Universidade Federal do Rio de Janeiro

Escola Politécnica

MBA em Engenharia de Computação e Sistemas (MBCA)

SEGURANÇA EM IOT: ANALISANDO RISCOS E AMEAÇAS

Autor:

Electronically signed by: Marcelo Cavalcanti da Costa
Marcelo Cavalcanti da Costa
marcelocavalcanti.info@gmail.com
29/12/2023 11:01:26
ID: 7GU3DGNBYGM-GE2TGOBZG4ZA-Z4VFZIPWTSHTQ

Marcelo Cavalcanti da Costa

Orientador:

Electronically signed by: Norberto Ribeiro Bellas
Norberto Ribeiro Bellas
norbertobellas@gmail.com
29/12/2023 18:39:53
ID: 0GU3DGNBYGM-GE2TGOBZG4ZA-GASG73I5DSVPU

Norberto Ribeiro Bellas, M. Sc.

Coorientador:

Electronically signed by: Manoel Villas Boas Junior
Manoel Villas Boas Junior
mvbjunior@poli.ufrj.br
30/12/2023 12:12:27
ID: 8GU3DGNBYGM-GE2TGOBZG4ZA-AKOU3VRYNKCWA

Manoel Villas Boas Junior, M. Sc.

Examinador:

Electronically signed by: Vinicius Drumond Gonzaga
Vinicius Drumond Gonzaga
viniciusdgonzaga@gmail.com
02/01/2024 10:12:51
ID: 3GU3DGNBYGM-GE2TGOBZG4ZA-ISZFTXOOQ7ZHU

Vinicius Drumond Gonzaga, M. Sc.

Examinador:

Electronically signed by: Claudio Luiz Latta de Souza
Claudio Luiz Latta de Souza
claudio.latta@poli.ufrj.br
02/01/2024 11:39:58
ID: 0GU3DGNBYGM-GE2TGOBZG4ZA-GASG73I5DSVPU

Cláudio Luiz Latta de Souza, M. Sc.

Rio de Janeiro
Agosto de 2023

Declaração de Autoria e de Direitos

Eu, **Marcelo Cavalcanti da Costa,** CPF 077237007-92, autor da monografia segurança em IoT: analisando risco e ameaças, subscrevo para os devidos fins, as seguintes informações:

1. O autor declara que o trabalho apresentado na defesa da monografia do curso de Pós-Graduação, Especialização MBA - Engenharia de Computação e Sistemas da Escola Politécnica da UFRJ é de sua autoria, sendo original em forma e conteúdo.

2. Excetuam-se do item 1 eventuais transcrições de texto, figuras, tabelas, conceitos e ideias, que identifiquem claramente a fonte original, explicitando as autorizações obtidas dos respectivos proprietários, quando necessárias.

3. O autor permite que a UFRJ, por um prazo indeterminado, efetue em qualquer mídia de divulgação, a publicação do trabalho acadêmico em sua totalidade, ou em parte. Essa autorização não envolve ônus de qualquer natureza à UFRJ, ou aos seus representantes.

4. O autor declara, ainda, ter a capacidade jurídica para a prática do presente ato, assim como ter conhecimento do teor da presente Declaração, estando ciente das sanções e punições legais, no que tange a cópia parcial, ou total, de obra intelectual, o que se configura como violação do direito autoral previsto no Código Penal Brasileiro no art.184 e art.299, bem como na Lei 9.610.

5. O autor é o único responsável pelo conteúdo apresentado nos trabalhos acadêmicos publicados, não cabendo à UFRJ, aos seus representantes, ou ao(s) orientador(es), qualquer responsabilização/ indenização nesse sentido.

6. Por ser verdade, firmo a presente declaração.

Rio de Janeiro, 05 de agosto de 2023.

Electronically signed by:
Marcelo Cavalcanti da Costa
Marcelo Cavalcanti da Costa
marcelocavalcanti.info@gmail.com
29/12/2023 11:01:26
ID: 7GU3DGNBYGM-GE2TGOBZG4ZA-Z4VFZIPWTSHTQ

Marcelo Cavalcanti da Costa

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

Av. Athos da Silveira, 149 - Centro de Tecnologia, Bloco H, sala -212,
Cidade Universitária, Rio de Janeiro - RJ - CEP 21949-900.

DEDICATÓRIA

Dedico este trabalho a Deus em primeiro lugar, por todos os benefícios que têm me feito.

Aos meus pais, João Mariano e Ivanilda Cavalcanti (*in memoriam*), molas mestras neste processo, geradores de estímulo, os quais são os meus exemplos de vida e superação em momento de dificuldade. E a minha querida avó Terezinha Francisca Cavalcanti (*in memoriam*), que me dizia: - A vida é um constante aprendizado e que ainda tinha muito a aprender, afirmando isso aos seus 82 anos.

A minha esposa Célia e aos meus filhos, Daniel e David incansáveis na arte de compreender, tolerar e motivar-me a cada dia para vencer esta grande e significativa etapa de nossas vidas.

AGRADECIMENTO

Agradeço a Deus pela sabedoria que me concede, e ao professor Norberto pela ajuda, orientação e compreensão na realização deste trabalho junto a esta renomada Universidade Federal do Rio de Janeiro - UFRJ.

RESUMO

Frequentemente as mídias de comunicação e tecnologia informam de ocorrências de ataques de hackers sejam em ambientes corporativos ou em ambientes comerciais, e até mesmo dentro de nossas residências, em ambiente que possuem acesso à internet. A imaginação dos atacantes para desenvolver novos tipos de invasão não tem limites, sempre explorando as tecnologias que surgem ou se atualizam no mercado. Muito se fala em tecnologia Internet das Coisas (IoT), para automatização residencial, industrial, cientifica e/ou corporativa de Machine Learning (ML) ou do subconjunto Deep Learning (DP) ao usar estes meios pelos acessos da internet, através de Wireless Fidelity - Wi-Fi ou rede cabeada, qualquer um desses dispositivos estão suscetíveis a um ataque hacker. Para um hacker as informações encontradas são de grande importância, sejam elas quais forem, independente da tecnologia utilizada, pois, o seu objetivo principal é a vulnerabilidade da área explorada. Ao se usar dispositivo que possui uma arquitetura IoT, que utiliza a Internet também como meio de comunicação para tráfico de informações, abre-se um grande leque de possibilidades, que possam vir a existir, e em algum momento expor sua vulnerabilidade a possíveis ataques hackers. Pesquisas recentes informam uma tendência do crescimento de casas inteligentes e a utilização desses dispositivos IoT na automação residencial. Podemos dizer que a segurança da informação não está somente vinculada a ambientes corporativos e industriais, mas também nestas novas tecnologias de mercado Internet of Things (IoT), que pode ser utilizada também em automação residencial. Porquanto neste trabalho planejamos ilustrar sobre a construção de um dispositivo de automação residencial, cujo objetivo é capturar os dados referentes a temperatura e umidade do ar, utilizando IoT, tais como, microcontrolador NodeMCU, programação phyton e armazenamento das informações em Cloud. Com este protótipo teremos um cenário IoT para identificar tipos de invasões que podem ocorrer e o que pode ser feito para minimizar ou eliminar tais ações.

Palavras Chaves: Cloud, I-Iot, SmartHome, MQTT, Security.

ABSTRACT

Communication and technology media frequently report occurrences of hacker attacks in corporate or commercial environments, and even within our homes, in environments that have internet access. The imagination of adventurers to develop new types of invasion has no limits, always exploring technologies that appear or are updated on the market. There is a lot of talk about Internet of Things (IoT) technology, for residential, industrial, scientific and/or corporate automation of Machine Learning (ML) or the Deep Learning (DP) subset when using these means via internet access, through Wireless Fidelity - Wi-Fi or wired network, any of these devices are susceptible to a hacker attack. For a hacker, the information found is of great importance, whatever it is first, regardless of the technology used, as its main objective is the vulnerability of the exploited area. When using a device that has an IoT architecture, which also uses the Internet as a means of communication for information traffic, a wide range of possibilities opens up, which may come to exist, and at some point expose its vulnerability to possible cyber attacks. hackers. Recent research reports a growing trend in smart homes and the use of these IoT devices in home automation. We can say that information security is not only linked to corporate and industrial environments, but also in these new Internet of Things (IoT) market technologies, which can also be used in home automation. For now, in this work we plan to illustrate the construction of a home automation device, whose objective is to capture data relating to air temperature and humidity, using IoT, such as NodeMCU microcontroller, phyton programming and storing information in the Cloud. With this prototype we will have an IoT scenario to identify types of intrusions that can occur and what can be done to minimize or eliminate such actions.

Keywords: Cloud, I-Iot, SmartHome, MQTT, Security.

SIGLAS

AMQP	Advanced Message Queuing Protocol / Protocolo Avançado de Enfileiramento de Mensagens
BLUETOOH	Conexão sem fio de curto alcance
BOTNET	Formada pela junção das palavras "robot" e "network"
DOS	Denial of Service / Negação de serviço
DDoS	Distributed Denial of Service / Negação de serviço distribuída
DDS	Diálogo Diário de Segurança
DHT22	Sensor de temperatura e umidade
GSM	Global System for Mobile Communications / Sistema Global para Comunicações Móveis
IP	Internet Protocol / protocolo de internet
IPv4	Protocolo de Internet versão 4
IPv6	Protocolo da internet versão 6
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
IoT	Internet of Things / Internet das Coisas
JSON	Javascript Object Notation / Notação de objeto Javascript
JPL	Jet Propulsion Laboratory / Laboratório de Propulsão a Jato
LoRa	Long Range Network / Rede de longo alcance
LPWAN	Low-power wide-area network / Rede de área ampla de baixa Potência
LoraWAN	Low Power Wide Area Networking / Rede de área ampla de baixo consumo de energia
6LowWPAN	IPv6over Low power Wireless Personal Area Networks/ Redes de área pessoal sem fio IPv6over de baixa potência
LTE	Long Term Evolution / Evolução a Longo Prazo
MQTT	Message Queuig Telemetry Transport Transporte de Telemetria Message Queuig
M2M	Machine-to-Machine máquina a máquina
MALWARE	Software Malicioso
MANETs	São redes sem fio, com topologia altamente volátil e totalmente arbitrária
NASA	National Aeronautics and Space Administration Administração Nacional Aeronáutica e Espacial
NodeMCU	Micro controlador
NFC	Near-field communication / Comunicação de campo próximo
NB IoT	Narrowband IoT / banda estreita IoT
PYTHON	Python Software Foundation, linguagem de programação
QoS	Qualidade de Serviço
Raspberry PI	Um micro-computador completo
RFID	Radio Frequency Identification / Identificação de rádio frequência
SIGFOX	Operadora de rede global francesa

TLS	Transport Layer Security / Segurança da Camada de Transporte
TCP	Transmission Control Protocol / Protocolo de Controle de Transmissão
Wi-Fi	Wireless Fidelity / Fidelidade sem fio
WiGIG	Conjunto de protocolos de rede sem fio de 60 GHz
WBAN	Wireless Body Area Network / Rede de área corporal sem fio
WLAN	Wireless Local Area Network / Rede local sem fio
WSN	Wireless Sensor Network/ Sensores de Rede sem Fio
WAN-RAN	Wireless Área Satélite Network / Rede sem fio Área Satélite
WAN-MAN	Wireless Metropolitan Area Network / Rede sem fio de área Metropolitana
WiMAX	Interface sem fio para redes metropolitanas
XMPP	Extensible Messaging and Presence Protocol / Protocolo Extensível de Mensagens e Presença
Zigbee	Protocolo de comunicação e transferência de dados sem fio
4G	Padrão de tecnologia de quarta geração
5G	Padrão de tecnologia de quinta geração

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

Sumário

CAPÍTULO 1

Introdução

1.1 - Tema

A Internet das Coisas é uma inovação tecnológica, baseada em artefatos já consolidados como a Internet e objetos inteligentes (GALEGALE, 2016, pag1) [1]. Em 2016 (Nesheim e Rosnes) [2] afirmavam que as casas inteligentes iriam fazer parte de 0,77% das famílias do mundo todo e que se esperava um crescimento de 2,79% até 2020. E essa previsão realmente está bem mais concreta atualmente, surgindo uma preocupação quanto à garantia de segurança e integridade desses dispositivos e dos dados que estes dispositivos armazenam. Esta preocupação se deve à existência de inúmeras vulnerabilidades nos dispositivos de Internet das Coisas (IoT). De acordo com (MUKHERJEE et al) [3];

> ” Os sistemas de Internet das coisas (IoT) são geralmente compostos por uma rede de dispositivos conectados ou” coisas”. O termo” coisa” aqui pode constituir qualquer dispositivo inteligente, desde dispositivos, sensores automóveis, dispositivos bioquímicos, sensores na segurança interna, para dispositivos de monitoramento do coração em um corpo humano. Na verdade, qualquer objeto que consiga coletar e transferir dados pela rede para”nó” ou core plataforma de computação em nuvem pode ser uma parte de um sistema IoT. ”

Um exemplo que demonstra a necessidade de que se estude a respeito de segurança e a segurança aplicada à Internet das Coisas, se deu ao fato ocorrido a um ataque à National Aeronautics and Space Administration (NASA) em abril de 2018. O ataque ocorreu, após um *Raspberry Pi* que não estava autorizado a ser vinculado à rede do Jet Propulsion Laboratory (JPL), foi alvo de hackers [4]. Os invasores conseguiram roubar 500 megabytes de dados de um dos principais sistemas do laboratório. Além disso, eles usaram essa conexão para encontrar uma vulnerabilidade em uma das portas que lhes permitisse ir mais fundo na rede do JPL. Com isso, eles tiveram acesso a várias informações importantes, incluindo a Deep Space Network da NASA, uma rede de instalações de comunicação de espaçonaves. Como resultado, as equipes de segurança de alguns programas espaciais, como o Veículo de Tripulação Orion e a Estação Espacial Internacional, optaram por se desconectar da rede da agência.

