento de novos protocolos cada vez mais generalistas em suas aplicações e com mais possibilidades de comunicação entre nós.

No início do desenvolvimento dos protocolos de roteamento e encaminhamento de dados em RSSF, os protocolos e seus algoritmos eram projetados especificamente ter um bom desempenho em sua aplicação. Com sua especificidade, veio uma rigidez do protocolo em outros tipos de aplicação, então cada tipo de RSSF precisava de uma abordagem própria para ser eficiente. Um exemplo de protocolo desenvolvido neste período é o LEACH (HEINZELMAN; CHANDRAKASAN; BALAKRISHNAN, 2000).

Depois disso, as abordagens evoluíram para protocolo de coleta de dados no formato de árvore, onde foi estabelecida a hierarquia nó e sorvedouro. Com isso, foi possibilitado a comunicação genérica de aplicação de coleta de dados para um nó central, na base da árvore, criando os caminhos de menos custo de cada um dos nós e o sorvedouro. Um exemplo de protocolo desenvolvido neste período é o CTP (FONSECA et al., 2006).

Finalmente, os padrões de comunicações evoluíram utilizando outros algoritmos e protocolos que possibilitariam a comunicação entre os nós, não necessariamente estabelecido dentro da hierarquia sorvedouro-nó. Um protocolo muito utilizado nesta aplicação foi o RPL (ALEXANDER et al., 2012), e os outros protocolos anexados à ele que possibilitaram não só a comunicação entre nós, mas também a adaptação do protocolo IPv6 dentro da rede.

O RPL é um protocolo de roteamento desenvolvido especificamente para *Low power and Lossy Networks* (LLN). Ele utiliza uma topologia em árvore DODAG (*Destination Oriented Directed Acyclic Graph*), onde cada nó na rede é representado por um nó na árvore e o caminho da raiz até o nó de destino é utilizado como rota. Os nós são ranqueados

com base na distância em saltos do sorvedouro da rede, com base no rank, é estabelecido uma relação de pai e filho entre nós, até o sorvedouro, a raiz da árvore. O protocolo é projetado para lidar com a dinâmica da rede, permitindo que novos nós sejam realocados, adicionados ou removidos dinamicamente, sem afetar a conectividade e o desempenho geral da rede. Os nós que compõem a rede podem possuir tabela de roteamento ou não, com ou sem armazenamento respectivamente. Com suas características otimizadas para as WSNs, o RPL é amplamente utilizado em aplicações de IoT (Internet das Coisas) e automação industrial.

A Camada de Transporte é responsável pela segmentação dos dados em pacotes menores para mensagens maiores que tamanho máximo permitido pelo meio. A camada também realiza o controle de fluxo para evitar congestionamentos na rede e o controle de erros para garantir que os dados cheguem corretamente ao destino. Dois protocolos de camada de transporte amplamente utilizados em IoT são o User Datagram Protocol (UDP)(POSTEL, 1980) e o Transmission Control Protocol (TCP) (EDDY, 2022). O protocolo de camada de transporte normalmente é escolhido dependendo do protocolo de camada de aplicação utilizado pelo projeto.

A camada de Aplicação suporta as aplicações de rede, e é responsável por fornecer serviços e funcionalidades específicas para as aplicações que a utilizam. Um protocolo amplamente utilizado na camada de aplicação é o CoAP (Constrained Application Protocol) (SHELBY; HARTKE; BORMANN, 2014) , que é projetado para redes de dispositivos com recursos limitados, como redes de sensores sem fio.

CoAP é um protocolo de comunicação de aplicação eficiente para dispositivos com recursos limitados, como sensores e atuadores. Ele é baseado no protocolo HTTP, mas foi adaptado para atender às necessidades específicas dos dispositivos IoT. O protocolo opera utilizando UDP, utilizando mensagens pequenas e simples para reduzir o consumo de recursos dos dispositivos.

Ataques à Redes de sensores sem Fio

As redes de sensores sem fio (RSSF) são sistemas complexos que permitem a coleta de dados em diversos ambientes, como em aplicações de monitoramento ambiental, controle de tráfego e segurança. No entanto, essas redes estão sujeitas a vários tipos de ataques, que podem comprometer a integridade, confidencialidade e disponibilidade da rede. Os ataques às redes de sensores sem fio podem ser classificados diferentes maneiras, com ataques ativos e passivos, físicos e ataques de rede. Cada tipo de ataque pode ter diferentes impactos e requer diferentes contramedidas para proteger a rede (DEOGIRIKAR; VIDHATE, 2017a). Nesta seção indica-se alguns tipos de ataques interessantes para o trabalho e revisão de literatura.

Vale lembrar indicar que, como pressuposto neste trabalho, nós maliciosos são

nós autenticados que foram comprometidos ao longo da operação, e passam a ter comportamento malicioso. Tentativas de invasão na hierarquia da rede não são tomadas em consideração pelo trabalho e o IDS.

Blackhole e *Greyhole*: Estes ataques são classificados como ativos à rede, sua implementação se baseia na absorção de pacotes que deveriam ser roteados para seu destino final. Em uma rede com árvore DODAG, como as que utilizam protocolo RPL, quando um nó malicioso implementa ataques deste tipo, todos os nós-filhos perdem conectividade com o sorvedouro da rede. A diferença de *Blackhole* e *Greyhole* é a relação entre os pacotes retransmitidos e absorvidos, onde no primeiro todos os pacotes são absorvidos e no segundo parte deles são retransmitidos. Ataques do tipo *Greyhole* são mais difíceis de serem detectados por um IDS por este fator.

Flooding: Este ataque é classificado como ativos à rede, sua implementação se baseia em sobrecarregar a rede utilizando *spam* de requisições, mensagens ou pacotes dependendo da camada utilizada para implementar o ataque. Por exemplo, na camada de rede utilizando o RPL como protocolo de roteamento, um nó malicioso ataca a rede a inundando por enviar muitos pacotes de configuração de rede DIO/DAO em um curto intervalo de tempo. Na camada de aplicação, um nó malicioso pode realizar o mesmo realizando muitas requisições de sua aplicação em pouco tempo.

Wormhole: Este ataque é classificado como passivo. sua implementação se baseia em modificar parâmetros do protocolo de roteamento para fazer com que pacotes passam pelo nó para serem retransmitidos, e quando passam, o nó malicioso acessa e analisa os dados que não deveria ter acesso. No protocolo RPL, pode-se implementar o ataque de Wormhole com o nó malicioso realizando uma redução do próprio ranking para receber e analisar pacotes de outros nós. Se passando pelo sorvedouro ou *cluster heads* proximos ao *sink*.

2.1.2 Sistemas de Detecção de Intrusão

Intrusion Detection Systems (IDS) são sistemas construídos para analisar a ocorrência de anomalias em uma rede de computadores, varrendo a rede em busca de comportamentos anômalos ou suspeitos que possam indicar o comprometimento da integridade da rede. IDSs também podem ser desenvolvidos para, além de detectar a intrusão, localizar o dispositivo que apresentou o comportamento e agir sobre ele.

Classificação de um IDS

A classificação de sistemas de detecção de intrusão é feita levando em conta os aspectos de: localização do IDS, o método de detecção e a natureza dos dados utilizados.

Pode-se realizar a definição de sua categorização para garantir que a construção do

projeto atenderão aos requisitos. Assim, podemos levantar aspectos de projetos da rede a serem analisados quanto às seguintes seções deste documento:

Localização do IDS

Os IDSs podem ser classificados de acordo com sua localização na rede em três tipos: centralizados, distribuídos e uma combinação dos dois métodos de maneira híbrida.

1. IDSs centralizados são instalados em um único ponto da rede, geralmente na borda ou no núcleo da rede. Eles coletam dados de toda a rede e os analisam em um único local. Isso pode ser uma vantagem, pois permite que o IDS tenha uma visão abrangente da rede e possa detectar intrusões que possam passar despercebidas por IDSs distribuídos. A principal desvantagem desta abordagem é a necessidade de enviar toda a informação necessária à detecção até o ponto central da rede, podendo gerar problemas de congestionamento e overhead de pacotes.
2. IDSs distribuídos se refere a um sistema onde a detecção de intrusão ocorre em distribuída em diferentes pontos da rede. Isso torna o IDS mais resiliente a ataques e possibilita a melhor definição não só da ocorrência do ataque, mas também da sua localização aproximada. A principal desvantagem desta abordagem é a necessidade de processamento de dados em diferente pontos da rede, fazendo com que a abordagem seja mais custosa computacionalmente em localidades que podem não possuir os recursos necessários.
3. IDSs híbridos combinam as características dos IDSs centralizados e distribuídos. Realizam a maior parte do processamento centralizando em um sistema central, mas também utiliza da arquitetura distribuída para possibilitar a inferência de localidade do ataque e ser mais resiliente à intrusões. Isso combina as vantagens dos dois tipos de IDSs.

Método de detecção

O IDS pode ser classificado conforme a abordagem utilizada para realizar a detecção de intrusão. Sendo classificados em diferentes tipos: (i) Baseados em especificação; (ii) Baseados em assinatura; e (iii) Baseados em anomalia; Juntamente com a possibilidade de mistura dos métodos utilizando uma abordagem (iv) Híbrida.

1. Baseados em especificação: utilizam de limites superiores e inferiores nas métricas coletadas pelo IDS para definir uma faixa onde o sistema está com seu funcionamento padrão. Caso esses limites sejam ultrapassados o IDS relata a intrusão. Esta abordagem exige pouco custo computacional, mas é limitada em detectar ataques que podem contornar os *thresholds* para causar danos à rede.

2. Baseados em assinatura: Utilizam de reconhecimento de padrões de ataques já definidos no banco de dados para a classificação do estado da rede, ou seja, apenas detecta a intrusão que já tiver . São ideais para detectar ataques específicos que foram implementados e utilizados para seu treinamento. Por isso, não são capazes de detectar intrusões além das já predefinidas no treinamento do modelo. IDSs desta categoria possuem alto custo computacional, mas são capazes de resultados mais precisos de ataques previstos em seu banco de dados.

3. Baseados em anomalia: Utilizam do métricas de funcionamento de rede para estabelecer um padrão estatístico de como a rede deve se comportar. Caso a rede apresente um comportamento divergente, ele será considerado anômalo, e portanto uma intrusão. O método é capaz de detectar ataques que não estão previstos pelo administrador do sistema. Mas o método de detecção apresenta uma alta taxa de falso-positivo por considerar perturbações naturais à rede um tipo de anomalia, como interferência entre a comunicação de dispositivos, por exemplo.

Natureza dos dados utilizados

Os IDSs podem ser classificados de acordo com a natureza dos dados que eles utilizam para detectar intrusões; (i) IDSs baseados em rede (NIDS) coletam dados do tráfego de rede para detectar intrusões. Eles monitoram o tráfego de rede em busca de padrões ou atividades suspeitas; (ii) IDSs baseados em host (HIDS) coletam dados de um host individual para detectar intrusões. Eles monitoram os arquivos, processos e logs do host para detectar atividades suspeitas. Também existem as abordagens (iii) híbridas, que utilizam das duas técnicas para realizar a detecção de intrusão da rede a qual está vistoriando.

A figura 3 apresenta um IDS aplicado à uma rede de dispositivos IoT, onde o sistema está sendo processado no sorvedouro da rede. Este IDS conhece o padrão de funcionamento correto da rede e avalia se as métricas da rede fogem deste padrão. Pela classificação apresentada acima, podemos categorizá-la em um NIDS centralizado com detecção baseada em anomalias.

Na revisão sistemática de literatura, Ozkan-Okay et al. (2021) descreve as funções e requisitos importantes de serem levados em conta na construção de um IDS, bem como fatores que determinam sua efetividade em detectar intrusões. Mesmo com as diferentes abordagens de IDS, que variam as funcionalidades com os requisitos de sua aplicação, para desenvolver um sistema detector de intrusão, ele deve realizar as seguintes atividades em sua rotina: (i) Realizar registros, onde o IDS deve salvar as informações localmente ou remotamente em um servidor para poder realizar a comparação e categorização de novos dados com os registrados.(ii) Identificar de eventos que fogem ao padrão dos dados registrados como normais; (iii) Realizar a notificação da ocorrência de anomalias no sistema;

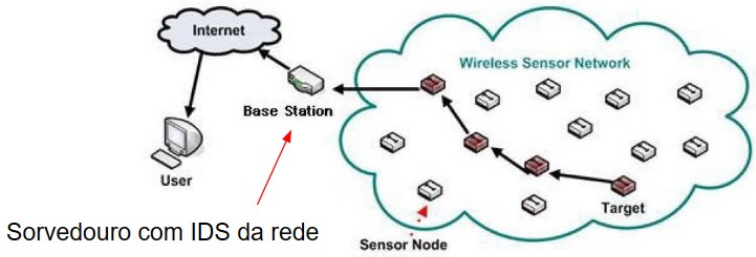


Figura 3 – RSSF com NIDS no nó sorvedouro
Fonte: (RASAL; GUMASTE; DEOKATE, 2015)

e por fim (iv) Gerar relatórios que apresentam o funcionamento do sistema no período analisado.

Um IDS que realiza suas funções de maneira suficiente apresenta os seguintes fatores de medição de desempenho: (i) Confiabilidade e durabilidade do sistema; (ii) Realizar Detecções rápidas e acuradas; (iii) Taxa de falso-positivo minimizadas; (iv) Taxa de detecção Maximizada; (v) Utilização de Software/Hardware minizada; (vi) Habilidade de detectar a localização da intrusão; e (vii) Possibilidade de trabalhar em conjunto com outras tecnologias;

Ozkan-Okay et al. (2021) apresenta que o fator em comum de todos os tipos de IDS é a impossibilidade de garantir uma detecção de anomalia 100% acurada, sendo impossível eliminar a ocorrência de casos Falso-Positivo (FP) e Falso-negativo (FN). Em situações de aperfeiçoamento fino do sistema, na maioria das vezes diminuir um índice impacta no aumento do outro. Assim, os autores da revisão sistemática descrevem que grande parte dos desenvolvedores de IDS preferem reduzir casos FN em detrimento do aumento de ocorrências FP para atender melhor aos requisitos dos sistemas de detecção de Intrusão.

2.1.3 Inteligência Artificial

A Inteligência Artificial (IA) é uma área de estudo que busca desenvolver sistemas computacionais capazes de simular ou superar a capacidade humana de aprender, raciocinar e resolver problemas. De suas inúmeras aplicações, destaca-se a implementação de modelos de IA voltados à área de segurança em computação. Pensando em IDS, existem diferentes abordagens e técnicas para realizar a detecção de intrusão em um sistema. Uma das técnicas consiste na utilização de IA para classificar o conjunto de dados analisados e identificar a presença de anomalias.

Os sistemas de IA são compostos por algoritmos e modelos matemáticos que utilizam dados para aprender e tomar decisões. Existem diferentes técnicas e abordagem de Inteligência Artificial, como Aprendizado de Máquina, a estratégia utilizada neste trabalho, Redes Neurais Artificiais (RNA), Lógica Fuzzy, Máquinas de Vetores de Suporte (SVM),

entre outros. Cada um desses modelos têm suas próprias vantagens e desvantagens e são aplicados de acordo com o problema em questão. A seguir serão destacados abordagens e técnicas específicas de Inteligência artificial para serem explorados em futuras seções do trabalho.

Aprendizado de Máquina

Aprendizado de Máquina, ou *machine learning*, é um dos ramos de inteligência artificial que se baseia no estudo e desenvolvimento de modelos capazes de transformar *inputs* de dados em análises preditivas. Para este propósito, o modelo normalmente é treinando com exemplos de dados.

Podemos categorizar a estratégia de aprendizado de máquina em como são os dados que o modelo usa para seu treinamento. Em um âmbito geral, podemos os classificar em: (i) supervisionado, que usa de dados já rotulados, ou seja, previamente classificados para encontrar padrões e classificar novos dados ainda não rotulados; (ii) não supervisionado, que possui um algoritmo que busca padrões nos dados analisados para realizar sua classificação às cegas; (iii) Semi-supervisionado, que combina os dois tipos anteriores para usar os dados classificados para guiar o modelo na análise dos dados não classificados; e por fim (iv) Aprendizado por Reforço, onde o modelo deve tomar decisões em um ambiente dinâmico utilizando uma recompensa para sistemas com bom desempenho. Neste último, o modelo toma suas decisões maximizando as recompensas. Neste trabalho será utilizado a estratégia de Aprendizado de máquina supervisionado para o desenvolvimento do Sistema de detecção de intrusão. O modelo será treinado com dados rotulados a partir de simulação para processar novas entradas utilizando classificadores.

Avaliação de Modelos Classificadores

Quando realizamos o projeto de um modelo classificador deve-se definir como será medida sua performance. Descrever corretamente como avaliar o desempenho de um classificador auxilia a maximizar sua produção de predições corretos e a validar o funcionamento satisfatório do modelo.

Métricas de avaliação de desempenho permitem avaliar comparativamente modelos que utilizam diferentes estratégias de aprendizagem de máquina. Na literatura, comumente é utilizado métricas estatísticas comparativas das classificações realizadas pelo modelo com o resultado esperado, e correto, da predição. Estes valores estatísticos são calculados com base nos valores categorizados na Matriz de Confusão do resultado do modelo (DEMŠAR, 2006).

Uma matriz de confusão é uma tabela que mostra como um modelo de classificação categoriza um conjunto de dados. Ela é representada por uma tabela quadrada com duas

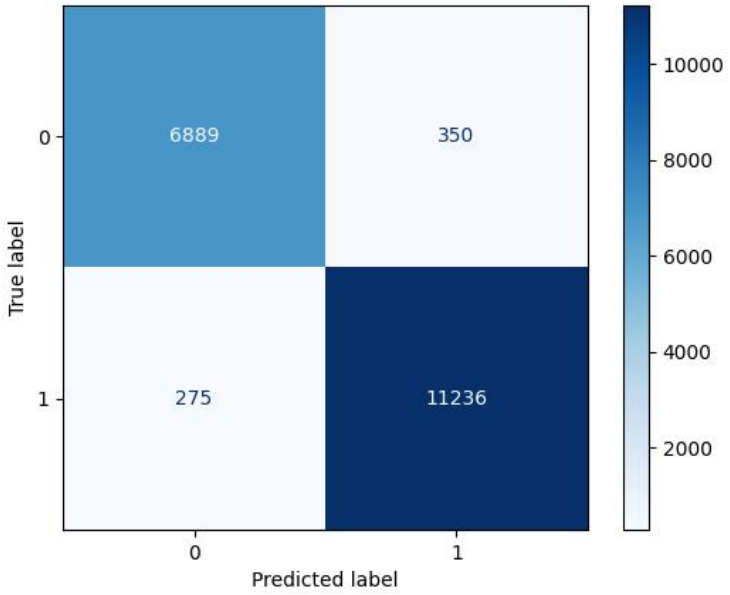


Figura 4 – Exemplo de matriz de confusão
Fonte: autoral

linhas e duas colunas, onde cada linha representa a classe real de um dado e cada coluna representa a classe prevista pelo modelo.

Os quatro valores apresentado na matriz de confusão são: (i) Verdadeiros positivos (VP), dados que são classificados corretamente como pertencentes à classe esperada. (ii) Falsos positivos (FP): dados de outra classe são classificados incorretamente como pertencentes à classe esperada. (iii) Verdadeiros negativos (VN): dados de outra classe são classificados corretamente como pertencentes a outra classe. (iv) Falsos negativos (FN): dados da classe real são classificados incorretamente como pertencentes a outra classe. A matriz de confusão é uma ferramenta frequentemente utilizada para a avaliação de modelos classificadores em IA porque fornece uma visão geral do desempenho do modelo em diferentes tipos de dados. A Figura 13 apresenta um exemplo de matriz de confusão.

Com os valores de Verdadeiro Positivo, Falso Positivo, Falso Negativo e Verdadeiro Negativo expressados na matriz de confusão podemos calcular os índices estatísticos de (i) Acurácia; (ii) Precisão; (iii) *Recall*; e (iv) *F1-Score*.

A **Acurácia** avalia o percentual de classificações corretas do modelo. O índice estatístico é dado pela relação entre o total de acertos (Verdadeiros Positivos e Verdadeiros Negativos) com o total de classificações realizadas pelo modelo. Assim, a métrica é dada pela equação 2.1.

$$Acurácia = \frac{VP + VN}{VP + FP + FN + VN}. \tag{2.1}$$

A **Precisão** é dada pela relação entre as classificações do tipo Verdadeiro Positivo e a soma das classificações positivas. Esta métrica expressa a métrica de confiabilidade do modelo em uma predição positiva. Assim, a métrica é dada pela equação 2.2. Uma baixa precisão nas classificações de um IDS acarreta em um sistema com altos índices de alarme falso, deixando o sistema menos confiante em suas classificações, como na história "O menino que gritava *Lobo!*" (ROULSTON; SMITH, 2004).

$$Precisão = \frac{VP}{VP + FP}. \tag{2.2}$$

O ***Recall***, revocação ou sensibilidade, é a métrica estatística que avalia a capacidade do modelo de classificar corretamente os resultados esperadamente positivos. A métrica é calculada pela relação entre a taxa de Verdadeiro Positivo com a soma das classificações esperadamente positivos. Assim, a métrica é dada pela equação 2.3. Um baixo *recall* em no classificador de um IDS acarreta em uma sensação de falsa segurança do modelo, com o sistema falhando em detectar os ataques quando ocorrem.

$$Precisão = \frac{VP}{VP + FN}. \tag{2.3}$$

Por fim, o ***F1-Score*** avalia as duas métricas anteriores, Precisão e *Recall* em uma média harmônica entre elas. Assim, a média se aproxima da média que possui o menor índice e é interpretado como uma medida de desempenho geral do modelo. Assim, a métrica é dada pela equação 2.4.

$$F1Score = 2 * \frac{1}{\frac{1}{Precisão} + \frac{1}{Recall}} = 2 * \frac{Precisão * Recall}{Precisão + Recall}. \tag{2.4}$$

Para uma aplicação de aprendizagem de máquina em sistemas de detecção de intrusão, deve-se priorizar a confiança na classificação do modelo. Isto pode ser expressado por meio de resultados altos, próximos a 1, nas métricas apresentadas anteriormente. Porém, dentro destes bons resultados, a literatura em IDS baseados em anomalia tende a valorizar a diminuição da taxa de falsos negativos, consequentemente maximizar a métrica de *Recall* como um meio de aumentar a confiabilidade no sistema para reportar o acidente quando ele acontece.

2.2 Revisão de Literatura

Outros sistemas de detecção de intrusão que utilizam técnicas de IA aplicados à redes de sensores sem fio foram desenvolvidos. Esta revisão de literatura realizará a análise destes trabalhos destacando aspectos de técnica de IA utilizada, padrões e protocolos de rede usados, como o sistema foi treinado e sua eficácia em detectar intrusões.