Outro exemplo de ataque, foi o Mirai, descoberto por pesquisadores do "MalwareMustDie" em agosto de 2016[5]. Embora não tenha sido o primeiro malware de IoT a ser descoberto, certamente é o mais proeminente, após derrubar grande parte da internet na costa leste dos EUA, as coisas pioraram, quando a criadora do malware, apelidada de Anna-Senpai, divulgou o código-fonte. Desde então, hackers motivados em todo o mundo, usaram uma estrutura para criar suas próprias variantes de bonetes (tipo de ataque). Eventualmente, os criadores originais do malware foram presos e se declararam culpados no tribunal, mas o impacto da liberação do código acelerou significativamente a criação de bonetes. Novas variantes começaram a aparecer, adicionando novas funcionalidades e explorando uma variedade de vulnerabilidades em dispositivos IoT não seguros.

São estas questões que o presente trabalho pretende responder, através da elaboração de um protótipo em IoT usando o protocolo MQTT (*Message Queuing Telemetry Transport*) da camada de aplicação, usando dispositivo NodeMcu semelhante a dispositivos como arduino IDE, programação pyhon para medição de temperatura e unidade do ar, coletando esses dados e armazenando em Cloud, analisando assim um cenário IoT para identificar quais falhas de segurança pode haver e o que pode ser feito para minimizar ou eliminar estas falhas.

1.2 - Justificativa

De acordo com IoT-Analytics[6], O mercado cresceu para 12,3 bilhões de dispositivos IoT conectados e cerca de US$ 160 bilhões em gastos corporativos com IoT. As perspectivas continuam muito positivas.

Com o aumento do uso de dispositivos inteligentes e a mobilidade de alguns destes dispositivos, a Internet das Coisas se torna exposta a várias vulnerabilidades.
De acordo com Atzori, Iera e Morabito (2010) [7];

> "A Internet das Coisas é extremamente vulnerável a ataques por uma série de razões. Primeiro, muitas vezes, seus componentes passam a maioria do tempo desacompanhado; e assim, é fácil atacá-los fisicamente. Em segundo lugar, a maioria das comunicações são sem fio, o que torna a espionagem extremamente simples. "

Segundo o que foi dito por Ribeiro (2018) [8],

> "Na IoT há uma interconexão de dispositivos, onde vários objetos se encontram conectados, caso um deles tiver sua segurança comprometida e conectado à internet irá afetar todo o conjunto de dispositivos conectados, assim prejudicando a segurança e a resiliência da internet. "

Com base em um estudo feito por Eduard Kovacs[9], é possível afirmar que cerca de 70% dos dispositivos de Internet das Coisas é vulnerável a algum tipo de ataque. Como câmeras de segurança, termostatos, alarmes. Por isso podemos perceber que esses dispositivos estão suscetíveis a possíveis explorações de ataques.

1.3 - Objetivos

O objetivo deste trabalho é identificar e mitigar riscos de segurança cibernética associados aos dispositivos IoT durante todo o ciclo de vida do dispositivo, utilizando um protótipo IoT direcionado a captação de informação sobre a temperatura e umidade do ar em ambiente residencial, esse processo de captação de dados se dá através do sensor de temperatura e umidade do ar DHT22 através de protoboard conectada a placa micro-controladora nodeMCU, bem como uso da rede Wi-Fi integrada ao dispositivo para se conectar no ambiente de Cloud, e assim armazenar esses dados. Para os testes de segurança vamos analisar riscos que podem ocorrer e tentar mitigá-los.

1.3.1 - Objetivo Específico

- Descrever as tecnologias, arquiteturas de redes e sistemas que suportam IoT;
- Descrever sobre vulnerabilidade de segurança da informação em IoT;
- Demostrar protótipo em IoT na camada de aplicação MQTT;
- Descrever sobre a possíveis soluções de segurança para MQTT;

1.4 - Delimitação

Esse escopo não se aplica a identificar tipos de ataques já existentes em ambiente relacionados a rede corporativas que apresentam tráfego de dados externo e interno, independentemente de onde estão sendo utilizados esses acessos, pois as vulnerabilidades exploradas nestas redes são outras, os quais são mais voltados para rede ethernet, Wi-Fi e aplicação. Este trabalho tem uma abordagem de contribuição, para apresentar algum tipo de vulnerabilidade aplicável. Mostrar que falta de gerenciamento de risco e ameaças estão a todo momento a encontrar uma porta vulnerável, ou algum acesso não autorizado a ambientes que utilizam a tecnologia IoT.

1.5 - Descrição

Este trabalho está estruturado da seguinte forma:

No capítulo 2 será abordado e apresentando conceitos sobre embasamento teórico descrevendo o que é IoT, camadas de arquitetura, as tecnologias, a descrição do protocolo MQTT, e exemplificando tipos de ataques.

O capítulo 3 apresenta a proposta tecnológica para o desenvolvimento e construção do protótipo, e descreve etapas da implementação.

Os resultados obtidos serão demostrados no capítulo 4. Seguindo a conclusão do projeto e propostas para trabalhos futuros.

CAPÍTULO 2

Embasamento Terico

2.1 - Ambiente IoT

Acredita-se que o termo “internet of Thing”, em português “internet das Coisas”, ou abreviando “Iot” foi usado pela primeira vez por Kevin Asthon (Magrani,2018) [10] como título de uma apresentação que tinha como tema a ligação de RFID e outros sensores ao emergente tópico da internet. Atualmente, Iot representa conjuntos de objetos, serviços e tecnologia todos conectados à internet, com o objetivo principal de formar um sistema inteligente de dispositivos autônomos que satisfaça a todas as ligações de dispositivo a dispositivo e usuário a dispositivo, conforme (PACHECO, 2018) [11]:

> “A IoT surgiu graças aos avanços das tecnologias de miniaturização de componentes eletrônicos e das tecnologias de comunicação sem fio. O paradigma IoT estabelece que diversos itens do cotidiano podem ter acesso à internet, o que contribui para uma diversidade de benefícios à população. A previsão é que bilhões de “coisas” sejam capazes de conectar-se à internet para prover os mais diversos tipos de informações aos usuários. “

Um ambiente IoT típico tem vários atributos que o distinguem de sistemas tradicionais de networking, como, por exemplo, Sensores de Rede sem Fio(WSN). As redes IoT são constituídas por uma abundância de dispositivos e tecnologias, tendo em cada um a capacidade de comunicação entre protocolos. Utilizando a internet como estrutura de base, estes dispositivos e tecnologias conseguem manter uma comunicação viável e de baixa latência, estando tanto nas proximidades uns dos outros, como separados por grandes distâncias, procurando operar com o mínimo de consumos energéticos possíveis mediante protocolos de comunicação eficientes e sendo os mais seguros e autônomos possíveis.

De acordo com Santos et al. (2016) [12], a IoT pode ser entendida como a combinação de diversas tecnologias de comunicação e a integração de objetos em um ambiente físico ao mundo virtual. A Figura 2.1 ilustra os blocos básicos para a construção da IoT, seguindo de uma breve descrição.

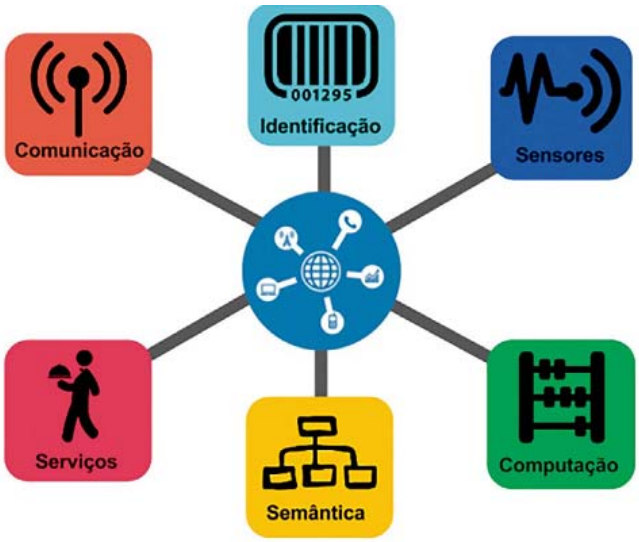

Figura 2.1 Blocos básicos da IoT.
Fonte: Santos et al. (2016, p. 6) [12]

Identificação: Tecnologias empregadas para identificar os objetos (RFID, Near Field Communication [NFC] e endereçamento IP).

Sensores: coletam informações sobre o contexto em que os objetos se encontram e armazenam os dados em data warehouse, centros de armazenamento ou nuvens.

Comunicação: Abrange diversas técnicas usadas para conectar objetos inteligentes. Algumas das tecnologias empregadas são Wi-Fi, IEEE 802.15.4, Bluetooth e RFID.

Computação: Unidade de processamento (microcontroladores, processadores e FPGAs). Executa algoritmos locais nos objetos inteligentes.

Serviços: Engloba serviços de identificação (responsáveis por mapear Entidades Físicas em Entidades Virtuais), de agregação de dados (coletam e sumarizam dados homogêneos ou heterogêneos obtidos dos objetos inteligentes), de colaboração e inteligência (tomam decisões e reagem de modo adequado a um determinado cenário) e de ubiquidade (promovem serviços de colaboração e inteligência em qualquer momento ou lugar necessários).

2.1.2 - Arquitetura de 5 Camadas

Descreveremos as camadas de arquitetura de Internet das Coisas, referenciada na figura 2.2, e entenderemos melhor suas funcionalidades, para que quando comentarmos sobre os tipos de ataques em segurança da informação, possamos compreender melhor

qual camada está mais suscetível a vulnerabilidade. A IoT apresenta descrições para vários níveis de camadas primitivas, de 3 a 4 camadas e composta de 5 a 6 níveis de camadas, mas no trabalho aqui apresentado nos basearemos nos 5 níveis de camada da arquitetura de internet das coisas.

A IoT inclui inúmeros dispositivos inteligentes conectados a uma ampla rede de Internet com a ajuda de várias tecnologias de rede, ao qual na maioria das vezes essas tecnologias são sem fio. Por isso torna a estrutura mais complexa e difícil de gerenciar. Portanto, uma arquitetura estrutural é necessária. A IoT não apresenta uma arquitetura padrão, mas dependendo do projeto, pode-se aplicar uma das estruturas de camadas existentes para Iot, neste trabalho descreveremos sobre a arquitetura de 5 camada

O desenvolvimento da IoT depende das tecnologias usadas, áreas de aplicação e aspectos de negócios. As arquiteturas estão divididas em camadas, algumas delas trabalham com 3 como física, middleware e aplicação, por exemplo, no entanto, a arquitetura de 5 camadas tem sido considerada a melhor para projetos de IoT:

Figura 2.2. Arquitetura de cinco camadas.
Fonte: Arquitetura e Infraestrutura de IoT,2019. [13]

Camada de percepção: É a base de toda a arquitetura e desenvolvimento de aplicação da IoT, incluindo leitores RFID, rede de sensores inteligentes e acesso a gateways, entre outros. Uma matriz de sensores detecta informações relevantes do

ambiente e passa para o gateway mais próximo. Em seguida, o gateway envia pela internet os dados coletados para a plataforma de processamento em segundo plano.

Camada de rede: É a principal responsável pela integração de diferentes tipos de redes, como internet, redes de comunicações móveis e redes de transmissão de TV. Além disso, também fornece roteamento, conversão de formato e de endereçamento.

Camada de middleware: É responsável pela inicialização de recursos, monitoramento de condição das operações dos recursos, coordenação do trabalho entre vários recursos e promoção de interações cruzadas entre as plataformas de recurso.

Camada de processamento de informações: Realiza a compreensão semântica da detecção de dados, além de fornecer consulta de dados, armazenamento, análise, mineração. A computação em nuvem pode proporcionar uma boa plataforma para armazenamento e análise desses dados, sendo um componente importante do processamento de informações.

Camada de aplicativo: A camada de aplicação, após a análise e processamento dos dados detectados, usa esses dados para fornecer aos usuários uma variedade de serviços. Os aplicativos IoT podem ser divididos em monitoramento de rede (logística, controle de poluição), tipo de controle (transporte inteligente, ambientes e residências inteligentes), escaneamento digital (rodovias e vias sem postos físicos de pedágio), entre outros.

A arquitetura de 5 camadas endereça capacidades de gerenciamento e segurança a cada uma das camadas, pode ser genérico ou específico para os dispositivos nas camadas de aplicação e rede. Além disso, a camada de aplicação adiciona confidencialidade nos dados, proteção, integridade, controle, privacidade, auditoria e segurança. A Camada de rede incorpora mecanismos de controle de integridade e confidencialidade, também fazendo o uso de protocolos seguros. Após essa breve descrição das camadas podemos ver que a camada de rede e aplicação podem apresentar algum tipo de vulnerabilidade caso a forma de conectividade na utilização dos protocolos não tenha sido bem definida, devido aos tipos de tecnologias existentes.