Almomani, Al-Kasasbeh e Al-Akhras (2016) propõe criar um dataset para treinamento de IDS aplicados à redes de sensores sem fio utilizando o protocolo LEACH. Para a coleta de dados maliciosos na rede, a equipe realizou a implementação de nós maliciosos que replicam ataques do tipo Blackhole, Grayhole, Flooding e Scheduling. A rede foi simulada utilizando o simulador de rede NS-2, onde seus parâmetros foram documentados e disponibilizados como o Dataset WSN-DS. Como prova de conceito, foi criado um NIDS baseado em Deep Learning para analisar Redes baseadas no protocolo LEACH. A rede neural artificial obteve acurácia de classificação de 92.8%, 75.6%, 99.4%, 92.2% para cada tipo de ataque citado anteriormente, além de 99.8% na categorização de situações normais de rede. O artigo apresenta uma solução utilizando um protocolo de roteamento, o LEACH, ultrapassado por protocolos atuais adotados na pesquisa em IoT.

Alrashdi et al. (2019) descreve um sistema chamado AD-IoT, implementando um NIDS baseado em aprendizado de máquina para detectar anomalias em ciberataques em dispositivos IoT em cidades inteligentes. O modelo descrito no artigo utilizou de aprendizado supervisionado com o algoritmo *Random Forrest*. O modelo foi treinado com o dataset UNSW-NB15, um dataset criado para o treinamento de NIDS composto de pacotes coletados de uma rede cabeada. Os autores extrapolaram o escopo do dataset, de uma rede de computadores, para treinar o classificador, para uma rede de dispositivos IoT. Sem focar um tipo de ataque específico, o sistema detecta anomalias na comunicação dos dispositivos analisando os pacotes da rede a qual ele está inserido e retorna a ocorrência ou não de comportamento anômalo com uma acurácia de 99.34%.

Almiani et al. (2020) propõe um sistema de detecção de intrusão baseado em uma rede neural recorrente profunda (DRNN) para redes de sensores sem fio IoT. A DRNN é treinada com o dataset NSL-KDD adaptado pelos autores. O dataset inclui pacotes de rede simulando tráfego da Internet utilizando *tcpdump* e pacotes gerados por ataques do tipo DoS, DDoS, MITM e Worm. A DRNN é capaz de detectar a ocorrência de ataques com uma acurácia média de 92.42%. O artigo apresenta a solução de DRNN como adequadas para detectar ataques e que ocorrem ao longo do tempo em tempo real, como ataques DoS e DDoS por serem capazes de aprender padrões em séries temporais.

Moustafa, Turnbull e Choo (2018) propõe um sistema de detecção de intrusão baseado em um conjunto de técnicas de aprendizado de máquina para redes de sensores sem fio IoT. O sistema é composto por três módulos: (i) Extração de recursos: gerar variáveis para entrada do modelo utilizando estatísticas de fluxos de rede do dataset. (ii) Modelos de classificação: Realizando a classificação dos dados estatísticos como normais ou maliciosos utilizando três algoritmos de aprendizado de máquina: Random Forest, SVM e KNN. (iii) Agrupam as classificações para gerar uma decisão final sobre a detecção de intrusão na rede. Este trabalho usa os datasets USNW-NB15 e NIMS botnet para o treinamento e validação dos dados. Os datasets correspondem à trafego de rede HTTP,

Tabela 1 – Resumo comparativo de literatura relacionada em termos de Técnica de Inteligência Artificial usada, Configuração da Rede vistoriada, Dataset utilizado e Acurácia do Modelo

Trabalho	**Técnica de IA**	**Configuração de Rede**	**Dataset**	**Acurácia**
WSN-DS	ANN	WSN - LEACH	WSN-DS	Blackhole - 92.8%, Greyhole - 75.6%, Flooding - 99.4%, Scheduling - 92.2%
AD-IoT	ML - Random Forest	IETF IoT	UNSW-NB15	Anomalia - 99,43%
DRNN	DRNN	Tráfego da Internet	NSL-KDD	DoS - 98.27% Probe - 97.35% R2L - 64.93% U2R - 77.25%
Ensemble IDS	SVM, Random Forest, KNN	HTTP, DNS, MQTT	USNW-NB15 e NIMS botnet	Taxa de detecção HTTP - 95,92% DNS - 95,02% MQTT não disponibilizado

MQTT e DNS. A acurácia de detecção de intrusão foi de acima de 95% nos dois tráfegos.

Os trabalhos apresentados na seção de revisão de literatura apontam a uma lacuna de pesquisa definida. Onde a pesquisa da área não utiliza de protocolos seguindo a padronização internacional definida pela IETF (TSCHOFENIG et al., 2015), e muitas vezes utilizam dados de outras abordagens de redes incompatíveis com o paradigma IoT. Todas as referências foram comparadas em termos de Técnica de IA, Configuração de Rede, Dataset e Acurácia do modelo. Esta comparação está apresentada na Tabela 1.

3 Método do Trabalho

Esta pesquisa consiste na proposição, implementação e avaliação de um sistema destinado a detectar invasões em redes de dispositivos IoT através de predições advindas de um modelo de Machine Learning. O intuito é estabelecer o comportamento padrão dessas redes, valendo-se de métricas de rede e padrões específicos da arquitetura dos dispositivos, permitindo assim a detecção de intrusões em tempo real. O método de pesquisa é definido pela hipótese de que a união de métricas de rede e de operação dos dispositivos auxilia na detecção de intrusão em IDSs baseados em aprendizado de máquina. As tarefas executadas no trabalho são listadas abaixo.

3.1 Tarefas

A pesquisa começa com a determinação do escopo IoT a ser estudado para elencar os parâmetros usados para a análise do sistema proposto, definir tipos de dispositivos IoT e a arquitetura usada para a construção da rede observada pelo sistema detector para a coleta de dados experimentais. *Datasets* disponíveis na literatura também são utilizados para auxiliar no desenvolvimento e treinamento do modelo.

Após a definição do escopo, a próxima etapa envolve a replicação de literatura de caráter semelhante ao projeto. O artigo escolhido foi o Alrashdi et al. (2019), um trabalho de conferência que propõe criar um NIDS (*Network-based Intrusion Detection System*) utilizando Machine Learning.

Depois, é realizada a simulação de uma rede de dispositivos IoT visando a elaboração de um ambiente para realizar os experimentos. Com isso, é possível determinar o funcionamento normal da rede e realizar um conjunto de diferentes simulações de invasões à dispositivos para desenvolver o sistema detector. A rede conta com 64 nós em uma topologia de malha utilizando o padrão IEEE802.15.4 como camada física. A pilha de protocolos é composta pelo padrão IEEE802.15.4 e a padronização IETF IoT até a camada de transporte (PALATTELLA et al., 2013).

Para a simulação do ataque à rede, é realizada a implementação de diferentes ataques à rede simulada. Os ataques de *Blackhole* e *Greyhole* são desenvolvidos utilizando manipulações à hierarquia e *ranking* dos nós na rede criada pela árvore DODAG no protocolo RPL. Também será simulado ataques do tipo *Flooding* pela camada de rede, com o envio incessante de pacotes DIO/DAO de configuração da árvore DODAG pelos nós maliciosos, e pela camada de aplicação, com o envio incessante de requisições UDP pelos nós maliciosos.

Concomitante à rede desenvolvida, é estudado a aplicação de Machine Learning no processo de avaliação do estado da rede visando determinar o método mais indicado para a solução (aprendizado supervisionado, não supervisionado, por reforço), bem como os modelos e bibliotecas de classificação por IA preferíveis para a pesquisa.

Com o modelo de classificação treinado, o sistema é adaptado para funcionar em redes e aplicações reais de sistemas ciber-físicos. E com os novos padrões de funcionamento, replicar testes do sistema, para determinar viabilidade, escalabilidade, precisão e tempo de detecção de anomalias da solução. Com as diferentes redes, podem ser adotadas diferentes arquiteturas de rede para as simulações da pesquisa (redes centralizadas, distribuídas, *fog computing*, etc.).

Depois é realizada a investigação e discussão dos resultados do sistema, com um enfoque nos problemas de segurança amenizados pela solução detectora desenvolvida. E, por fim, é discutido os *tradeoffs* de se utilizar um modelo de aprendizado de máquina para detectar falhas de segurança, os *tradeoffs* de segurança a maior sobrecarga de pacotes sendo enviados na rede.

Os testes para validação do modelo são realizados de duas formas: (i) Validando o modelo com a divisão de seu dataset em treinamento, teste e validação e obtendo seus resultados; e (ii) Realizando a detecção de uma intrusão corretamente analisando as predições do modelo conforme o tempo de simulação. Com resultados as métricas de avaliação do modelo indicando o desempenho, o modelo identificando corretamente a ocorrência de ataques e o sistema atendendo aos requisitos propostos na Seção 3.2, o projeto e implementação do modelo está validado.

3.2 Requisitos

Abaixo são elencados os requisitos do trabalho, o sistema de detecção de intrusões baseado em *Machine Learning* aplicado à uma rede de dispositivos IoT. Eles são destacados iniciando-se pelos requisitos da rede base, da implementação dos ataques, passando pelos requisitos do Sistema de Detecção de Intrusão implementado, até o método de Machine Learning utilizado como base à detecção do IDS.

3.2.1 Dispositivos IoT

Os requisitos da rede IoT são voltados para a taxonomia e definição da rede a qual se deseja desenvolver o sistema de detecção de intrusão. O projeto foi desenvolvido para preencher um *gap* na literatura de NIDS para IOT. Literatura que apresenta soluções de sistemas detectores de anomalias voltados para IoT utilizando protocolos ultrapassados e não alinhados com a padronização da IETF (TSCHOFENIG et al., 2015), datasets que

não atendem às peculiaridades do paradigma IoT e resultados obtidos não replicáveis em redes reais.

Outro fator elencado pelos requisitos de rede são os tipos de ataques e sua implementação na rede. Para isso será utilizado o emulador de rede Contiki Cooja, desenvolvido para emulação de dispositivos e redes de sensores sem fio. As implementações dos ataques são realizadas de maneira a representar os tipos mais comuns de invasões maliciosas à RSSF descritos na literatura. Os nós maliciosos estarão autenticados e farão parte da rede como pressuposto, tentativas de invasão na hierarquia da rede não são tomadas em consideração pelo trabalho e o IDS.

Com isso a rede utiliza a seguinte pilha de protocolos, escolhidos por utilizar protocolos modernos e serem amplamente utilizadas na academia e indústria: Camada física: IEEE802.15.4; Camada de Enlace: Carrier Sense Multiple Access (CSMA); Camada de Rede: IPv6 over Low Power Personal Area Network (6loWPAN) + Routing Protocol for Low-Power and Lossy Networks (RPL); Camada de Transporte: User Datagram Protocol (UDP).

3.2.2 Sistema de Detecção de Intrusão

Além dos requisitos descritos na seção 2.1.2, levantamos outros requisitos do IDS implementado neste trabalho. Estes são Detecção de Anomalias, Escalabilidade, Classificação em *Real-time* e foco em eficiência energética. Os requisitos são descritos e explicados abaixo:

- Detecção de anomalias: O sistema deve ser capaz de detectar anomalias ou comportamentos anormais na rede de sensores, como a presença de dispositivos não autorizados e o uso de rede, processador e energia anômalos.
- Escalabilidade: O sistema deve ser capaz de lidar com uma grande quantidade de sensores sem fio, também capaz de se adequar ao crescimento do tamanho da rede junto com a mesma.
- Classificação em *Real-Time*: O sistema deve avaliar a rede de sensores sem fio a qual monitora em tempo real e detectar a invasão enquanto ela ainda ocorre.
- Eficiência energética: Os sensores sem fio possuem fonte de energia limitada. Mesmo com a transmissão dos parâmetros de operação e de rede, a aplicação do sistema deve minimizar o impacto de consumo de energia na rede de sensores sem fio que está atrelada.

3.2.3 Aprendizado de Máquina Supervisionado

O Modelo de Aprendizado de máquina supervisionado é treinado com os dados coletados a partir da emulação da rede. O modelo deve ser adaptável a diferentes topologias de redes que utilizam protocolos semelhantes à rede original. O requisito de Segurança durante o treinamento será discutido a fundo posteriormente na construção da tese de mestrado, elencando o *trade-off* de se utilizar técnicas de IA em IDS. Os requisitos são descritos e explicados abaixo:

- Adaptabilidade: O modelo deve ser capaz de se adaptar a diferentes aplicações e topologias de redes, bem como diferentes dispositivos que as compõe. Para mudanças na rede, deve ser necessário retreinar o modelo.
- Minimização de Falsos Negativos: O sistema deve priorizar a diminuição de alertas do tipo FN (falso negativo) para garantir a detecção em caso de anomalias na rede.
- Segurança em seu treinamento: Para o treinamento do sistema, deve estar certificado de que a rede está íntegra e que os parâmetros classificados como correto não estejam enviesados (OPREA; SINGHAL; VASSILEV, 2022).

3.3 Tecnologias Utilizadas

Durante o desenvolvimento do protótipo, são utilizadas tecnologias e ferramentas adequadas. Essa seção apresenta as tecnologias e ferramentas mais relevantes utilizadas no projeto. Elas abordam ferramentas para simulação de rede, coleta e tratamento de dados, até a criação do modelo classificador. Vale notar que no decorrer do projeto, todas as simulações de rede, coleta de dados, treinamento e previsões do modelo foram conduzidas em um computador equipado com o sistema operacional Ubuntu Desktop 22.04 canônico. O desktop possui um processador Intel i7-13700, 32GB de RAM e um disco de 1TB.

Contiki-NG

O Contiki-NG (OIKONOMOU et al., 2022) é um sistema operacional open-source para dispositivos IoT de baixa potência. Ele é baseado no Contiki-OS (CONTIKI-OS, 2018), um sistema operacional desenvolvido pelo projeto Contiki, com a alteração do código fonte para simplificar funcionalidades providas pelo SO. A proposta do Sistema operacional é ser eficiente em termos de custo computacional e de memória para dispositivos IoT com recursos limitados. Ele é baseado em um *microkernel*, o que significa que é modular e pode ser personalizado para atender às necessidades específicas de um dispositivo utilizando a adição e remoção de módulos do SO no *cross-compiling* para dispositivos IoT.

O sistema operacional inclui uma série de recursos e ferramentas que o tornam uma especializado para o desenvolvimento de sistemas IoT. Esses recursos incluem: (i) suporte para diferentes protocolos utilizados em LLNs, como 6LoWPAN e CoAP; (ii) Suporte para uma diferentes hardwares de sensores e atuadores utilizados em IoT; (iii) Suporte para criptografia e segurança; (iv) Suporte para emulação com o Cooja.

Cooja

O Cooja (CONTIKI-OS, 2019) é um software criado pelo Projeto Contiki que é capaz de realizar a emulação de redes de dispositivos IoT. O software é utilizado principalmente como uma ferramenta de desenvolvimento de redes de sensores sem fio que utilizam o sistema operacional. Nele é possível simular o comportamento real de redes de sensores sem a necessidade de dispositivos físicos para tal.

O software se baseia em um ambiente de simulação composto de: (i) um grid tridimensional, onde é possível realizar a inserção de dispositivos emulados, (ii) ferramentas de controle de simulação, onde é possível controlar seu funcionamento com funções de parar, continuar, acelerar e desacelerar a simulação; (iii) apresentação das saídas dos nós, com portas seriais, LEDs e a análise dos pacotes trocados dos dispositivos.

Utilizar um simulador no trabalho é interessante pela flexibilidade e facilidade de realizar experimentos com diferentes cenários e configurações diminuindo o tempo necessário para completá-las, além de não necessitar a utilização de hardware real.

Plataformas de Hardware

O Tmote Sky (MOTEIV, 2006), também conhecido como TelosB é um módulo de sensor sem fio desenvolvido pela Moteiv Corporation. Ele é baseado em um microcontrolador TI MSP430F1611 com clock de 8 MHz e um chip de rádio CC2420 compatível com o padrão IEEE 802.15.4. O microcontrolador possui 10 KB de memória RAM e 48 KB de memória flash. Além do hardware computacional, a plataforma possui dois sensores de luz e sensores de temperatura e umidade integrados. Ele é uma plataforma de baixo consumo de energia, alta confiabilidade e facilidade de desenvolvimento para aplicações de redes de sensores sem fio, cujo projeto é aberto.

O Zolertia Z1 (ZOLERTIA, 2013) é uma plataforma compatível e semelhante à família de motes Tmote™, também sendo baseados na arquitetura MSP430+CC2420 da Texas Instruments. A principal diferença das plataformas são os melhorias de hardware por parte do Z1 que oferecem cerca de 2x de desempenho em aspectos de capacidade computacional e energia. O Z1 Zolertia possui um microcontrolador MSP430F2617 de 16 bits e baixo consumo de energia, que opera com clock 16 MHz. O microcontrolador possui 8 KB de memória RAM e 92 KB de memória flash. Ele também possui um transceptor de

Plataformas	CPU [mA]	LPM [uA]	Rx [mA]	Tx [mA]
Zolertia Z1	10	23	18.8	17.4
Sky Mote	2.4	21	19.7	17.4

Tabela 2 – Consumo de corrente das plataformas de Hardware por operação. CPU indica o consumo de corrente (CC) em modo CPU. LPM indica o CC do dispositivo em *Standby* (Low Power Mode). Rx indica o CC do rádio em modo de receptor. Tx indica o CC do rádio em modo de transmissor. Fonte: Datasheets (MOTEIV, 2006) (ZOLERTIA, 2010)

rádio CC2420 de 2,4 GHz, que é compatível com os protocolos IEEE 802.15.4, 6LowPAN e ZigBee™. Além do hardware computacional, a plataforma possui um acelerômetro digital de três eixos, um sensor de temperatura digital.

O poder computacional e o consumo de energia são onde as duas plataformas diferenciam-se. O maior poder computacional acaba acarretando em um maior consumo de energia por parte do Z1. A maior diferença entre eles é expressa no consumo com a CPU em funcionamento, com a maior memória flash, taxa de clock, e o maior consumo dos sensores embarcados. A Tabela 2 apresenta o consumo de corrente médio de seções das duas plataformas em diferentes modos de operação.

Algoritmo de Aprendizado de Máquina

O *XGBoost*, abreviação de *eXtreme Gradient Boosting*, é um algoritmo de aprendizado de máquina que se baseia em árvores de decisão e utiliza uma estrutura de *Gradient Boosting*. As árvores de decisão são uma representação visual de um conjunto de regras de decisão que ajudam a tomar decisões com base em diferentes métricas. Cada nó interno da árvore representa uma decisão a ser tomada, enquanto os nós folha representam os resultados ou as ações a serem executadas, no caso de um modelo classificador, a classificação de um conjunto de dados. O *Gradient Boosting* combina os resultados de uma grande quantidade de classificadores, tipicamente árvores de decisão, que se combinam para formar um comitê de decisão.

O *Gradient Boosting* funciona criando árvores de decisão sequenciais onde cada árvore se concentra em corrigir os erros do modelo anterior com um estimador, a diferença entre o valor previsto do real, cada novo estimador tenta corrigir os erros de seu antecessor. Com os valores estimados dos erros residuais é construído um gradiente, e cada árvore subsequente construída tenta minimizar o gradiente da anterior, melhorando a acurácia do das árvores.

Em problemas de previsão que envolvem dados estruturados, algoritmos baseados em árvores de decisão são considerados como mais indicados para serem utilizados. Assim, o *XGBoost* foi escolhido como modelo classificador para este projeto.

4 Projeto e Especificação

4.1 Arquitetura Geral da Solução

Para concluir o objetivo do projeto, de propor, implementar e avaliar um IDS para IoT, e solucionar as duas questões de pesquisa, a arquitetura da solução foi concebida considerando o ambiente de simulação e o sistema de detecção de eventos. A arquitetura geral contém 4 módulos: (i) a implementação dos ataques e simulação de rede, (ii) a coleta de dados do simulador, (iii) o treinamento estatístico dos dados coletados, e (iv) a criação do sistema classificador. O fluxo de dados da solução apresenta a relação de transmissão de dados entre os módulos, apresentado na Figura 5.

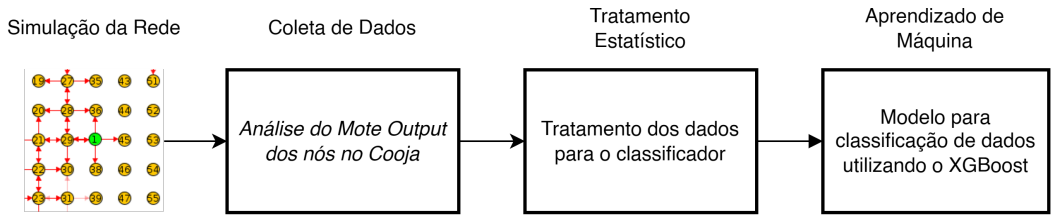


Figura 5 – Fluxo de dados da solução implementada.

A solução proposta foi projetada seguindo o método do projeto, os requisitos levantados e as tecnologias a serem empregadas. A implementações dos ataques são realizadas no sistema operacional Contiki-NG (OIKONOMOU et al., 2022) e as simulações rodadas no *Cooja* (CONTIKI-OS, 2019). A coleta de dados se dá pela interface serial coletada pela ferramenta *Mote Output*. Os dados passam por um janelamento e um tratamento estatísticos para criação de métricas para o modelo classificador. Por fim, é utilizado o *XGBoostClassifier* (CHEN; GUESTRIN, 2016) como modelo classificador e motor de inferência do Sistema de Detecção de Intrusão do projeto. Assim, a arquitetura geral do projeto é ilustrada pela Figura 6.

4.2 Simulação de Rede

Uma simulação do Contiki Cooja é composta pelos nós da rede e por configurações de emulação, ambas descritas em um arquivo de Configuração de Simulação Cooja (arquivo de extensão .CSC). Além de programar o código fonte dos dispositivos e os dispor em um espaço tridimensional, é possível definir alcance e taxa de transmissão e recepção do rádio de todos os nós. Também é possível interagir com os sensores de diferentes modos, apertar botões, analisar conteúdo de memória, acender LEDs. Com isso, modificando

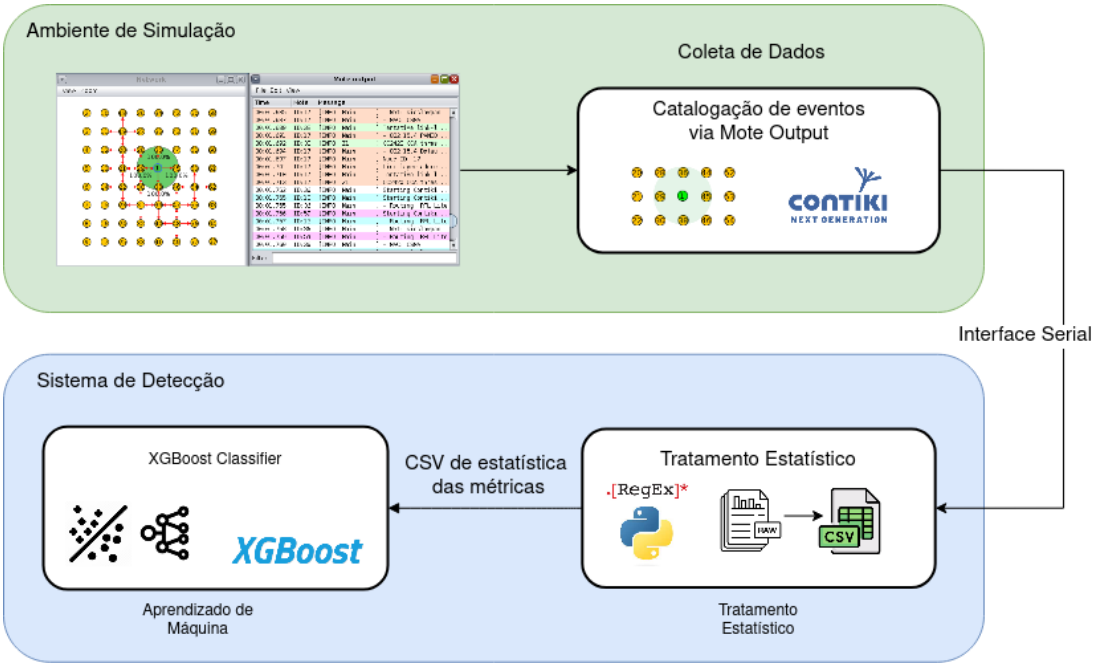


Figura 6 – Diagrama da Arquitetura geral da solução.