2.1.3 – Tecnologia

Para todas essas tendências esperadas com inovações IoT, surge a necessidade de grande atenção com segurança e privacidade para as formas de tecnologias empregadas

na utilização de aplicações usando IoT, vamos entender como se apresenta essa forma de conectividades em IoT.

São inúmeras as possibilidades de elementos que podem se conectar à rede, tornando ambientes residenciais e empresariais mais autônomos, além de transformar cidades com soluções inteligentes, proporcionando crescimento econômico e impulsionando o desenvolvimento tecnológico. Nesse sentido, protocolos de comunicação são muito importantes, por serem linguagens de comunicação entre computadores e dispositivos que permitem a troca de mensagens e transmissão de dados.

De acordo com Leite, Martins; Ursini (2017) [14], a IoT utiliza tecnologias já consolidadas que permitem a comunicação entre dispositivos que, em sua maioria, são tecnologias envolvidas na comunicação por rádio (Radio Access Technologies) e sem fio, e, portanto, utilizam padrões que fornecem alcance e flexibilidade, como IEEE 802.11, IEEE 802.15 e IEEE 802.16. Para cada área de cobertura, costuma haver um tipo de tecnologia usada.

Área metropolitana: Chamada de Wireless Metropolitan Area Network (WAN-MAN). Exemplos: LTE GSM, WiMAX 802.16, SIGFOX, NB-IoT, LoRa e MimoMAX.

Cobertura via satélite: Chamada de Wireless Área Satélite Network (WAN-RAN). Exemplo: VSAT.

Área local: Chamada de Wireless Local Area Network (WLAN). Exemplos: Wi-Fi (IEEE 802.11), Wi-GIG (80.11.ad), e AdHoc (IEEE 802.15.4).

Rede P2P: rede ponto a ponto (peer-to-peer) que engloba as tecnologias de identificação por radiofrequência (RFID) e near-field communication (NFC).

Rede pessoal: Chamada de Wireless Personal Area Network (WPAN). Exemplos: Bluetooth, Zigbee e Wireless Body Area Network (WBAN).

Em relação à área metropolitana, é possível aprofundar os estudos no protocolo LoRa[15] por ser apresentado como opção predominante em redes IoT no mundo todo. Esse protocolo utiliza uma plataforma de baixa potência e um longo alcance. Sua tecnologia pode ser empregada na IoT em equipamentos de fabricação, dispositivos portáteis, eletrodomésticos, iluminação de ruas, entre outros dispositivos. Normalmente, as redes LoRaWAN são dispostas em uma topologia do tipo “estrela de estrelas”, em que os gateways retransmitem mensagens entre os dispositivos finais e um servidor de rede central, e o servidor possui a responsabilidade de rotear os pacotes de cada dispositivo para o servidor de aplicação associado. O protocolo LoRaWAN possui criptografia simétrica e utiliza chaves de sessão derivadas de chaves-raiz do dispositivo. No backend,

o armazenamento das chaves-raiz e as operações de derivação de chave associadas são asseguradas por um servidor de associação, como representado na Figura 2.3.

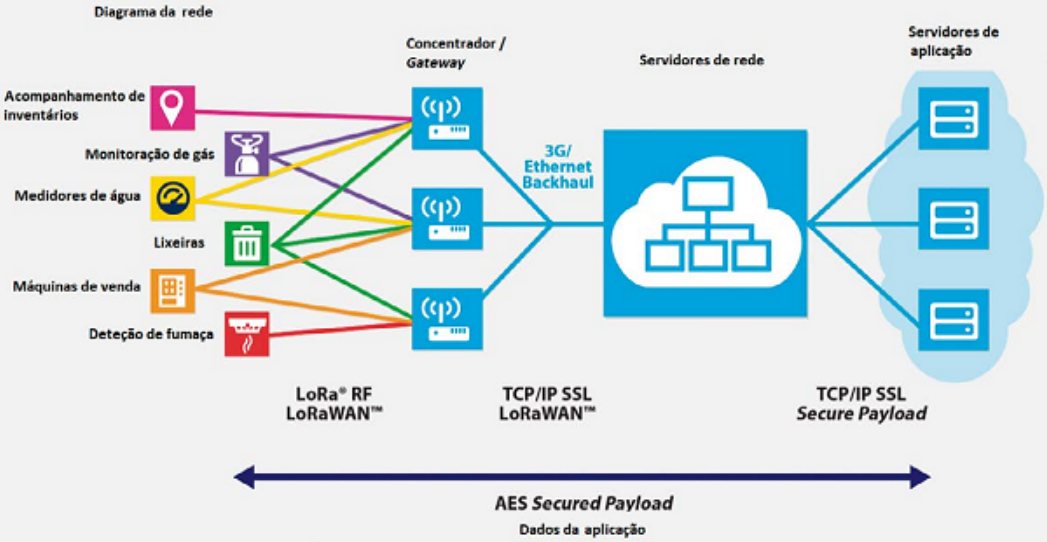

Figura 2.3. Tecnologia LoRa em IoT.
Fonte: 3glteinfo.com/lora/lora-architecture/, 2020. [41]

As principais tecnologias utilizadas para a conectividade IoT nos diversos tipos de rede são (LEITE; MARTINS; URSINI, 2017) [15]:

Redes de sensores: São responsáveis por gerar informações básicas e específicas (pressão, temperatura, umidade, luminosidade) na rede IoT. Para permitir a troca de informações fixas ou móveis ad hoc, a responsabilidade fica a cargo desses sensores de rede. Existem sensores capazes de gerar informações em conjunto com outros sensores, e sensores embutidos em etiquetas RFID.

Redes ad hoc: o nome ad hoc é uma expressão latina, que significa literalmente "para isso", "para essa finalidade específica", em oposição a algo pré-concebido, pré-padronizado. São redes formadas em diversos nós, graças a protocolos adaptativos e roteamento dinâmico. Podem ser classificadas como móveis (Mobile Ad-hoc Networks - MANET), veiculares (Vehicular Ad-hoc Networks - VANETs), para smartphones (Smatphone Ad-hoc Networks - SPANs), baseadas na internet (Internet-based MobilAdhoc Networks - iMANETs) e militares ou táticas (MANETs).

Tecnologia RFID: identifica e rastreia etiquetas em objetos a partir de campos magnéticos. É composta por aplicações, mediador, leitores, antenas e etiquetas. Seu armazenamento é feito em uma tag com 96 bits de dados, que pode ser ativa ou passiva e assistida com bateria. Seu funcionamento ocorre a partir de um sinal de rádio emitido por

um leitor e recebido pela tag, que envia suas informações de identificação. Após essa fase, os dados são enviados ao mediador para serem filtrados e agrupados, sendo então repassados a uma aplicação.

NFC: Apresenta um conjunto de tecnologias sem fio de curta distância (menor que 10 centímetros) com padrão ISO/IEC 18000-3. Seu uso ocorre principalmente em recursos de pagamentos móveis utilizando smartphones. Permite uma comunicação segura e sem fio entre dispositivos compatíveis. Entre os dispositivos usados, estão cartões de bilhetes eletrônicos, crachás, telefones celulares, tablets e quaisquer outros dispositivos que possuam um chip NFC.

Bluetooth: É um sinal de radiofrequência por difusão a curtas distâncias, com flexibilidade e alta qualidade. Utiliza o padrão IEE 802.15.1.

ZIGBEE: Responsável pelas funções de Address Translation, Packet Segmentation and Profiles e Network Routing. Utiliza o padrão IEE 802.15.4.

Wi-Fi: É um sinal de radiofrequência por difusão, flexível, de alta qualidade e médias distâncias. Possui um upgrade voltado a altas taxas de comunicação, chamado de WI-GIG. Utiliza o padrão IEEE 802.11.b, g, n.

IPV6: Padrão de endereçamento TCP/IP de 128 bits. É a evolução do IPV4 (antecessor do IPV6), para permitir maior quantidade de endereçamento, frente ao surgimento de abundância de dispositivos conectados.

Mobile 4G (LTE) e Mobile 5G: São redes de celulares móveis utilizadas pela maior parte da população para busca de informações na nuvem, com uso de protocolos do tipo HTTP ou HTTPS (com segurança e criptografia).

2.1.4 - Protocolos de Comunicação

Segundo Tripathy e Anuradha (2018) [16], padrões devem ser estabelecidos para fornecimento de segurança e proteção aos dispositivos no ambiente de IoT. Com isso, protocolos são implementados para garantir essa comunicação segura com diversos tipos de software, desenvolvedores e soluções, além de impedir fragmentações dos recursos de TI, minimizando o risco de ameaças à segurança. A fragmentação pode acontecer devido à heterogeneidade na tecnologia para IoT, relacionada com o grande volume de fornecedores e padrões no desenvolvimento dos dispositivos, fazendo com que cada um utilize seu próprio protocolo para atender a seus propósitos.

Em sua grande parte, os ataques cibernéticos acontecem direcionados as fraquezas dos protocolos da camada de rede e de aplicação, explorando sua vulnerabilidade. Os protocolos, de acordo com Delicado, Pires e Batista (2013) [17], servem para que dispositivos se comuniquem de forma interoperável e se integrem facilmente a outros dispositivos e sistemas de diferentes fornecedores, estabelecendo uma língua universal entre todas as marcas independentemente do fabricante ou modelo utilizado, permitindo acesso e controle de qualquer parte do mundo. Essa comunicação pode ocorrer por meio da plataforma de *middleware* para a IoT. Desse modo, é estabelecida uma comunicação confiável e assertiva entre os dispositivos conectados, garantindo disponibilidade e qualidade das aplicações durante as execuções.

De acordo com Takabi, Joshi e Ahn (2010) [18], a comunicação segura em IoT está voltada para o gerenciamento da informação por meio de protocolos de segurança que atuam diretamente no controle de tráfego de pacotes de dados. Técnicas de filtragem por firewalls com criptografia entre os canais de comunicação visam oferecer um ambiente seguro para os gateways de borda da rede, responsáveis pela autenticação do acesso e detecção de possíveis movimentos de invasão ou violação.

De acordo com Leite, Martins e Ursini (2019), os protocolos IoT seguem um modelo, com diferentes aplicações e referências, composto dos seguintes submodelos:

Domínio: Possui a responsabilidade de identificação e agrupamento dos sistemas.

Informação: Promove o direcionamento ou transporte das informações geradas pelos dispositivos.

Funcional: Responsável por agrupar as funções existentes nos sistemas.

Comunicação: Estabelece a comunicação dividida em sete camadas de protocolos, seguindo o modelo Open System Interconnection (OSI) da International Organization for Standardization (ISO).

Segurança, Confiança e Privacidade: Organiza, define e agrupa funcionalidades de segurança da informação, privacidade e certificação.

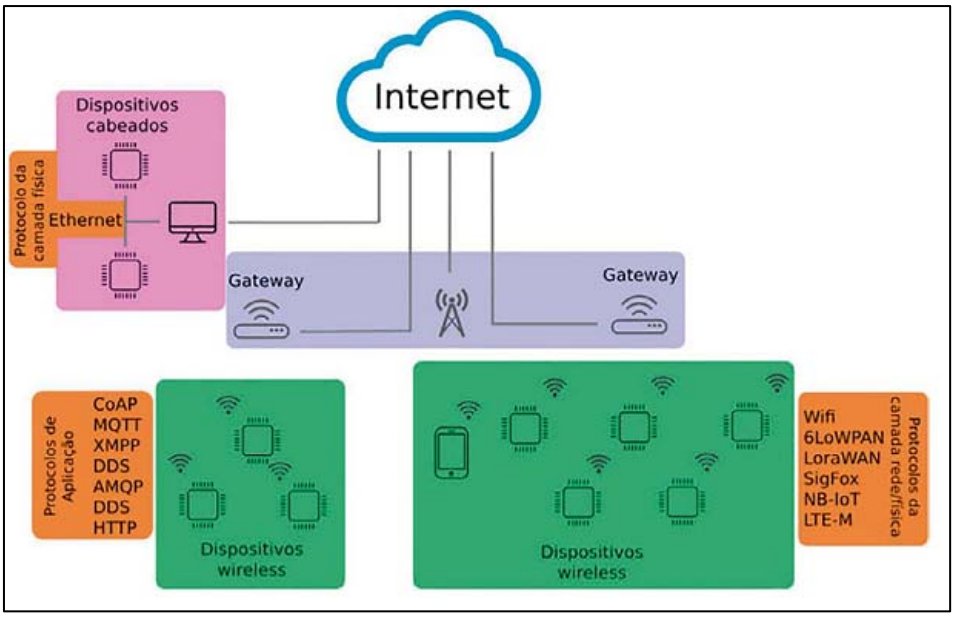

Figura 2.4 Pilhas IoT relacionadas com as camadas do modelo wireless.
Fonte: Adaptada de Palarmini, 2019. [19]

2.1.5 - Camada de aplicação

Protocolo de Aplicação Restrita (CoAp): É um protocolo projetado para aplicativos M2M e é especializado na transferência da web para uso com nós restritos e redes restritas em IoT.

Transporte de Telemetria da Fila de Mensagens (MQTT): É um protocolo leve e o mais popular para enviar fluxos de dados simples de sensores para aplicativos e *middleware*.