Parâmetros	Configuração
Semente da simulação	123456
Camada Física	IEEE802.15.4
Camada de Enlace	ContikiMAC
Camada de Rede	6loWPAN+RPL-lite
Camada de Transmissão	UDP
Topologia	Grid 8x8
Posição do Servidor	Central
Distância dos nós	45 m
Alcance de Transmissão	50 m
Número de Servidores	1
Número de Clientes	63
Intervalo entre requisições	1 min
Raiz da rede	Servidor
Plataforma do servidor	Zolertia Z1
Plataforma do cliente	Zolertia Z1

Tabela 3 – Configuração dos parâmetros da simulação realizada

código fonte compilado dos nós é possível usar uma simulação para diferentes cenários de estudo do comportamento de rede. A configuração das simulações é apresentado na Tabela 3. O arquivo CSC das simulações realizadas no trabalho pode ser encontrado no Github: <https://github.com/AleMarquis/ids4lln>

Cada dispositivo pode ser programado para imprimir informações na porta serial conforme seu funcionamento. A ferramenta *Mote Output*, nativa do *Cooja*, compila as informações de todos os nós e as compila em um documento de *Logs* da simulação.

Utilizar todos os *logs* disponíveis na ferramenta não é interessante para o projeto pela impossibilidade de aplicar esta *clarevidência* em redes locais. O *Mote Output* possui onisciência de tudo o que ocorre nos nós da rede, o que nem sempre é o caso em aplicações IoT. O sistema terá apenas acesso ao relatório de informações do servidor da rede. Para usar informações de outros nós, elas devem ser enviadas até o servidor central para o sistema processá-los. O Mote output é a interface serial de mensagens dos dispositivos emulados pelo Cooja. A Figura 7 apresenta a interface do software com as ferramentas Network e Mote Output.

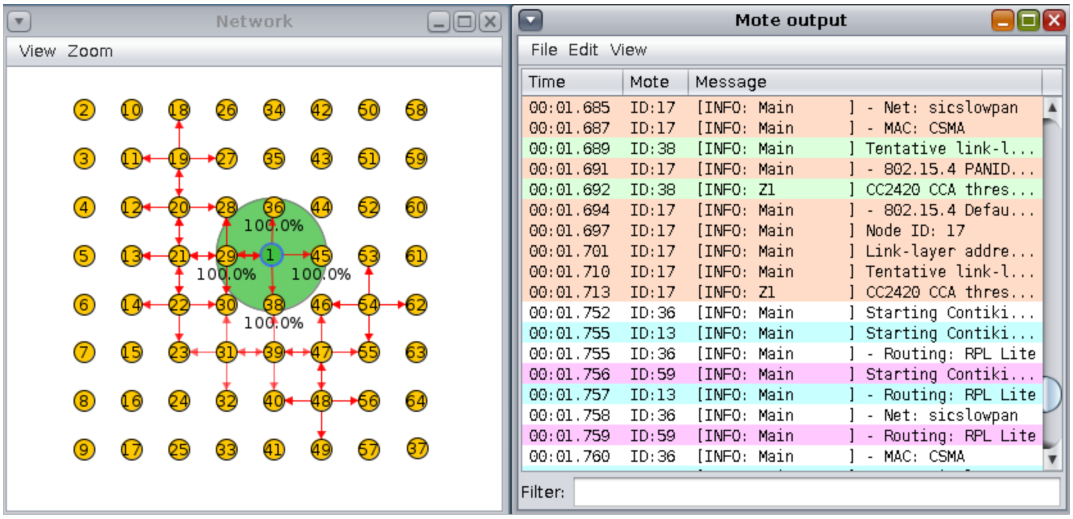


Figura 7 – Interface do software Contiki Cooja. À esquerda é apresentada a topologia da simulação e à direita o mote output, a interface serial de mensagens dos dispositivos emulados

As simulações realizadas no Cooja para coleta de dados foram baseadas na implementação padrão do RPL-UDP (CONTIKI-NG, 2022a) da documentação do Contiki-NG. Esta simulação está presente no projeto Contiki para desenvolvedores usarem as configurações como base de implementações dos protocolos RPL (BARTHEL et al., 2012) (CONTIKI-NG, 2022b) e UDP (POSTEL, 1980). O código fonte foi modificado para funcionar como coleta de dados por um servidor central utilizando dois tipos de nós: Clientes e Servidores. O código dos nós clientes se baseiam na transmissão periódica de métricas de temperatura e pressão ao servidor. A função do código do servidor é receber as mensagens, enviá-las para a porta serial, estabelecer a árvore de roteamento da rede e participar dela como nó raiz.

Esta simulação é composta por 64 nós: 1 servidor e 63 clientes em uma topologia de *grid*, onde os nós só conseguem se comunicar com seus vizinhos imediatos, ou seja, com o nós imediatamente em cima, abaixo, e aos lados, conforme indica a Figura 8. Os clientes transmitem os índices de temperatura e pressão de 1 em 1 minuto. Em sequência desta mensagem, os sensores são configurados para transmitir outra suas métricas de operação: tempo de CPU e energia gasta pelo nó.

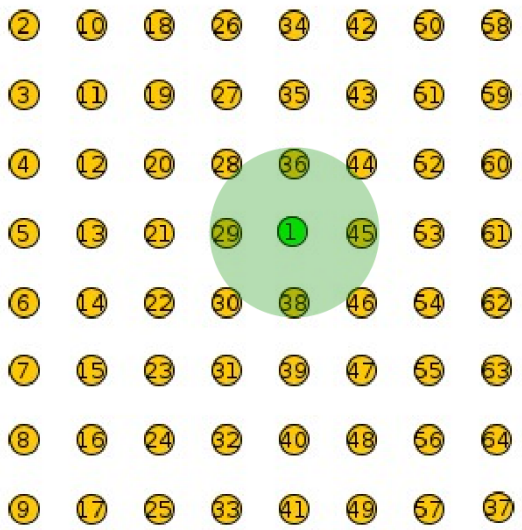


Figura 8 – Topologia da rede simulada. A Figura apresenta o nó servidor (nó 1, em verde) e os nós clientes (nós em laranja). O alcance de comunicação entre nós é representado pelo círculo em verde claro

No Contiki Cooja é possível escolher a plataforma e o processador implementado na rede, o Cooja emula o dispositivo escolhido, imitando seu comportamento e suas limitações. Nos experimentos realizados neste trabalho a plataforma de hardware utilizada foi a *Zolertia Z1* por ser uma plataforma suportada pelo Contiki-NG e por melhor atender os requisitos de memória e processamento para os processos implementados.

No início do projeto a proposta inicial era realizar as simulações de rede com uma pilha de protocolos completa, utilizando o padrão IEEE802.15.4, 6loWPAN, RPL, UDP e o CoAP como camada de aplicação. Duas restrições, no emulador e no hardware disponível fizeram com que as simulações não implementassem a camada de aplicação. A implementação padrão do protocolo CoAP do Cooja não realiza a implementação do protocolo RPL durante as simulações, impossibilitando o roteamento dos pacotes. O hardware disponível para emulação no Contiki-NG não possuía recursos computacionais suficientes para a implementação do protocolo CoAP. A impossibilidade de emulação do dispositivo fazia com que a coleta de dados de energia e tempo de CPU fosse impraticável.

Outro problema encontrado durante a simulação foi a limitação do tamanho máximo de rede. O RPL sem armazenamento faz com que todos os pacotes sejam transmitidos à raiz da rede, assim, o servidor necessita de ter em sua tabela de roteamento rotas à todos os outros dispositivos que constituem a rede. O consumo de memória da tabela cresce conforme o tamanho da rede, experimentos empíricos durante as simulações mostraram o número limite de nós da rede suportados pelo servidor de plataforma *Zolertia Z1* foi de 80 nós. Assim, o número de nós escolhido para a rede foi de 64 sensores.

4.2.1 Implementação dos Ataques

A IETF estabelece uma pilha de protocolos para aplicações de IoT, considerada o padrão em áreas de LLNs e IoT (PALATTELLA et al., 2013). Contudo, dispositivos e protocolos estão suscetíveis a diversos tipos de ataques que podem comprometer sua disponibilidade. Cada tipo de ataque pode ter impactos distintos, demandando contramedidas específicas para proteger a rede. Os ataques ativos são aqueles que ativamente atuam no funcionamento da rede nas camadas da pilha de protocolos, onde são classificados pela camada afetada (DEOGIRIKAR; VIDHATE, 2017b) (ZHOU; FANG; ZHANG, 2008). Para o treino do classificador do IDS foram implementados os ataques *Blackhole* e *Greyhole*, implementados na camada de enlace, e *Flooding*, implementado na camada de transporte. Os três ataques foram executados em nós utilizando o Contiki-NG.

Os ataques do tipo *Blackhole* são ativos, com o atacante descartando os pacotes que deveriam ser encaminhados na rede. Quando se utiliza um protocolo de roteamento baseado em topologia de árvore, o nó malicioso que executa esse tipo de ataque interrompe a conectividade dos nós filhos com o destino final da rede.

Para realizar esse ataque, é necessário modificar o *framer* do nó malicioso. A implementação do *framer* é fornecido pelo sistema operacional de acordo com o protocolo de enlace. Ao alterar o código responsável pela criação dos quadros transmitidos pelo nó, desativando a rotina de criação de quadros, impede-se que o nó malicioso encaminhe qualquer tipo de pacote após ativar o comportamento mal-intencionado. O Pseudocódigo 1 ilustra essa modificação do *framer*.

Pseudocódigo 1: Modificação do Framer para implementação de ataque Blackhole.

```
if iniciarAtaque and nodeID == attackerNodeID then
    Packet Blackholed
else
    Create and Foward Frame
end if
```

Os ataques *Greyhole* compartilham semelhanças com os ataques *Blackhole* ao descartar pacotes em vez de encaminhá-los. A principal diferença reside no fato de que os ataques *Blackhole* absorvem todos os pacotes recebidos, enquanto os *Greyhole* absorvem apenas parte deles. Portanto, os ataques *Greyhole* tendem a ser mais difíceis de detectar.

A implementação desse tipo de ataque é semelhante à do *Blackhole*, mas apresenta um comportamento de encaminhamento seletivo. As alterações no *framer* do nó malicioso impedem parte das transmissões e encaminhamentos de pacotes. Para determinar quais pacotes são descartados, é realizado um sorteio de um número aleatório de 0 a 100, e se esse número for menor que a porcentagem pré-definida, o pacote é descartado. O Pseudocódigo da implementação do ataque *Greyhole* pode ser encontrado no Pseudocódigo 2.

Pseudocódigo 2: Modificação do Framer para implementação de Greyhole.

```
dropRatio = 30,
          = 60,
          = 90
if iniciateAttack and nodeID==attackerNodeID and rand % 100 ≤ dropRatio then
    Packet Blackholed
else
    Create and Foward Frame
end if
```

O ataque do tipo *Flooding* é considerado ativo e sua execução visa sobrecarregar a rede por meio do envio repetido de pacotes. Para realizar o ataque de *Flooding* na camada de transporte também foi utilizado como referência um programa exemplo de comunicação UDP disponível no Contiki-NG (CONTIKI-NG, 2022a), no qual o cliente envia requisições a cada 60 segundos. O nó atacante, por sua vez, envia novas requisições a cada 300 milissegundos, inundando, assim, a rede e o destino das requisições UDP.

Este tipo de ataque não apenas sobrecarrega o destino dos dados, mas também afeta outros nós ao forçá-los a rotear as mensagens até seu destino. Esse comportamento malicioso gera dificuldades no acesso ao meio por parte dos outros nós da rede e provoca colisões de mensagens, atrasando as requisições. Além disso, o ataque congestiona o *buffer* de mensagens dos nós intermediários entre o atacante e o destino da rede, resultando na perda de pacotes recebidos e gerados por esses nós, além do roteamento de uma grande quantidade de pacotes vizinhos. O pseudocódigo dessa modificação está disponível no Pseudocódigo 3.

Pseudocódigo 3: Modificação do código do dispositivo para implementação de Flooding.

```
while True do
    if eTimerExpired then
        Send UDP request
        if nodeID==attackerNodeID and iniciateAttack then
            setTimer(0,3s)
        else
            setTimer(60s)
        end if
    end if
end while
```

Os cenários de simulação realizados abrangem todas as implementações ataques em diferentes densidades de atacantes e tempo do início de ataque. Todos os dados coletados pelos cenários de simulação abordados estão disponíveis na pasta *trainset* disponível no Github do projeto: <https://github.com/AleMarquis/ArtigoWTICGSBSeg23>.

```
"Horário de recebimento ID: 1 [INFO: App   ] RECIEVED: conteúdo do Pacote
PACKET LENGTH: tamanho do pacote | FROM: endereço IPv6 do Cliente"
```

Figura 9 – Padrão dos pacotes recebidos pelo servidor durante a simulação

```
01:55.461  ID:1  [INFO: App ] Received: 'No: 0 | Temperature: 25 [°C] |
Pressure: 100000 [Pa]' Packet length: 53 | from: fd00::c30c:0:0:b

01:53.047  ID:1  [INFO: App ] Received: 'CPU 1(s) / Energy 61855(uC/m)'
Packet length: 29 | from: fd00::c30c:0:0:2e
```

Figura 10 – Exemplos dos pacotes de aplicação e operação, respectivamente

4.3 Tratamento Estatístico dos Dados

Como discutido anteriormente, um dos requisitos para utilizar modelos classificadores é a utilização de dados estruturados para treinamento e classificação. As consequências de ataques à rede IoT muitas vezes modificam o funcionamento da rede e o roteamento dos pacotes dos nós ao servidor. Os ataques podem fazer com que dados cheguem incompletos ou até impeçam a transmissão de pacotes, dificultando a estruturação dos dados analisados pelo classificador. Estes problemas no escopo tratado pelo projeto foram contornados com um tratamento estatístico dos dados, um processo de tradução dos pacotes recebidos pelo servidor para 14 entradas do modelo, somadas ao horário de chegada do pacote e sua classificação de normal ou anômala, esta última para o treinamento supervisionado.

Os dados brutos possuem o formato de arquivo de texto (.txt) e contém informações sobre o pacote enviado, juntamente com seu conteúdo. Um pacote normal relatado pelo servidor possui estrutura base apresentada pela Figura 9

Os clientes foram programados para enviar dois tipos de pacotes diferentes: (i) Pacotes de aplicação, apresentando informações para aplicação do sistema, com o índice de temperatura e pressão indicados pelos sensores do dispositivo; e (ii) Pacotes de Operação, apresentando informações de operação do dispositivo, com o tempo que a plataforma esteve em modo de CPU ativo e a energia consumida por minuto.

A Figura 10 apresenta exemplos dos pacotes de aplicação e operação, respectivamente

Os *logs* de uma simulação podem possuir até dezenas de milhares destes pacotes em um arquivo. Para a detecção de intrusões utilizando a estatística das métricas analisadas é utilizada a estratégia de janelamento dos dados, utilizando uma fila *First-in First-Out* de pacotes, onde a janela desliza no eixo crescente do tempo. O janelamento dos dados é

Figura 11 – Exemplo do funcionamento do janelamento implementado nos dados. Neste caso: Tamanho da Janela = 4 e Velocidade da janela = 2.

utilizado para que todos os cálculos estatísticos sejam aplicados em um espaço amostral local das métricas, possibilitando a detecção da ocorrência de ataque neste espaço de tempo. A Figura 11 apresenta como o janelamento ocorre no tratamento estatístico dos dados.

A abordagem do janelamento nos permite contornar limitações de acesso à informação inerente à redes de dispositivo IoT sem fio. Porém, esta abordagem, pela análise local dos dados, acaba necessitando que a densidade de pacotes de informação dos nós seja homogênea para não impactar no modelo de inferência. Esta densidade constante faz com que, nesta arquitetura, seja necessária a transmissão de pacotes de operação do dispositivo em todo envio de pacotes de aplicação. Isto acaba gerando impacto na rede pela constante de pacotes de operação, onde um dispositivo enviava um pacote por minuto, agora passa mandar dois, efetivamente duplicando a quantidade pacotes na rede.

Após o janelamento dos dados brutos, elencou-se informações importantes de rede e operação para a inferência de ataques. Além das métricas indicativas de ataques elencadas por Carrer e Margi (2023), outras de operação dos dispositivos e de rede foram utilizadas como entrada do sistema. A lista final das variáveis analisadas está descrita abaixo.

1. Média e desvio padrão do tempo que cada dispositivo passou com a plataforma de hardware em CPU nas mensagens analisadas;
2. Média e desvio padrão do tamanho médio dos pacotes enviados para o servidor nas mensagens analisadas;
3. Média e desvio padrão da energia gasta por minuto pelos sensores nas mensagens analisadas;
4. Desvio padrão do número de requisição do cliente nas mensagens analisadas;

5. Média e desvio padrão do *Freshness* do pacote enviado nas mensagens analisadas. *Freshness* pode ser definido como o quão recente é o pacote e é utilizado para analisar atraso em redes para pacotes que devem ser retransmitidos para chegar ao seu destino;

6. Média e desvio padrão do valor apontado como temperatura do sensor nas mensagens analisadas;

7. Média e desvio padrão do valor apontado como pressão do sensor nas mensagens analisadas;

8. Quantidade de diferentes clientes que enviaram mensagens ao servidor nas mensagens analisadas:

O código que realiza o tratamento estatístico é um script em Python que tem o objetivo de criar um arquivo CSV com dados extraídos de um arquivo de texto. O código faz a extração dos dados indicados pelas entradas brutas utilizando *Regex*, uma forma de usar expressões regulares, sequências de caracteres, como padrão de busca. Daí, são extraídos a hora de recebimento, o tempo de CPU, o tamanho do pacote, o gasto de energia, o número da requisição, o valor da temperatura e pressão e o ID do remetente de cada pacote. Este código está disponível no Github em: <https://github.com/AleMarquis/ids4lln>

Após isto, é construída uma função que que retorna um dicionário com os dados processados. O dicionário contém listas com os valores extraídos pela função de extração dos dados, e guarda as variáveis estatísticas apresentadas anteriormente. Também é realizada a classificação dos dados com base na comparação da hora de recebimento do pacote com o tempo de inicialização do ataque.

É criado um arquivo chamado 'treino.csv' em modo de escrita e cria um objeto writer para escrever os dados no formato *CSV*. O janelamento é definido como uma função que percorre as linhas do arquivo de texto em intervalos definidos pelo tamanho e velocidade da janela. Os dados estatísticos são escritos no *CSV*, que será a entrada para o treinamento do modelo classificador.

4.4 Modelo Classificador

Conforme apresentado na arquitetura do sistema, os dados após o tratamento estatístico são levados ao classificador para o treinamento do modelo. O modelo escolhido foi o *XGBoost* por seu desempenho relativo a outros modelos classificadores baseados em árvores de decisão (CHEN; GUESTRIN, 2016). O código utiliza as *Features* apresentadas pelo CSV para o treinamento do modelo. Assim, o modelo pode receber outros *Logs* de simulações para categorizar as novas entradas em normais ou anômalas.

O arquivo CSV é dividido em uma matriz de dados e um vetor de suas classificações para o treinamento do modelo utilizando aprendizado por reforço. O dataset então é dividido em dados de treino e de teste com uma proporção de 70% e 30% do conjunto de dados originais, respectivamente. O desbalanceamento das classes é diminuído estimando os pesos das amostra por classificação utilizando a biblioteca *Sklearn*. O modelo do XGBClassifier é instanciado com a biblioteca *XGBoost* usando o parâmetro de pesos para classificações binárias *scale_pos_weight* igual a 5, que é uma forma de dar maior impacto nas classificações de ocorrência de intrusão em rede. Isto acarreta em uma ligeira diminuição da acurácia do sistema, mas aumenta sua taxa de *Recall*, e no geral, a confiabilidade no IDS.

Com isso, é realizada a classificação de novos dados, seja utilizando a partição de treino do *dataset*, como de novas simulações com casos *Out of Distribution*. O código calcula e imprime as métricas de avaliação do modelo, como acurácia, precisão, revocação e F1-Score, usando as funções do *sklearn*. Por fim, apresenta a matriz de confusão do classificador, usando as funções nativas do *sklearn*. Os resultados são apresentados na próxima seção.

4.5 Resultados da pesquisa

Podemos dividir os resultados da pesquisa em duas partes, os voltados à simulação de rede e implementação dos ataques, fazendo parte do ambiente da simulação na arquitetura da solução, e os resultados do modelo classificador, que faz parte do sistema de detecção da arquitetura da solução.

4.5.1 Implementação dos Ataques e Dataset de IoT para detecção de intrusão

Como resultados das simulações de rede temos a contribuição da implementação dos ataques *blackhole*, *greyhole* e *flooding* no sistema operacional Contiki-NG, juntamente com a análise de comportamento da rede IoT sobre estes ataques. Os impactos dos ataques na rede foram analisados por meio da resposta do servidor e da interrupção do protocolo de roteamento. As simulações abrangeram diversas densidades de atacantes e distintas distâncias entre o atacante e o servidor. Os resultados evidenciam a habilidade de correlacionar as métricas obtidas com o tipo de ataque e a proximidade geográfica do nó atacante. Esta evidência de correlação da mudança de métricas de rede com a ocorrência do ataques é um resultado alcançado deste trabalho. Carrer e Margi (2023) apresenta estas discussões em maiores detalhes.