Protocolo Extensível de Mensagens e Presença (XMPP): É uma tecnologia XML aberta para a comunicação em tempo real, que alimenta uma vasta gama de aplicações, incluindo mensagens instantâneas, presença e colaboração.

Serviço de Distribuição de Dados (DDS): É um padrão de IoT para comunicação máquina a máquina em tempo real, escalável e de alto desempenho.

Protocolo Avançado de Enfileiramento de Mensagens (AMQP): É um protocolo de camada de aplicativo para ambientes de middleware orientados a mensagens.

Protocolo de Transferência de Hipertexto (HTTP): É a base da comunicação de dados para a World Wide Web. Hipertexto é um texto estruturado que usa links lógicos (hiperlinks) entre nós que contêm textos.

2.1.6 - Camada de rede/física:

Wi-Fi: padrão baseado em 802.11n, atualmente com uso mais comum em residências e atuante na faixa de bandas de frequências de 2,4 GHz e 5 GHz.

6LowWPAN: Utilizada como protocolo de rede de área pessoal sem fio de baixa potência.

IPv6: É uma das primeiras soluções a ser aplicada em larga escala em cima dos padrões oferecidos pelo IETF nessa categoria.

LoraWAN: Padrão voltado para aplicativos de rede de área ampla (WAN) e projetado para redes de baixa potência com comunicação bidirecional, segura e móvel, de baixo custo em IoT.

SigFox: Desenvolvida para muitos aplicativos M2M, funcionando com uma bateria pequena e exigindo apenas baixos níveis de transferência de dados e curto alcance de Wi-Fi.

NB-IoT: É um padrão de comunicação sem fio para IoT que pertence à categoria de redes de área de longa distância e baixa potência (LPWAN). Permite conectar dispositivos que precisam de pequenas quantidades de dados, baixa largura de banda e bateria de longa duração.

LTE-M: É uma nova tecnologia na área de longa distância de baixa potência (LPWA) desenvolvida para aplicações de IoT, sendo um protocolo para comunicações celulares de baixa largura de banda que se conecta à internet, dispositivos de baixa complexidade, transmitindo pequenas quantidades de dados por longos períodos, com baixo consumo de energia.

Bluetooth: É um sinal de radiofrequência por difusão a curtas distâncias, com flexibilidade e alta qualidade. Utiliza o padrão IEE 802.15.1.2.1.6 - Comparação entre os principais protocolos.

2.1.7 - Comparação entre os principais protocolos:

Nos protocolos para comunicação e transporte, destacam-se Wi-Fi, Bluetooth e LPWAN, muito procurados quando se trata de conexão e transmissão entre dispositivos. De modo geral, todos eles desenvolvem seu papel de conector, cada qual com suas características, como alcance, frequência e capacidade, obedecendo a padrões da IEEE, regulamentações e normas de segurança para consumo inteligente de energia. Já se tratando de transmissão de dados, alguns dos protocolos envolvidos são MQTT, CoAP e XMPP, que funcionam para garantir que a transmissão ocorra e seja bem-sucedida.

Agora que conhecemos os principais protocolos, vejamos uma comparação entre os principais protocolos de IoT de acordo com algumas aplicações.

O modelo de camadas para tratar as comunicações na internet absorve os protocolos para dispositivos conectados à IoT, classificando-os de acordo com suas funcionalidades. Esse processo de camadas se torna um apoio para solucionar problemas de interoperabilidade a partir da grande variedade de soluções e tecnologias de comunicação existentes.

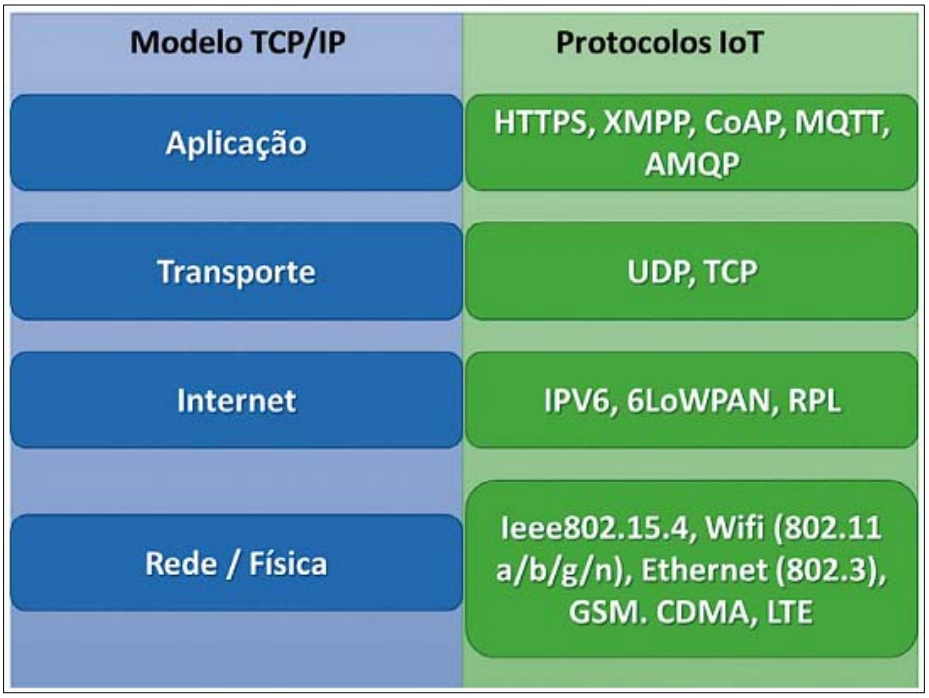

Figura 2.5. Diferença entre protocolos nas camadas ISO e IoT
Fonte: Adaptada de Sofia, 2019. [23]

Ao examinarmos, então, alguns protocolos de IoT relacionados com a camada de rede/física, nota-se que, em cada camada, o que se destaca como mais importante são as tecnologias de redes e protocolos, como padrões de celular, Wi-Fi e Ethernet, além de casos específicos como LPWAN, Bluetooth Low Energy (BLE), ZigBee, NFC e RFID (SETHI; SARANGI), dando mais destaque aos protocolos abaixo, o qual iremos comentar no estudo de caso:

LPWAN: Desenvolvida para tecnologias de redes sem fio de longo alcance com uso de baixa potência e sem necessidade de muita largura de banda, como por exemplo, utilização de sensores sem fio.

Celulares: Dispositivos móveis da rede GSM e CDMA já não se utilizam mais, logo o foco está em 4G e 5G, utilizando os padrões e comunicação LPWAN, NB-IoT e LTE-M, devido a características de baixo consumo de banda e de energia.

ZigBee: Opera no espectro de comunicação sem fio de 2,4 GHz. Tem alcance maior que o BLE, em até 100 metros. Também possui uma taxa de dados um pouco menor que o BLE, máximo de 250 Kbps e 270 Kbps, respectivamente. Foi projetado para aplicativos de construção e automação residencial e baseado no IEEE 802.15.4.

RFID: Tecnologia que trabalha com identificação por radiofrequência. Etiquetas RFID anexadas a dispositivos IoT armazenam dados e devem ser escaneadas por leitoras específicas, com alcance típico inferior a um metro. Tecnologia bastante utilizada em rastreamento de itens de inventário em aplicativos de IoT industriais e de varejo, além de sistemas de pagamentos.

Assim, a IoT é constituída por uma variedade de protocolos, muitos dos quais atuam nos dispositivos para reduzir custos e consumo de energia. A Tabela 2.1 apresenta uma comparação entre alguns desses protocolos na camada de aplicação.

Tabela 2.1. Comparação entre protocolos IoT na camada de aplicação

Protocolos	UDP / TCP	Arquitetura	Segurança e QoS	Tamanho do cabeçalho (bytes)	Comprimento máximo (bytes)
MQTT	TCP	Pub / Sub	Ambos	2	5
AMQP	TCP	Pub / Sub	Ambos	8	–
CoAP	UDP	Req / Res	Ambos	4	20 (típico)
XMPP	TCP	Ambos	Segurança	–	–
DDS	TCP / UDP	Pub / Sub	QoS	–	–

Fonte: Adaptado de Salman, 2015. [20]

Mediante as comparações realizadas entre o modelo da TCP/IP com os protocolos IoT, podemos verificar que existem áreas que precisam de blindagem com a camada de rede que tem como foco principal, em termos de segurança, a gestão de segurança da rede, ou seja, deve haver preocupação em ter mecanismos para evitar congestionamentos e protocolos de autenticação de dispositivos, e a camada de aplicação tem o principal foco de segurança na proteção dos dados processados mediante encriptação, autenticação, detecção de intrusos e possíveis distorções dos dados recolhidos ou processados. No próximo capítulo descreveremos tipos de vulnerabilidades em rede IoT.

2.2.1 - Tipos de Ataques

A definição dos tipos de ataques é necessária pelo fato de que, a partir dela, é possível classificar cada um dos ataques que foi realizado contra o alvo e, possibilita definir, também, os pilares da segurança da informação que são afetados. Conforme o trabalho de Akram, Konstantas e Mahyoub (2018) [24] é possível separar os ataques à Internet das Coisas em quatro tipos:

Ataques Físicos: São ataques que possuem como alvo o hardware, ou seja, componentes, como, por exemplo, tags RFID, microcontroladores, atuadores e sensores.

Ataques a Protocolos: Esse tipo de ataque é voltado aos protocolos os quais a Internet das Coisas utiliza, sejam tais protocolos da camada de conectividade, rede, roteamento, aplicação e de transporte.

Ataques aos Dados: É um tipo de ataque que possui como foco apenas os dados localizados tanto no componente ou na nuvem.

Ataques ao Software: São voltados a aplicações localizadas em componentes de Internet das Coisas, firmware, sistemas operacionais, gateway da aplicação e serviços.

Após essa classificação é necessário definir quais são os pilares da segurança da informação os quais tais ataques podem acabar infringindo:

Confidencialidade: O processo de comunicação segura existe para garantir que as mensagens sejam compreendidas somente pelo remetente e pelo destinatário.

Integridade: O processo que assegura que o conteúdo dos dados comunicados não tenha sido alterado durante a transmissão.

Não-repudio: O processo no qual um sistema de Internet das Coisas pode validar o incidente ou não de um evento.

Disponibilidade: É a garantia de que os dados estejam disponíveis a qualquer tempo e garante a prestação contínua do serviço.

Privacidade: O processo no qual um sistema de Internet das Coisas segue regras ou políticas de privacidade e permite que os usuários controlem seus dados sensíveis.

Auditabilidade: Garantir a capacidade de um sistema de Internet das Coisas de executar o monitoramento firme de suas ações.

Confiabilidade: A garantia da capacidade de um sistema de Internet das Coisas de remover a identidade e confirmar a confiança em terceiros.

Após isto, é possível classificar alguns ataques e definir quais pilares podem ser atingidos.

2.2.2 - Ataques Físicos

Os ataques físicos são aqueles que possuem como alvo o hardware, ou seja, componentes que estão presentes no ambiente. A lista a seguir apresenta os principais ataques deste tipo.

Ataques de replicação de objetos: um adversário primeiro captura fisicamente apenas um ou dois legítimos, depois clona ou replica os que fabricam essas réplicas com a mesma identidade (ID) com o nó capturado e, finalmente, implanta um número de clones crítico por toda a rede. (KHAN et al., 2013) [25].

Trojan de Hardware: uma alteração ou inclusão maliciosa em um circuito integrado (CI) que alterará sua função pretendida ou fará com que ela execute uma função

maliciosa adicional (ROONEY; SEEAM; BELLEKENS, 2018).[26] (CHAKRABORTY; NARASIMHAN; BHUNIA, 2009) [27].

Object Jamming: É definido como a interrupção das comunicações sem fio existentes, aumentando a razão sinal e/ou ruído no lado do receptor, através da transmissão de sinais sem fio interferentes (OSANAIYE; ALFA; HANCKE, 2018). [28] ,(GROVER; LIM; YANG, 2014) [29].

Engenharia Social: É o ato de tirar proveito das vítimas para obter informações confidenciais, que podem ser usadas para fins específicos ou vendidas no mercado clandestino e Dark web (SALAHDINE; KAABOUCH, 2019) [30].

2.2.3 - Ataques a Protocolos

Este tipo de ataque é voltado aos protocolos os quais a Internet das Coisas utiliza, sejam esses de camada de conectividade, rede, roteamento, aplicação e de transporte. Serão apresentados os principais protocolos os quais a Internet das Coisas utiliza e a lista com os principais ataques a cada um destes.

2.2.3.1 - Wi-Fi

A lista, a seguir, apresenta os principais ataques ao protocolo Wi-Fi.

ChopChop: ele permite que um atacante criptografe as mensagens de troca sem conhecer a chave (CANEILL; GILIS, 2010) [31].

Google Replay: Ao definir o “Google.com”, como uma página inicial, um invasor pode simplesmente descobrir parte do fluxo principal do log do Google baixado toda vez que os usuários abrem o site (CANEILL; GILIS, 2010) [31].

Dictionary Attack: Um ataque de dicionário é uma técnica na qual um invasor pode violar um Wi-Fi protegido por senha, adivinhando sua senha, tentando milhões ou bilhões de possibilidades, como palavras em um dicionário (CANEILL; GILIS, 2010) [31].

2.2.3.2 - RPL

Os itens, a seguir, representam os principais ataques ao protocolo RPL.

Sinkhole attack: o objetivo do adversário é atrair quase todo o tráfego de uma área específica por um nó comprometido. (JEBA; PARAMASIVAN, 2012) [32].

Wormhole attack: um invasor grava pacotes (órbitas) em um local na rede, encapsulá-los em outro local e os transmite novamente para a rede. (JEBA; PARAMASIVAN, 2012) [32].

Sybil attack: um único nó apresenta várias identidades para outros nós na rede. (JEBA; PARAMASIVAN, 2012) [32].

Hello flooding attack: um invasor envia ou substitui os pacotes HELLO de um protocolo de roteamento de um nó para outro com mais energia. (JEBA; PARAMASIVAN, 2012) [32].

2.2.3.3 - TCP-UDP

A lista a seguir apresenta os principais ataques ao protocolo TCP-UDP.

UDP flood: um invasor envia, aleatoriamente, inúmeros pacotes UDP a diferentes portais para forçar objetos a enviar pacotes ICMP de volta, o que pode tornar alguns deles inacessíveis (KUMARASAMY; GOWRISHANKAR, 2012) [33].

TCP Hijacking: um invasor seleciona e adivinha os números de sequência e as somas de verificação das entidades comunicadas. Em seguida, o invasor pode injetar um pacote TCP malicioso que contém os números de soma e de sequência esperados pelo destinatário, que não possui um mecanismo para validar a origem do pacote, considerando-o legítimo (ZHENG; POON; BEZNOSOV, 2009) [34].

TCP SYN flooding: Esse ataque consiste em um conjunto de pacotes TCP SYN espionados e direcionados à porta da vítima. Servidores da Web, como de correio, FTP e os objetos conectados, são vulneráveis a esse ataque (KUMARASAMY; GOWRISHANKAR, 2012) [33].

Uma típica rede IoT é geralmente caracterizada pela monitorização e gestão de uma abundância de diferentes serviços e dispositivos, que podendo estar geograficamente separados, têm de adquirir, processar e atuar conforme os dados recolhidos pelos seus nós finais. Devido à sua heterogeneidade de recursos, os sistemas IoT devem ter uma preocupação significativa que não consegue ser respondida por soluções mais tradicionais, visto que estas não estão preparadas para um número grande tanto de dispositivos distintos como de diferentes plataformas computacionais, cada um com os seus problemas distintos, relativamente a memória e limitações de processamento. Isto

significa que uma rede IoT terá muitos pontos de acesso vulneráveis a acesso indesejado ou *exploits*. As redes IoT têm cinco grandes tipos de problemas (ver Figura 2.6), que representam ameaças a um ambiente estável em termos segurança, privacidade e confidencialidade:

Confiança do Ambiente: Este tipo de vulnerabilidades refere-se a ameaças causadas no próprio *hardware* ou mesmo no ambiente físico circundante deste. Insere-se aqui ataques que podem variar desde alterar o funcionamento do dispositivo (apontar uma câmara para um ângulo diferente, por exemplo), a forçar o dispositivo a tomar ou registar ações por movimentos, ou sons, ou até mesmo a remoção, ou vandalização do dispositivo.

Autenticações: Este assunto aborda métodos fracos ou mesmo inexistentes de autenticação. Por exemplo, muitos acessórios Bluetooth utilizam *passwords* como *0-0-0-0* ou *1-2-3-4* durante emparelhamentos, sem qualquer maneira fácil do utilizador mudar a sua palavra-passe (*passwphrase*), o que torna estes acessórios de uma grande vulnerabilidade. Métodos fracos de autenticação podem também ser mais facilmente vítimas de ataques onde se tenta adivinhar a passe, quer seja por um número gigante de tentativas (ataque *brute forcing*), ou por passes relacionadas (ataques de dicionário).

Confiança da Rede: Este tipo de problemas ocorre quando um dispositivo deposita confiança na rede a que está associado e não toma medidas de precaução como, por exemplo, métodos de autenticação. Isto significa que o dispositivo assume que o que seja ou esteja conectado à mesma rede é seguro também, o que resulta numa possível vulnerabilidade a possíveis intrusões que tenham conseguido acesso à rede.

Privilégios: Esta categoria de problemas se diz ao abuso de aplicações e serviços que cedem para certos canais de informação mais dados do que os necessários para operar. Isto pode ser observado, por exemplo, em aplicações de *smartphones* que cedem a dispositivos informações que, em uso regular, não deveriam ter acesso.

Falhas de Implementação: Finalmente, este tipo de problemas representa vulnerabilidades causadas por implementações defeituosas de *software*, como acessos garantidos por *bugs*, *exploits* ou fugas de informação (*leaks* de credenciais, por exemplo).

Figura 2.6. Diagrama das Áreas Vulneráveis da Segurança de Redes IoT.
Fonte: N. Zhang [21], 2019.

2.4 - Propriedades de Ataques

Devido à sua natureza complexa e heterogénea, uma rede IoT pode ser vulnerável a muitos diferentes ataques, sendo importante analisar o que os definem e em que consistem. Conforme o diagrama, um ataque que incide numa rede IoT pode ser classificado por 6 atributos distintos (ver Figura 2.7):

Figura 2.7: Diagrama das Características de ataque a uma rede IoT
Fonte: N. Zhang [21], 2019.

Camada Alvo: Como já foi visto no capítulo 2 uma rede IoT é composta por diferentes *layers(camadas)*, o que significa que um ataque irá sempre incidir sobre um dispositivo ou serviço presente numa, ou mais camadas da rede. Portanto, a primeira

propriedade de um ataque que se consegue identificar é a camada cujo ataque terá como alvo.

Dispositivo Alvo: O segundo atributo que é possível identificar é o dispositivo ou o conjunto destes que uma intervenção maliciosa tentará tirar partido. Um ataque pode ter como alvo tanto um único sensor como todo um grupo de dispositivos diferentes que, em conjunto, representam um sistema de uma rede, ao qual é importante definir o dispositivo vulnerável e não apenas a camada a que este pertence.

Canal de Transmissão: Outra propriedade de um ataque que se consegue observar é o meio de comunicação da invasão, ou seja, o canal usado para se ligar à rede alvo. Este canal pode assumir várias formas, quer seja como canais remotos, Wi-Fi ou Bluetooth por exemplo, como canais mais próximos de um dispositivo, como toques físicos, gestos ou sinais acústicos.

Consequências: O quarto atributo observável é a consequência direta do ataque. Uma invasão de um sistema pode provocar vários resultados na rede que visa incidir, que podem variar desde *leaks* de informação sensível, a impedimento de acesso a certos serviços da rede (DoS) ou até mesmo a controle total, ou parcial de certos elementos da rede.

Tamanho: A quinta propriedade que é possível identificar é o número de dispositivos ameaçados. Por exemplo, um intruso pode focar seus recursos num único dispositivo para prejudicar a rede, limitando o ataque a esse único dispositivo. Porém, o mesmo intruso pode também utilizar destes para influenciar outros dispositivos ligados a mesma rede, aumentando assim a escala da invasão.

Furtividade: Finalmente, a sexta e última propriedade de uma invasão é a sua capacidade de permanecer em segredo antes, durante e depois da intervenção na rede. O utilizador da rede pode ficar completamente alheio ao ataque dirigido, ter conhecimento parcial ou mesmo total sobre a intrusão e as suas consequências, geralmente devido às mudanças (ou falta destas) no ambiente regular da rede causadas pela intrusão.

2.4.1 - Exemplos de Ataques

Uma intrusão numa rede IoT pode tomar várias formas e utilizar múltiplos métodos para atingir os diferentes serviços do sistema alvo. Esta secção serve para enumerar alguns exemplos de ataques, elucidando o tipo de ataques que incidem nos diferentes níveis da arquitetura de uma rede IoT, como visto no quadro 2.1.

Quadro 2.1: Lista de Exemplos de Ataques a uma rede IoT.

Exemplos de Ataques		
Alto Nível	Médio Nível	Baixo Nível
Software Exploits	Interferência de Conexões Ataques *Sinkhole*	Ataques de Interferências Ataques Sybil de Baixo Nível Deprivação de Sono

Fonte: M. A. Khan e K. Salah, 2018.[22]

Alto Nível

Software Exploits: Para ganhar acesso ao sistema, um intruso pode tentar aproveitar-se de qualquer tipo de falha de software presente em aplicações mobile ou nos serviços web que a rede possa utilizar, como a Cloud. Este tipo de ataque, em norma, aproveita-se de vulnerabilidades causadas por atualizações defeituosas.

Médio Nível

Interferência de Conexões: Devido à necessidade de identificação única de cada dispositivo de uma rede IoT, o processo de comunicação para a atribuição dessa dita identificação é atacado. Isto pode não só fornecer a um intruso acesso a múltiplos tipos de informação sobre os dispositivos da rede, como também impedir que circulem dados entre partes do sistema, dificultando assim a utilização de certos serviços na rede afetada.

Ataques *Sinkhole*: Este tipo de intrusão distingue-se pelo desvio de informação que circule na rede para um nó externo a esta, permitindo assim que um intruso ceda a informação confidencial e tome ações contra a rede baseadas nessa informação. Este tipo de ataques geralmente resulta em violações de privacidade e confidencialidade, mas também podem ser usados para problemas mais severos como o bloqueio dos nós afetados.

Baixo Nível

Ataque de Interferência: O objetivo deste tipo de ataque é a disrupção de certos dispositivos através da emissão de certos sinais (como certas radiofrequências, por exemplo) sem sentido, ordem ou protocolo perceptíveis de maneira que este se confunda com os sinais que o dispositivo estaria suposto a receber. Isto causa confusão na coleção e análise de informação nos dispositivos afetados, afetando o fluxo de informação destes, culminado assim num comportamento defeituoso do equipamento.

Ataque *Sybil* de Baixo Nível: Este tipo de ataque planeja danificar os processos normais de funcionamento da rede através da conexão de nós maliciosos com falsas identidades. Ao gerar endereços MAC aleatórios, o intruso tenta disfarçar os seus nós como parte da rede de maneira a extinguir recursos desta, o que leva aos dispositivos legítimos terem o seu acesso à rede negado.

Depravação de Sono: Este tipo de ataque procura a disrupção do funcionamento normal de dispositivos de energia limitada. Ao forçar certos dispositivos que estariam desenhados para funcionar intermitentemente a correr procedimentos interruptamente, um intruso pode forçar um dispositivo a desligar-se temporariamente, negando assim a disponibilização do serviço do dispositivo afetado.

2.5 - Protocolo MQTT

O protocolo MQTT, assim como qualquer protocolo em redes de computadores, não é senão um conjunto de especificações padronizadas para comunicação de máquinas. Sendo assim, existe uma série de diferentes implementações do protocolo MQTT, cada qual com suas particularidades em relação a modos de armazenamento e algoritmos de decisão. O Mosquitto, projeto desenvolvido pela Eclipse Foundation, visa implementar os componentes especificados no padrão do protocolo MQTT.

O objetivo central do protocolo MQTT é permitir a comunicação entre objetos da IoT e definir o papel de cada objeto a partir de um modelo de operação. Entre as operações fundamentais está a definição do papel de cada dispositivo, a ordem de envio de mensagens e o formato das mensagens. O MQTT pode ser aplicado em diferentes soluções de IoT, como no controle de equipamentos elétricos nas indústrias, controle de equipamentos de segurança, controle de equipamentos eólicos e fotovoltaicos, entre outras aplicações (CONCEIÇÃO; COSTA, 2019) [32]. Como recurso para troca de

mensagens, o protocolo MQTT apresenta um servidor de aplicação chamado Broker. A função do Broker é servir como um sistema intermediário nas trocas de mensagens. A Figura 2.8 ilustra mensagens sendo publicadas em servidor MQTT Broker por diversos dispositivos de diferentes finalidades ao redor do mundo.

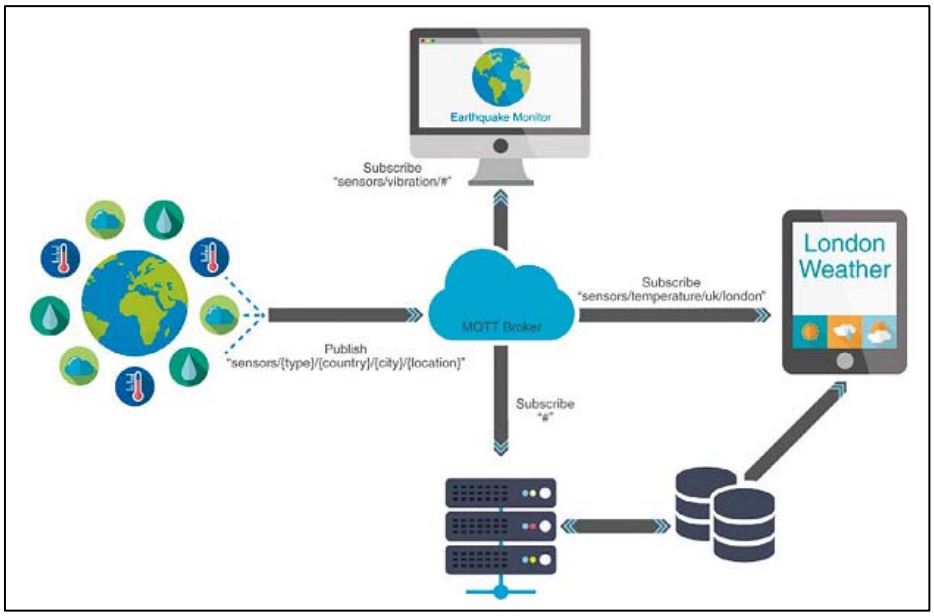

Figura 2.8 - Publicação de mensagens no MQTT Broker
Fonte: Stack Overflow [36], 2019.