Outra contribuição está no dataset de simulações de redes de dispositivos IoT criado para o treinamento do modelo. O dataset foi gerado por diferentes simulações de ataques à rede IoT apresentada na arquitetura da solução, os dados foram coletados a

partir dos pacotes recebidos pelo sorvedouro da rede. O resultado é a geração de um dataset que contem métricas de rede e operação dos dispositivos que a compõe. O dataset está disponível em formato bruto, com os registros das simulações em *TXT*, e após a estruturação dos dados por tratamento estatístico em *CSV*. O formato das features do CSV estão apresentados na Tabela 4. O dataset pode ser encontrado no Github do projeto: <https://github.com/AleMarquis/ids4lln>

Feature	**Descrição**	**Tipo de dado**
mean_time	Tempo médio do recebimento	Timestamp
cpu_time_mean	Média de tempo de CPU	Integer [s]
cpu_time_stdev	Desvio padrão de tempo de CPU	Float [s]
packet_lenght_mean	Média de tamanho de pacote	Integer
packet_lenght_stdev	Desvio padrão de tamanho de pacote	Float
energy_mean	Média de taxa de consumo de energia	Float [uC/min]
energy_stdev	Desvio padrão de taxa de consumo de energia	Float [uC/min]
req_num_stdev	Desvio padrão do número do pacote enviado	Float
freshness_mean	Média de *Freshness* dos pacotes	Integer
freshness_stdev	Desvio padrão de *Freshness* dos pacotes	Float
temp_val_mean	Média dos valores de temperatura	Float [ºC]
temp_val_stdev	Desvio padrão dos valores de temperatura	Float [ºC]
pressure_val_mean	Média dos valores de pressão	Float [Pa]
pressure_val_stdev	Desvio padrão dos valores de pressão	Float [Pa]
number_of_clients	Número de Clientes ativos no intervalo	Integer
classification	Classificação da janela em normal ou anômala	0 ou 1

Tabela 4 – Lista de *features* passadas ao modelo após o tratamento estatístico das métricas

4.5.2 Avaliação do Modelo

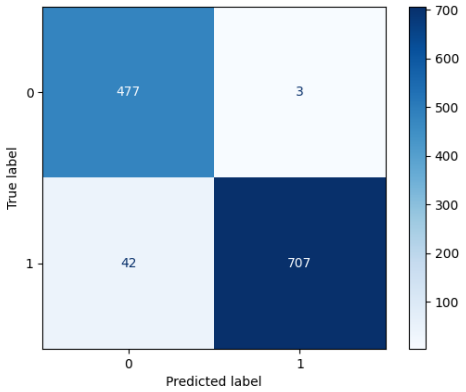
Por fim, são apresentados as métricas de avaliação do modelo para diferentes testes de validação conforme o método da pesquisa: (i) Validação por divisão do *dataset* em treino e teste; e (ii) Validação por classificação com dados de simulações *Out of Distribution*. Para esta, foi realizada validações com cada ataque implementado no trabalho: Blackhole, Greyhole e Flooding. Também foi calculada a acurácia do modelo em casos de redes normais, onde não existe a ocorrência do ataque. A Tabela 5 apresenta os resultados de Acurácia, Precisão, *Recall* e *F1-Score* para cada um destes cenários citados anteriormente.

		Out of Distribution			
Métricas	Treino-Teste	Blackhole	Greyhole	Flooding	Rede Normal
Acurácia	96.49%	96.34	86.20%	92.90%	94.68%
Precisão	96.77%	99.58%	72.37%	85.08%	-
Recall	97.37%	94.39%	84.62%	100.00%	-
F1-Score	97.07%	96.92%	78.01%	91.94%	-

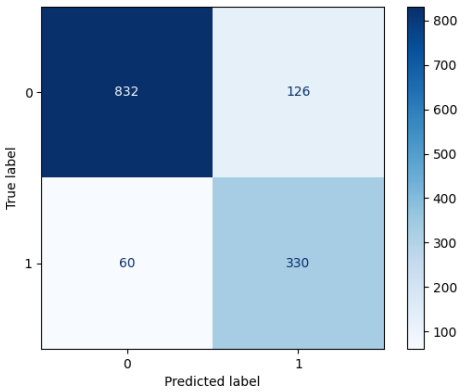
Tabela 5 – Tabela de métricas de avaliação do modelo de XGBoost por método de avaliação

A Figura 12 apresenta as matrizes de confusão dos cenários de *Out of Distribution* e Rede normal da Tabela 5. Nos dados apontados é possível ver a consequência das ações tomadas na criação do modelo alinhados aos requisitos do sistema de detecção de intrusão. A diminuição dos casos de Falsos Negativos, ataques que ocorreram, mas não foram detectados pelo IDS, pode ser percebida pela métrica de *Recall* e também indicada na matriz de confusão em seu quadrado inferior esquerdo.

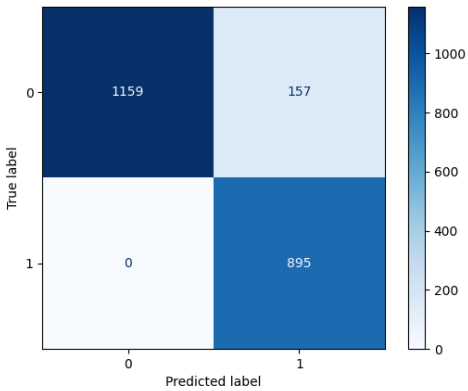
As métricas de avaliação apontaram um pior desempenho do IDS em detecção de ataques do tipo Greyhole. Este comportamento é esperado pela maior sutileza do ataque, o *selective forwarding* dificulta na percepção do ataque uma vez que pode se passa por instabilidade da rede. Ataques com consequências mais perceptivas na rede, como o caso do Flooding apresentam melhores as métricas de detecção por serem mais evidentes em sua ocorrência.



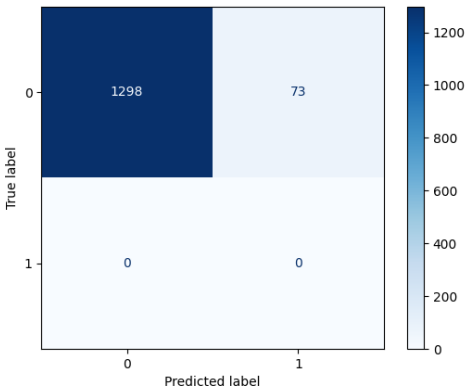
(a) Ataque Blackhole



(b) Ataque Greyhole



(c) Ataque Flooding



(d) Rede Normal

Figura 12 – Matrizes de Confusão dos testes Out of Distribution do modelo

A Figura 13 apresenta a matriz de confusão geral do sistema de detecção de intrusão e corresponde à performance geral do sistema. O desempenho do método de avaliação do classificador por treino-teste acaba sendo maior que a média das avaliações *Out of*

Distribution pelos padrões apresentados durante os ataques se repetindo nos dados de treino e teste. As *Features*, que são percebida pelo classificador, não são amplamente modificadas além do início do ataque. Isto implica em *Features* de redes sob ataque semelhantes entre si, assim, temos uma maior facilidade de classificação de dados do *dataset* de treino e melhores métricas de avaliação.

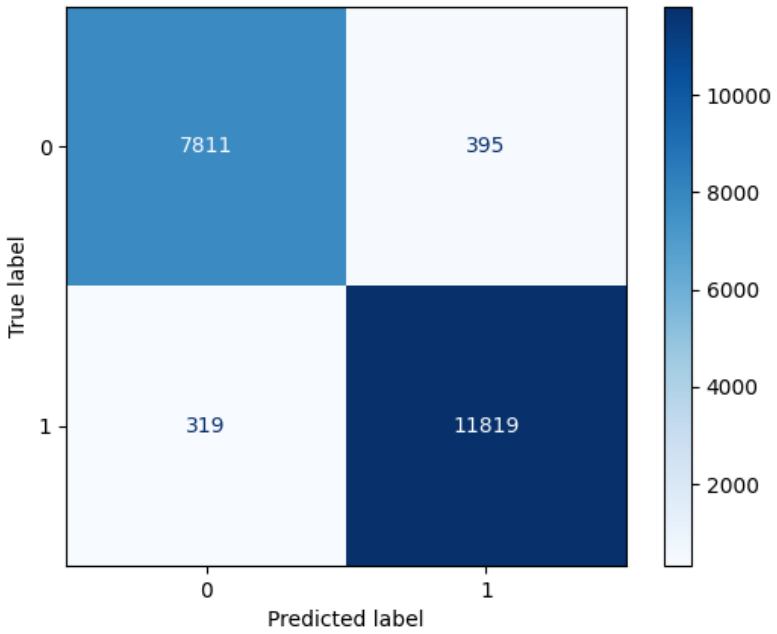


Figura 13 – Matriz de confusão do modelo XGBoost com validação usando a divisão treino-teste do *dataset*

Além dos testes utilizando métricas de avaliação de modelo, foi realizado o teste de tempo de detecção de ataque. O sistema realizava as predições e, quando a primeira detecção de ataque era registrada, o horário médio dos pacotes no janelamento era comparado com o horário de início de ataque. Porém, a priorização de diminuição de Falsos Negativo acarretou em um pequeno aumento na taxa de Falsos Positivos, condições normais de rede que foram classificadas como ataque. Os Falsos Positivos indicados pelo sistema impediam o cálculo correto do tempo de início do ataque. Esta classificação errada ocorre durante instabilidade de rede, que são inerentes ao ambiente IoT simulado no projeto.

Os resultados das predições obtiveram desempenho satisfatório quando se trata de uma ferramenta para aprimorar a segurança da rede de uma arquitetura IoT. O modelo foi projetado para minimizar classificações Falsos Negativas e os resultados gerais do sistema apontaram em um modelo com métricas avaliativas satisfatórias, principalmente se tratando de *recall*. Nas simulações realizadas, o *tradeoff* de impacto na rede para a implementação do IDS não se mostrou negativo o suficiente para dissuadir os ganhos no viés da segurança, mesmo dobrando no número de pacotes transmitidos pelos dispositivos com o envio de métricas de operação.

5 Conclusões

A segurança é um requisito crítico na IoT, sendo a detecção de ataques à rede fundamental para garantir a integridade, confidencialidade e disponibilidade dessa rede. A literatura apresenta soluções de sistemas de detecção de intrusão (IDS) não alinhados com o paradigma da IoT, utilizando protocolos que não seguem padrões internacionais e fazem uso de dados de outros paradigmas de redes de computadores. Com o objetivo de preencher essa lacuna na pesquisa, este trabalho propõe, implementa e avalia um sistema de detecção de intrusão para a IoT. Esse sistema utiliza dados da operação dos dispositivos e das métricas da rede em que estão inseridos, baseando a detecção de ataques em aprendizado de máquina através do classificador *XGBoost*. Os dados da rede IoT são coletados por meio de simulações no Contiki Cooja, envolvendo redes sob ataques dos tipos *Blackhole*, *Greyhole* e *Flooding*. Uma análise estatística é aplicada aos dados coletados para torná-los compatíveis com o modelo de aprendizado supervisionado. O resultado dessa simulação é disponibilizado em formato de dataset para detecção de intrusão, contendo 15 características. Por fim, o modelo criado demonstra uma acurácia geral e uma taxa de *recall* superiores a 80% para todos os cenários de simulação implementados.

Além de seu objetivo, o projeto aborda duas questões de pesquisa que tratam da viabilidade da solução. A primeira discute o impacto da utilização de métricas de operação dos dispositivos na rede, o trabalho expõe a necessidade de dobrar o número de pacotes na rede para o envio das métricas ao modelo que irá utilizá-las, isto acarreta no atraso na entrega dos pacotes e prejudica sua taxa de entrega. A segunda discute a viabilidade de detecção de intrusão utilizando aprendizado de máquina, a pesquisa realizada aponta que, com a estruturação dos dados, é possível utilizar esta técnica de IA para a inferência de intrusão com desempenho satisfatório, e a pesquisa aponta que abordagem é viável na abordagem do IDS centralizado na rede

Para criar o modelo, foi preciso coletar dados por emulação de vários cenários de redes. A partir dos dados brutos das simulações, ocorreu a estruturação para torná-los úteis ao modelo classificador. Por fim, o modelo foi treinado com foco na minimização de casos de não detecção de ataques.

Na simulação de rede foi procurado a maior pluralidade de dados possível, buscando maior diversidade nas simulações realizadas para a coleta de dados. Os cenários de simulação abrangem a variedade de implementações de ataques com diferentes densidades de atacantes e tempos de início de ataque, chegando a 30 registros de simulações utilizadas pelo modelo.