Referente a Figura 2.8, esse alcance no envio de dados só é possível graças aos protocolos de comunicação em rede e dos protocolos envolvidos nos dispositivos de IoT. Nesse exemplo, é possível notar que mensagens publicadas pelos dispositivos sob protocolo MQTT podem ser acessadas via Broker a partir de aplicações web e aplicações mobile. Dessa forma, conforme a Figura 2.8, todo conteúdo compartilhado entre dispositivos de IoT são armazenados em sistemas gerenciadores de bases de dados para uma posterior análise a partir de gráficos gerados pelas aplicações desenvolvidas. Portanto, é possível afirmar que o protocolo MQTT é fundamental para a subsistência da IoT, principalmente em relação aos dados compartilhados por dispositivos sem fio através da rede de computadores.

2.5.1 – Mosquitto

O Eclipse Mosquitto é um agente de mensagens de software livre (licenciado por EPL/EDL) que implementa o protocolo MQTT versões 5.0, 3.1.1 e 3.1. Ele é leve e

adequado para uso em todos os dispositivos, desde computadores de placa única de baixa potência até servidores completos.

2.5.2 - Comunicação do Protocolo

Como já explicado, a função do MQTT Broker é atuar como intermediário no intercâmbio de mensagens entre dispositivos na rede (mais detalhes na próxima seção do capítulo). Para um cliente se conectar via Broker (responsável por receber as mensagens enviadas), inicialmente o cliente envia para o Broker a instrução CONNECT a partir de mensagem contendo diversos parâmetros de execução. Os nomes dos diferentes parâmetros e suas características são os seguintes (CONCEIÇÃO; COSTA, 2019) [32]:

KeepAlive: Representa o tempo disponível para o cliente poder permanecer no Broker e manter sua conexão ativa.

Astwillqos: O QoS da mensagem de último desejo.

Lastwilltopic: Mensagem de último desejo em um tópico no momento de encerramento da conexão.

Password e Username: Credenciais de autorização e autenticação do Broker.

Cleansession: Verifica se a conexão armazenou as mensagens que, porventura, possam ter sido perdidas e todas as assinaturas no Broker (conexão persistente).

A Figura 2.9 exibe um exemplo do uso dos parâmetros MQTT para uma conexão realizada pelo cliente via Broker em que será subscrita uma mensagem. É possível verificar na linha 6 a criação da variável keep_alive_broker, que controlará o período no qual o cliente poderá manter uma conexão ativa no servidor Broker. O método on_connect, na linha 10, é responsável por iniciar a conexão com o Broker e inscrever a mensagem (a variável topico_subscribe é passada ao método subscrite na linha 13).

```
1  import paho.mqtt.client as mqtt
2  import sys
3  #CONFIGURAÇÃO DO MQTT
4  Broker = "iot.eclipse.org" #broker publico utilizado.
5  porta_broker = 1883 #porta utilizada para comunicacao com o broker MQTT
6  keep_alive_broker = 60 #TEMPO CONFIGURADO EM SEGUNDOS do keep-alive
7  topico_subscribe = "MQTTRaspPiINCB"  #topico MQTT que o programa ira "ouvir" (fazer subscribe)
8
9  #Callback - conexao ao broker
10 def on_connect(client, userdata, flags, rc):
11     print("[STATUS] Conectado ao Broker.")
12     #faz subscribe automatico no topico
13     client.subscribe(topico_subscribe)
```

Figura 2.9 Trecho de código com conexão ao Broker, para Protocolo MQTT.
Fonte: Autor, 2023.

Já na Figura 2.10 abaixo, é possível perceber na linha 30 o método de conexão (connect) responsável pelo envio dos parâmetros (Broker, porta_ broker e o keep_alive_broker) necessários para a conexão ao Broker do MQTT.

```
16  def on_message(client, userdata, msg):
17      MensagemRecebida = str(msg.payload)
18      print("[MSG RECEBIDA] Topico: "+msg.topic+" / Mensagem: "+MensagemRecebida)
19  #programa principal:
20  try:
21          print("[STATUS] Inicializando MQTT...")
22          #inicializa MQTT:
23
24          #cria client MQTT e define funcoes de callback de conexao (client.on_connect)
25          #e recepcao de dados recebidos (client.on_message)
26          client = mqtt.Client()
27          client.on_connect = on_connect
28          client.on_message = on_message
29          #faz a conexao ao broker MQTT
30          client.connect(Broker, porta_broker, keep_alive_broker)
31          #mantem o MQTT funcionando por tempo indeterminado, sendo que todas as
32          client.loop_forever()
```

Figura 2.10 Trecho de código com passagem de parâmetro KeepAlive.
Fonte: Autor, 2023.

Em geral, o protocolo MQTT é caracterizado por conceitos básicos visando garantir a entrega das mensagens mantendo sua máxima leveza. O protocolo apresenta as seguintes características gerais (MOTA, 2017) [37].

Modelo publish/subscribe: é baseado no princípio de publicar mensagens e assinar tópicos.

Tópicos e assinaturas: As mensagens no MQTT são publicadas para tópicos, que podem ser entendidos como áreas de interesse. Os tópicos costumam ser formados por palavras separadas por uma barra (/) e que se assemelham a caminhos.

QoS: O MQTT apresenta três níveis de qualidade de serviço, cada qual indicando o esforço do servidor para garantir a entrega da mensagem. São estabelecidos os seguintes níveis de QoS.

2.5.3 - Estrutura do protocolo MQTT

O protocolo MQTT foi desenvolvido para permitir que sua infraestrutura interaja com os protocolos TCP/IP em projetos que possuam menor taxa de transferência na rede e com hardware leve e simples.

O Protocol Data Unit (PDU) do protocolo MQTT é encapsulado pelo protocolo Transmission Control Protocol (TCP), ou seja, os dados e o cabeçalho do MQTT são enviados na área de dados do TCP. O MQTT é um protocolo de mensagem com suporte para a comunicação assíncrona entre as partes (MENEZES et al., 2017) [34]. Segundo Conceição e Costa (2019) [32], a distribuição dos protocolos nas camadas pode ser esquematizada da seguinte forma:

- **Camada de aplicação:** MQTT;
- **Camada de transporte:** TCP;
- **Camada de rede:** IPv4/IPv6;
- **Camada Internet ou Intra-rede:** Ethernet, Wi-Fi.

Assim, o protocolo MQTT é responsável por definir o modelo de operação entre os equipamentos, especificando seus papéis, como serão dispostas as mensagens e a ordem entre elas. As funcionalidades da rede são mantidas pelas camadas TCP, pelo protocolo IP e por outras tecnologias de comunicação na camada inferior dessa arquitetura (como Wi-Fi, Ethernet, entre outras). Com o uso do protocolo IP, os dados podem ser transmitidos pela internet. Na Figura 11 abaixo, é possível verificar o fluxo de comunicação que se estabelece entre o elemento que publica informações (neste caso, sensores que medem a temperatura) e aquele que assina para receber tais informações (neste caso, dispositivos [clientes] representados por aplicações que rodam na nuvem ou por pequenos computadores com baixo poder de processamento chamados de Raspberry, ou Arduino).

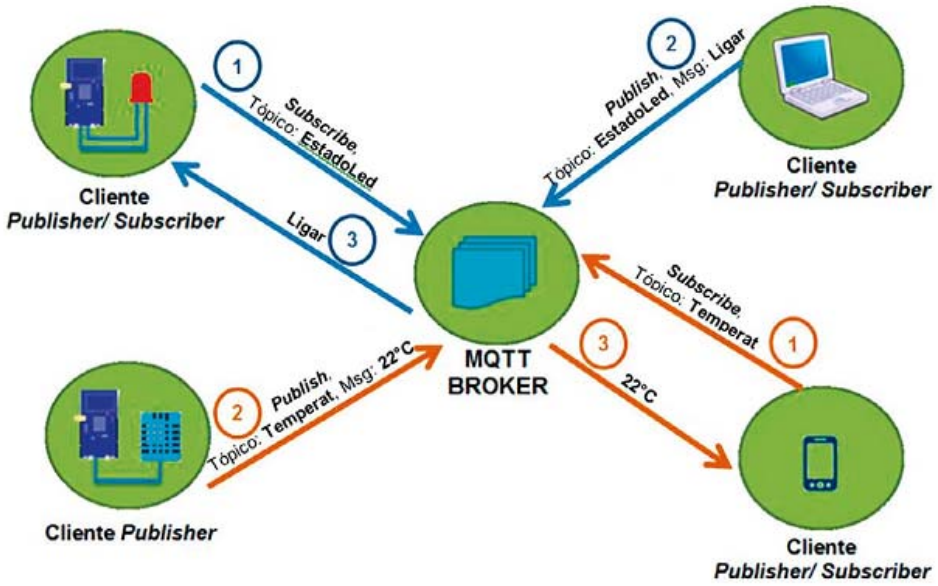

Figura 2.11, Fluxo de comunicação entre elementos com o protocolo MQTT.
Fonte: Silva e Ledel, 2019. [40]

Baseado na ilustração acima, construiremos um protótipo baseado nesta descrição, pegaremos uma parte desta figura e faremos nosso projeto similar, conforme imagem abaixo:

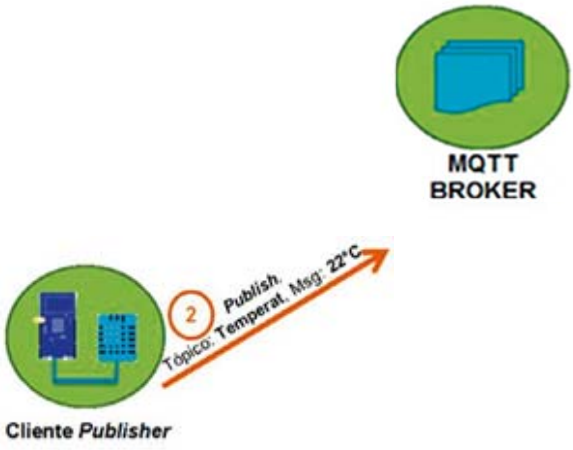

Figura 2.12. Exemplo de um Conexão Publisher Broker MQTT.
Fonte: Silva e Ledel, 2019. [40]

CAPÍTULO 3

Construindo um prototipo IoT

3.1- Problema de Segurança em projetos residenciais de IoT

Nos aspectos de segurança da informação, que desafios existem quando pensamos em sistemas IoT. O principal problema relacionado a segurança da informação são os meios de acesso, que estão relacionados por Wi-Fi, rede cabeada, autenticação e porta de acesso, que podem afetar sua privacidade e que está diretamente relacionada ao seu ambiente residencial. Por isso explicaremos na construção deste protótipo, que ao se adquirir um dispositivo IoT para automação residencial, estará passível de sofrer os mesmos ataques que ocorrem em grandes ambientes corporativos.

3.2 - Problemas destes ataques

O hardware que usamos neste projeto vem com uma placa Wi-Fi integrada ao NodeMCU, ao qual pode receber o mesmo tipo de ataques e vulnerabilidade que ocorrem em tecnologia da informação, baseada nas mesmas camadas de acesso tanto em rede, como a camada de enlace, e pode receber todos os tipos de ataque relacionado a rede Wi-Fi, como WLAN *Scanners*, *Man in the Middle*, IP *spoofing*, etc. Esses ataques também podem ser direcionados a rede cabeada, então se o dispositivo IoT residencial estiver programado para usar esse meio de comunicação interna, teremos a real certeza, de que poderá ser atacado.

Outro ponto importante que afeta projetos desta natureza é a forma de autenticação feita ao dispositivo IoT, pois provavelmente estará na mesma rede interna da residência, mesmo aplicando métodos de mascaramento de IP, sua integridade poderá ser comprometida. Outro fato importante é que toda plataforma IoT está diretamente ligada ao ambiente Cloud, então dessa forma não teremos como evitar passar por um processo de autenticação, seja por fator de conhecimento, fator de inferência, ou propriedade.

3.3 - Solução para esses Ataques

Pegando como exemplo o processo de autenticação podemos trazer uma solução para esse problema, fazer com que aplicação tenha um meio de autenticação confiável, aqui neste projeto ao utilizamos o protocolo MQTT, podemos efetuar uma chamada de autenticação segura usando o token, que pode ser gerado pela própria plataforma Cloud, na qual seria uma opção segura no processo de autenticação, pois a plataforma não apresenta uma forma de autenticação de múltiplos fatores, e sim autenticação de um único fator que se dá por um token gerado aleatoriamente, e inserido na chamada pela linha de código ao executar a conexão pelo MQTT, através da porta 1883 conforme figura 3.1.

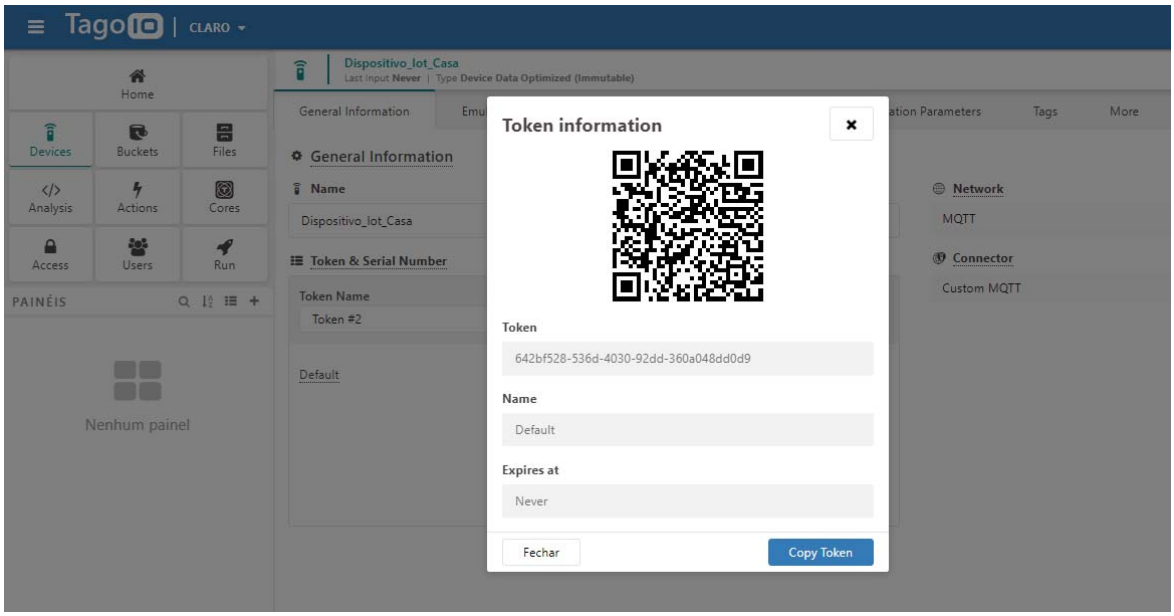

Figura 3.1 Token de autenticação do dispositivo IoT.
Fonte: Autor, 2023.

```
/* MQTT definitions */
#define MQTT_PUB_TOPIC "tago/data/post"
#define MQTT_USERNAME  "Dispositivo_Iot_Casa"  /* Coloque aqui qualquer valor */
#define MQTT_PASSWORD  "642bf528-536d-4030-92dd-360a048dd0d9"  /* coloque aqui o Device Token do seu dispositivo no Tago.io */

/* WIFI */
const char* ssid_wifi = "Marcelo Cavalcanti";     /*  WI-FI network SSID (name) you want to connect */
const char* password_wifi = "kizg9356"; /*  WI-FI network password */
WiFiClient espClient;

/* MQTT */
const char* broker_mqtt = "mqtt.tago.io"; /* MQTT broker URL */
int broker_port = 1883;                   /* MQTT broker port */
PubSubClient MQTT(espClient);
```

Figura 3.2 processos de autenticação.
Fonte: Autor, 2023.

3.4 - Protótipo

Dos protocolos citados neste projeto, construiremos um protótipo em Arduino IDE, usando MQTT da camada de aplicação, através desse protótipo pretendemos mostrar todo processo de construção, desde o hardware utilizado pelo protótipo até autenticação na plataforma Cloud, descreveremos o processo de instalação das bibliotecas de acesso ao hardware, ilustraremos algumas linhas de código desenvolvidas em phyton, as informações coletadas pelo sensor serão apresentas em forma de Dashboard, em Cloud. Neste primeiro momento visa-se automatizar os dispositivos de maneira que informe as condições do ambiente local, como temperatura e umidade, por uma rede interna residencial pelo acesso Wi-Fi, e descrever as vulnerabilidades que ocorrerem neste ambiente, utilizando o protocolo TCP que oferece ao MQTT o serviço de um serviço de conexão e entrega de mensagens.

O dispositivo funcionará capturando os dados referente a medição de temperatura e unidade do ar e os enviando, a cada n segundos, para uma nuvem, apresentando-os em um dashboard. A seguir, analisaremos condições de segurança aplicáveis ao dispositivo IoT.

Para construção deste protótipo serão necessários os seguintes componentes abaixo:

- Um NodeMCU (ESP8266 12-E)
- Um cabo micro-USB (programação e alimentação)
- Um sensor DHT22 (sensor de temperatura e umidade relativa do ar)
- Um resistor de 10K
- Um protoboard de 400 pontos
- Jumpers macho-macho

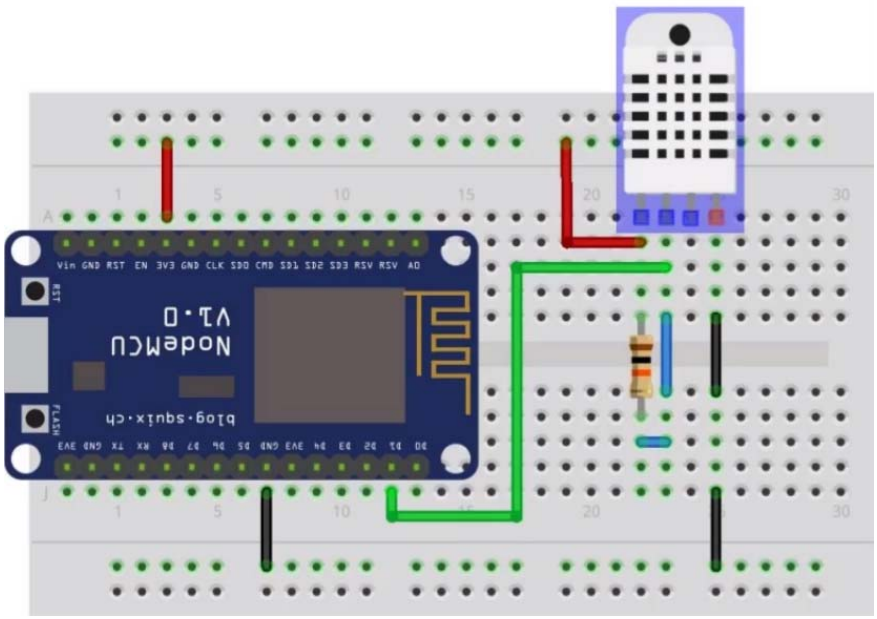

Figura 3.3, representatividade do Esquema elétrico do protótipo IoT
Fonte: Adaptado de Ferramenta Fritizing, 2022. [43]

Figura 3.4 NodeMCU modelo ESP8266
Fonte: Adaptado de (FILIPEFLOP, 2022).[44]

Figura 3.5, Sensor DHT22
Fonte: Adaptado de (FILIPEFLOP, 2022). [44]

3.4.1 - Processo de instalação de Bibliotecas

Por recomendação do fabricante do microcontrolador, é necessário que as bibliotecas estejam instaladas corretamente para criação do protótipo, utilizemos como ferramenta de teste o Arduino IDE para programação e depuração do protótipo. Após a instalação da ferramenta Arduino IDE, implantaremos a biblioteca do NodeMCU 1.0 ESP, conforme figura abaixo 3.6.

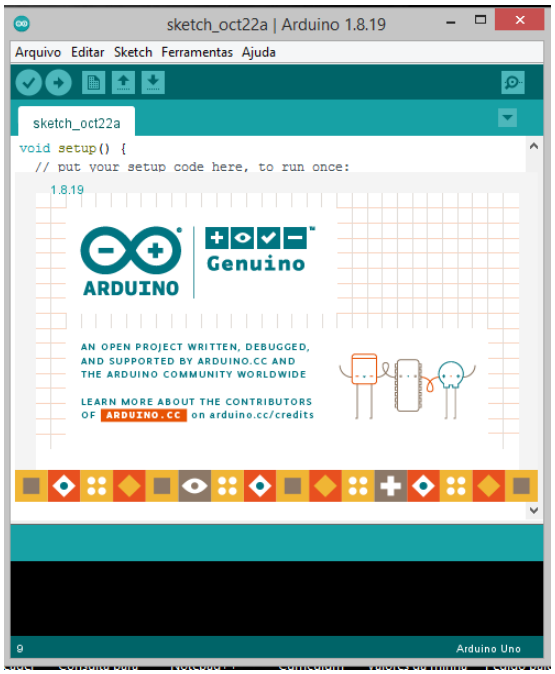

Figura 3.6 Arduino IDE
Fonte: https://www.arduino.cc/en/software, 2023. [42]

O procedimento abaixo se refere a instalação da biblioteca que irá fazer as chamadas do hardware NodeMCU, conforme recomendado pelo fabricante como boa prática. É sempre bom instalar a última versão do produto, pois as correções podem ser tratadas em tempos de execução, evitando assim possíveis falhas, neste caso as correções já foram aplicadas.

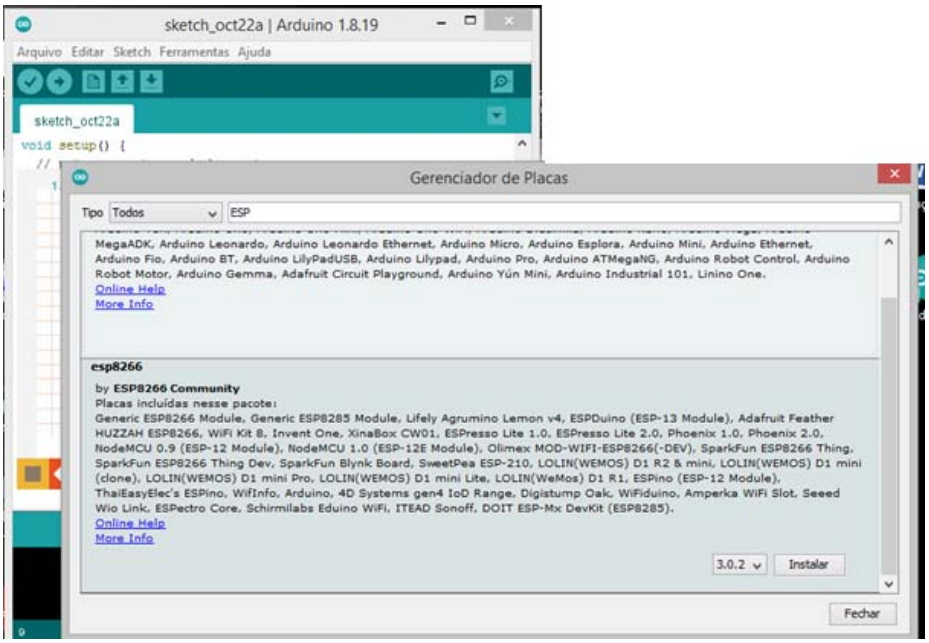

Figura 3.7 - Instalando Lib. ESP8266
Fonte: Arduino IDE, 2023. [42]

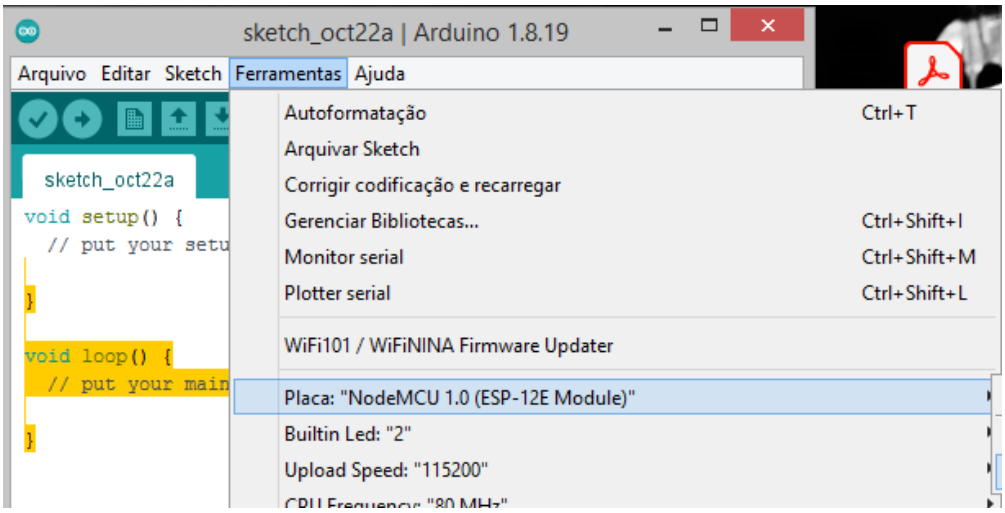

Figura 3.8, hardware liberado para operar com NodeMCU
Fonte: Arduino IDE, 2023. [42]

Para que as comunicações chamadas de envio de mensagem pelo broker sejam realizadas, é necessário instalar as bibliotecas do protocolo MQTT, que é o responsável pelo teste de conexão com o NodeMCU e o TAGO.IO, usando protocolo TCP da camada de transporte do modelo OSI, através do meio físico Wi-Fi. As bibliotecas são ArduinoJson, PubsubClient (MQTT) e homie-esp8266, que sempre solicitam a última versão. Por fim a instalação da biblioteca do sensor DHT22 é que será responsável pela coleta dos dados referente a temperatura e umidade do Ar.