Para a realização do tratamento estatístico foi discutido como seria realizada a estruturação dos dados para sua utilização no modelo classificador. A abordagem escolhida

foi o janelamento, o que causa um impacto direto na sobrecarga da rede. A estratégia de janelamento permite superar as restrições de acesso à informação inerentes às redes de dispositivos IoT sem fio. Contudo, essa abordagem, ao analisar localmente os dados, requer uma homogeneidade na densidade de pacotes nos nós para não afetar o modelo de inferência. Esta densidade constante exige a transmissão de pacotes de operação do dispositivo a cada envio de pacotes de aplicação nessa arquitetura. Resultando no aumento da quantidade de pacotes transitando na rede.

Para o modelo classificador, os resultados das previsões demonstraram um desempenho semelhante à outros trabalhos apresentados na seção de revisão de literatura, agora utilizando protocolos padronizados e um conjunto de dados coletados de ambientes IoT. Assim, o desempenho da ferramenta foi considerado satisfatório para reforçar a segurança na arquitetura IoT. O modelo foi concebido visando minimizar Falsos Negativos, e os resultados gerais do sistema indicaram métricas satisfatórias, especialmente em relação ao *recall*.

5.1 Perspectivas de Continuidade

No que se diz respeito na continuidade da pesquisa, é identificado como trabalho futuro a melhoria do modelo classificador e sua aplicação em uma rede real. Apesar de árvores de decisão serem uma estratégia simples de aprendizagem supervisionada, ainda existe dificuldades de aplicar estes modelos em dispositivos computacionalmente e energeticamente limitados. Futuramente serão elencados os *tradeoffs* da centralização do IDS no viés de sobrecarga da rede e analisadas abordagens descentralizadas para IDSs em IoT para dividir nos dispositivos o custo computacional.

Outro ponto de interesse em trabalhos futuros é utilizar a pilha de protocolos IETF IoT completa, utilizando o protocolo CoAP da camada de aplicação no projeto final. Está estipulado também a análise da possibilidade de utilizar técnicas de *Zero-shot learning* para a classificação de intrusão sem a implementação de ataques. Esta abordagem faz com que o IDS gerado seja totalmente baseado em anomalia conforme sua taxonomia.

Por se tratar de um projeto de pré-mestrado, as perspectivas de continuidade do projeto serão perseguidas no mestrado do aluno.

Referências

AL-TURJMAN, F.; ABUJUBBEH, M. Iot-enabled smart grid via sm: An overview. *Future Generation Computer Systems*, v. 96, p. 579–590, 2019. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X1831759X>. Citado na página 12.

ALBULAYHI, K. et al. Iot intrusion detection taxonomy, reference architecture, and analyses. *Sensors*, v. 21, n. 19, 2021. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/21/19/6432>. Citado na página 14.

ALEXANDER, R. et al. *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*. RFC Editor, 2012. RFC 6550. (Request for Comments, 6550). Disponível em: <https://www.rfc-editor.org/info/rfc6550>. Citado na página 20.

ALMIANI, M. et al. Deep recurrent neural network for iot intrusion detection system. *Simulation Modelling Practice and Theory*, Elsevier, v. 101, p. 102031, 2020. Citado na página 29.

ALMOMANI, I.; AL-KASASBEH, B.; AL-AKHRAS, M. Wsn-ds: A dataset for intrusion detection systems in wireless sensor networks. *Journal of Sensors*, Hindawi, v. 2016, 2016. Citado na página 29.

ALRASHDI, I. et al. Ad-iot: Anomaly detection of iot cyberattacks in smart city using machine learning. In: IEEE. *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. [S.l.], 2019. p. 0305–0310. Citado 2 vezes nas páginas 29 e 31.

ATZORI, L.; IERA, A.; MORABITO, G. Understanding the internet of things: definition, potentials, and societal role of a fast evolving paradigm. *Ad Hoc Networks*, v. 56, p. 122–140, 2017. ISSN 1570-8705. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1570870516303316>. Citado na página 17.

BARTHEL, D. et al. *Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks*. RFC Editor, 2012. RFC 6551. (Request for Comments, 6551). Disponível em: <https://www.rfc-editor.org/info/rfc6551>. Citado na página 39.

CARRER, A.; MARGI, C. Segurança em internet das coisas: Uma análise de comportamento de rede sob ataque. In: *23º Workshop de Trabalhos de Iniciação Científica e de Graduação, SBSeg 23*. [S.l.: s.n.], 2023. Citado 3 vezes nas páginas 15, 44 e 46.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. [S.l.: s.n.], 2016. p. 785–794. Citado 2 vezes nas páginas 37 e 45.

CONTIKI-NG. *rpl-udp — Contiki-NG documentation*. 2022. <https://docs.contiki-ng.org/en/develop/examples/rpl-udp/README.html>. Acessado em: 12/dez/2023. Citado 2 vezes nas páginas 39 e 42.

CONTIKI-NG. *RPL — Contiki-NG documentation*. 2022. <https://docs.contiki-ng.org/en/develop/doc/programming/RPL.html>. Acessado em: 11/jul/2023. Citado na página 39.

CONTIKI-OS. *ContikiOS - The Open Source Operating System for the Internet of Things*. 2018. <https://www.contiki-os.org>. Acessado em: 12/dez/2023. Citado na página 34.

CONTIKI-OS. *Introduction to Cooja*. 2019. <https://github.com/contiki-os/contiki/wiki/An-Introduction-to-Cooja>. Acessado em: 12/dez/2023. Citado 2 vezes nas páginas 35 e 37.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, JMLR. org, v. 7, p. 1–30, 2006. Citado na página 26.

DEOGIRIKAR, J.; VIDHATE, A. Security attacks in iot: A survey. In: IEEE. *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*. [S.l.], 2017. p. 32–37. Citado na página 21.

DEOGIRIKAR, J.; VIDHATE, A. Security attacks in iot: A survey. In: *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. [S.l.: s.n.], 2017. p. 32–37. Citado na página 41.

EDDY, W. *Transmission Control Protocol (TCP)*. RFC Editor, 2022. RFC 9293. (Request for Comments, 9293). Disponível em: <https://www.rfc-editor.org/info/rfc9293>. Citado na página 21.

FEENEY, L. M.; NILSSON, M. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In: IEEE. *Proceedings IEEE INFOCOM 2001. Conference on computer communications. Twentieth annual joint conference of the IEEE computer and communications society (Cat. No. 01CH37213)*. [S.l.], 2001. v. 3, p. 1548–1557. Citado na página 19.

FIGUEROA, H.; WANG, Y.; GIAKOS, G. C. Adversarial attacks in industrial control cyber physical systems. In: *2022 IEEE International Conference on Imaging Systems and Techniques (IST)*. [S.l.: s.n.], 2022. p. 1–6. Citado na página 12.

FONSECA, R. et al. The collection tree protocol (ctp). *TinyOS TEP*, v. 123, n. 2, 2006. Citado na página 20.

HEINZELMAN, W. R.; CHANDRAKASAN, A.; BALAKRISHNAN, H. Energy-efficient communication protocol for wireless microsensor networks. In: IEEE. *Proceedings of the 33rd annual Hawaii international conference on system sciences*. [S.l.], 2000. p. 10–pp. Citado na página 20.

LOHIYA, R.; THAKKAR, A. A review on machine learning and deep learning perspectives of ids for iot: Recent updates, security issues, and challenges. *Archives of Computational Methods in Engineering*, v. 28, 10 2020. Citado na página 14.

MOTEIV. *Tmote sky - Datasheet*. 2006. <https://www.snm.ethz.ch/snmwiki/pub/uploads/Projects/tmote_sky_datasheet.pdf>. Acessado em: 08/dez/2023. Citado 3 vezes nas páginas 8, 35 e 36.

MOUSTAFA, N.; TURNBULL, B.; CHOO, K.-K. R. An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. *IEEE Internet of Things Journal*, IEEE, v. 6, n. 3, p. 4815–4830, 2018. Citado na página 29.

OIKONOMOU, G. et al. The Contiki-NG open source operating system for next generation IoT devices. *SoftwareX*, v. 18, p. 101089, 2022. ISSN 2352-7110. Citado 2 vezes nas páginas 34 e 37.

OPREA, A.; SINGHAL, A.; VASSILEV, A. Poisoning attacks against machine learning: Can machine learning be trustworthy? *Computer*, IEEE, v. 55, n. 11, p. 94–99, 2022. Citado 2 vezes nas páginas 14 e 34.

OZKAN-OKAY, M. et al. A comprehensive systematic literature review on intrusion detection systems. *IEEE Access*, v. 9, p. 157727–157760, 2021. Citado 2 vezes nas páginas 24 e 25.

PALATTELLA, M. et al. Standardized protocol stack for the internet of (important) things. *IEEE Communications Surveys 'I&' Tutorials*, v. 15, n. 3, p. 1389–1406, 2013. Citado 2 vezes nas páginas 31 e 41.

PATEL, K. K.; PATEL, S. M.; SCHOLAR, P. Internet of things-iot: definition, characteristics, architecture, enabling technologies, application & future challenges. *International journal of engineering science and computing*, v. 6, n. 5, 2016. Citado na página 12.

POSTEL, J. *User Datagram Protocol*. RFC Editor, 1980. RFC 768. (Request for Comments, 768). Disponível em: <https://www.rfc-editor.org/info/rfc768>. Citado 2 vezes nas páginas 21 e 39.

RASAL, R.; GUMASTE, S.; DEOKATE, G. Effective survey on hierarchical routing protocols in wsn. *International Journal of Innovative Research in Engineering Management*, v. 2, p. 49–52, 05 2015. Citado 2 vezes nas páginas 7 e 25.

ROULSTON, M. S.; SMITH, L. A. The boy who cried wolf revisited: The impact of false alarm intolerance on cost–loss scenarios. *Weather and Forecasting*, American Meteorological Society, v. 19, n. 2, p. 391–397, 2004. Citado na página 28.

SHELBY, Z.; HARTKE, K.; BORMANN, C. *The Constrained Application Protocol (CoAP)*. RFC Editor, 2014. RFC 7252. (Request for Comments, 7252). Disponível em: <https://www.rfc-editor.org/info/rfc7252>. Citado na página 21.

TSCHOFENIG, H. et al. *Architectural Considerations in Smart Object Networking*. RFC Editor, 2015. RFC 7452. (Request for Comments, 7452). Disponível em: <https://www.rfc-editor.org/info/rfc7452>. Citado 2 vezes nas páginas 30 e 32.

TSIMENIDIS, S.; LAGKAS, T.; RANTOS, K. Deep learning in iot intrusion detection. *Journal of network and systems management*, Springer, v. 30, p. 1–40, 2022. Citado 2 vezes nas páginas 7 e 13.

WEISS, J. et al. Using machine learning to work around the operational and cybersecurity limitations of legacy process sensors. *Computer*, v. 55, n. 11, p. 106–111, 2022. Citado na página 13.

ZARPELãO, B. B. et al. A survey of intrusion detection in internet of things. *Journal of Network and Computer Applications*, v. 84, p. 25–37, 2017. ISSN 1084-8045. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1084804517300802>. Citado na página 14.

ZHOU, Y.; FANG, Y.; ZHANG, Y. Securing wireless sensor networks: a survey. *IEEE Communications Surveys 'I&' Tutorials*, v. 10, n. 3, p. 6–28, 2008. Citado na página 41.

ZOLERTIA. *Z1 Zolertia - Datasheet*. 2010. <https://issuu.com/zolertia/docs/z1_datasheet>. Acessado em: 08/dez/2023. Citado 2 vezes nas páginas 8 e 36.

ZOLERTIA. *Z1 - Zolertia*. 2013. <https://zolertia.sourceforge.net/wiki/index.php/Z1>. Acessado em: 11/jul/2023. Citado na página 35.