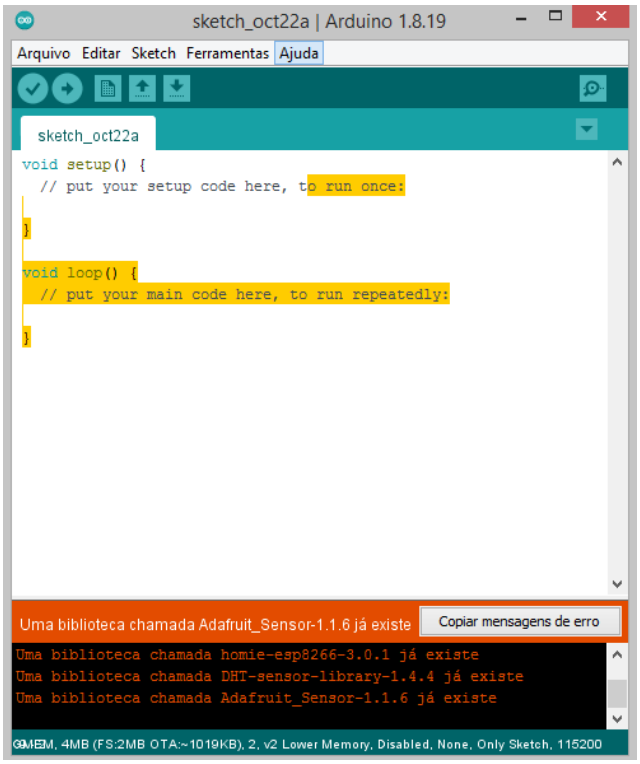

Figura 3.9, Instalação da Library DHT22- sensor e Connon Sensor Library.
Fonte: Arduinos IDE, 2023. [42]

3.4.2 - Implementação da Camada de aplicação

Conforme observado pela figura 3.8, o dispositivo responsável pela medição e envio da temperatura e umidade do ar para plataforma TAGO.IO, é composto pela ligação da placa NodeMCU com um sensor DHT22. Descreveremos o processo de código realizado em linguagem phyton, exemplificando abaixo suas linhas conforme figura 3.10.

Descrevendo a linha de código no Arduino IDE, são inclusos head de bibliotecas como Studio da linguagem padrão de C e C++, ESP8266 Wi-Fi para acesso pela conexão Wi-Fi do NodeMCU e o PubSubClient para utilização do protocolo MQTT, o qual será usado para enviar dados para a plataforma TAGO.IO, por meio de envio de mensagem TCP ao Arduino. O Json terá a função de formatação do payload da mensagem MQTT para a plataforma TAGO.IO, e por fim, o resultado pelo head DHT (sensor de temperatura e umidade). Descrição do código ver Anexo 1, conforme figura 3.10.

```
/*
 * Programa: código-fonte do dispositivo IoT prototipo
 * Autor: Marcelo Cavalcanti da Costa
 */
#include <stdio.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <DHT.h>

/* Definicoes gerais */
#define TEMPO_ENVIO_INFORMACOES    5000 //ms

/* Definicoes do sensor de temperatura */
#define DHTPIN  D1   /* GPIO que o pino 2 do sensor é conectado */

/* A biblioteca serve para os sensores DHT11, DHT22 e DHT21.
   No nosso caso, usaremos o DHT22, porém se você desejar utilizar
   algum dos outros disponíveis, basta descomentar a linha correspondente.
*/
//#define DHTTYPE DHT11   // DHT 11
#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321
//#define DHTTYPE DHT21   // DHT 21 (AM2301)
```

Figura 3.10 Linguagem de programação phyton.
Fonte: Autor, 2023.

Descrevendo a linha de código da chamada ao protocolo MQTT, mqqt_pub_topic, que fixa a informação como tago/data/post em conjunto com as informações a serem identificadas pelo device token, composto pelo mqtt_username e mqtt_password, será necessário informar o token de autenticação gerado pela ferramenta TAGO.IO, conforme figura 3.14, em seguida instala-se a lib. do broker, com a função de atuar como intermediário no processo de mensagens entre dispositivos e Cloud, a broker_port também é uma parte importante no processo de autenticação, que realiza a comunicação da camada de rede fazendo a chamada pela PubsubClient, fazendo com que objeto de conexão Wi-Fi troque mensagens MQTT. Descrição, código Anexo 1, conforme figura 3.11.

```
/* MQTT definitions */
#define MQTT_PUB_TOPIC "tago/data/post"
#define MQTT_USERNAME  "Dispositivo_Iot_Casa"  /* Coloque aqui qualquer valor */
#define MQTT_PASSWORD  "642bf528-536d-4030-92dd-360a048dd0d9"  /* coloque aqui o Device Token do seu dispositivo no Tago.io */

/* WIFI */
const char* ssid_wifi = "Marcelo Cavalcanti";     /*  WI-FI network SSID (name) you want to connect */
const char* password_wifi = "kizg9356"; /*  WI-FI network password */
WiFiClient espClient;

/* MQTT */
const char* broker_mqtt = "mqtt.tago.io"; /* MQTT broker URL */
int broker_port = 1883;                   /* MQTT broker port */
PubSubClient MQTT(espClient);
```

Figura 3.11. Configuração de Wi-Fi, porta e processo de autenticação.
Fonte: Autor, 2023.

E por último a Lib. do objeto DHT que faz a identificação do sensor sobre o pino, que está ligado no NodeMCU. Explicação da descrição Anexo 1, conforme figura 3.12.

```
/* objeto para comunicação com sensor DHT22  *
DHT dht(DHTPIN, DHTTYPE);
```

Figura 3.12, chamada de comunicação do sensor DHT22.
Fonte: Autor, 2023.

A descrição do código de iteração entre a camada de aplicação MQTT e o acesso à base de dados para gravação das informações em um ambiente em nuvem (Cloud) sobre a plataforma TAGO.IO. Com isso, executaremos a chamada função de conexão, usando broker para iniciar o processo de troca de mensagens, conforme figura 3.13.

```
void init_MQTT(void)
{
    MQTT.setServer(broker_mqtt, broker_port);
}
```

Figura 3.13 inicializando protocolo MQTT
Fonte: Autor, 2023.

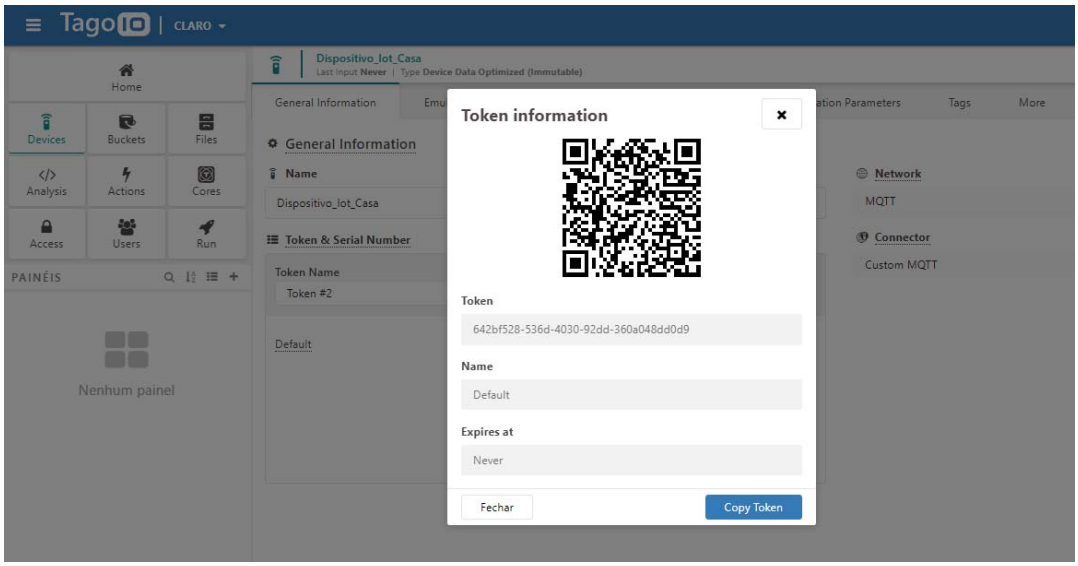

Figura 3.14, informa o Token de autenticação do dispositivo IoT.
Fonte: Autor, 2023.

Após início da conexão sobre protocolo MQTT, podemos notar a chamada da função “randomSeed (random (9999)” Que verifica se existem mais de uma chamada para aquela conexão já existente, caso exista por segurança a conexão em concorrência será interrompida, através do” mqqt_id_randomico”. Se a conexão foi realizada com sucesso irá informar “Broker MQTT conectado com sucesso”, caso contrário irá falhar ao tentar conectar. Código Anexo 1, Conforme Figura 3.15.

```
void connect_MQTT(void)
{
    char mqtt_id_randomico[5] = {0};

    while (!MQTT.connected())
    {
        Serial.print("* Tentando se conectar ao broker MQTT: ");
        Serial.println(broker_mqtt);

        /* gera id mqtt randomico */
        randomSeed(random(9999));
        sprintf(mqtt_id_randomico, "%ld", random(9999));

        if (MQTT.connect(mqtt_id_randomico, MQTT_USERNAME, MQTT_PASSWORD))
        {
            Serial.println("Conectado ao broker MQTT com sucesso!");
        }
        else
        {
            Serial.println("Falha na tentativa de conexao com broker MQTT.");
            Serial.println("Nova tentativa em 2s...");
            delay(2000);
        }
    }
}
```

Figura 3.15, Processo de Validação de autenticação.
Fonte: Autor, 2023.

Concluído o processo de autenticação, trataremos agora da forma de acesso à base dos dados do TAGO.IO, através da chamada das funções abaixo, para efetuar o acesso à plataforma TAGO.IO, sobre o protocolo MQTT, tivemos que passar as informações referentes ao JSON a serem enviadas ao TAGO.IO, como “variable”, “unit” e “Value”, conforme descrito ver Anexo 1 código, figura 3.16;

```
/* Funcao: envia informacoes para plataforma IoT (Tago.io) via MQTT
 * Parametros: nenhum
 * Retorno: nenhum
 */

/*
JSON a ser enviado para Tago.io:

{
    "variable": "nome_da_variavel",
    "unit"    : "unidade",
    "value"   : valor
}
*/
void send_data_iot_platform(void)
{
   StaticJsonDocument<250> tago_json_temperature;
   StaticJsonDocument<250> tago_json_humidity;
   char json_string[250] = {0};
   float temperatura_lida = dht.readTemperature();
   float umidade_lida = dht.readHumidity();
   int i;
```

Figura 3.16, código fonte.
Fonte: Autor, 2023.

```
   /* Imprime medicoes de temperatura e umidade (para debug) */
   for (i=0; i<80; i++)
       Serial.println(" ");

   Serial.println("----------");
   Serial.print("Temperatura: ");
   Serial.print(temperatura_lida);
   Serial.println("C");
   Serial.print("Umidade: ");
   Serial.print(umidade_lida);
   Serial.println("%");

   /* Envio da temperatura */
   tago_json_temperature["variable"] = "temperatura";
   tago_json_temperature["unit"] = "C";
   tago_json_temperature["value"] = temperatura_lida;
   memset(json_string, 0, sizeof(json_string));
   serializeJson(tago_json_temperature, json_string);
   MQTT.publish(MQTT_PUB_TOPIC, json_string);

   /* Envio da umidade */
   tago_json_humidity["variable"] = "umidade";
   tago_json_humidity["unit"] = "%";
   tago_json_humidity["value"] = umidade_lida;
   memset(json_string, 0, sizeof(json_string));
   serializeJson(tago_json_humidity, json_string);
   MQTT.publish(MQTT_PUB_TOPIC, json_string);
}
```

Figura 3.16.1 Envio dos dados coletados pelo Sensor DHT22.
Fonte: Autor, 2023.

Essa parte, figura 3.16.1, trata da formatação e preenchimento do valor correto identificados pelo sensor DHT22 e enviados pelo protocolo MQTT por brokers para serem armazenados na plataforma Cloud TAGO.IO. Estes dados enviados serão tratados pela ferramenta na plataforma, recebendo em tempo real as distintas informações, tanto para temperatura como unidade do ar. Conforme figura 3.17, código Anexo 1.

```
void setup()
{
    /* UARTs setup */
    Serial.begin(DEBUG_UART_BAUDRATE);

    /* Inicializa comunicacao com sensor DHT22 */
    dht.begin();

    /* Inicializa wi-fi */
    init_wifi();

    /* Inicializa MQTT e faz conexao ao broker MQTT */
    init_MQTT();
    connect_MQTT();
}

void loop()
{
    /* Verifica e garante conectividades wi-fi e MQTT */
    verify_wifi_connection();
    verify_mqtt_connection();

    /* Faz o envio da temperatura e umidade para a plataforma IoT (Tago.io) */
    send_data_iot_platform();

    /* Aguarda o tempo definido em TEMPO_ENVIO_INFORMACOES para o proximo envio */
    delay(TEMPO_ENVIO_INFORMACOES);
}
```

Figura 3.17 inicializações do processo IoT.
Fonte: Autor, 2023.

Cada uma dessas funções descreve a forma de como os dados são enviados utilizando protocolo MQTT, e as chamadas são executadas pelo broker. Todos esses procedimentos realizados na figura 3.17 nas linhas de código, foram tratados na camada de rede e aplicação da arquitetura IoT, conforme descrito no capítulo 2 deste estudo de caso, representado pela camada de aplicação na plataforma TAGO.IO, da figura 3.18.

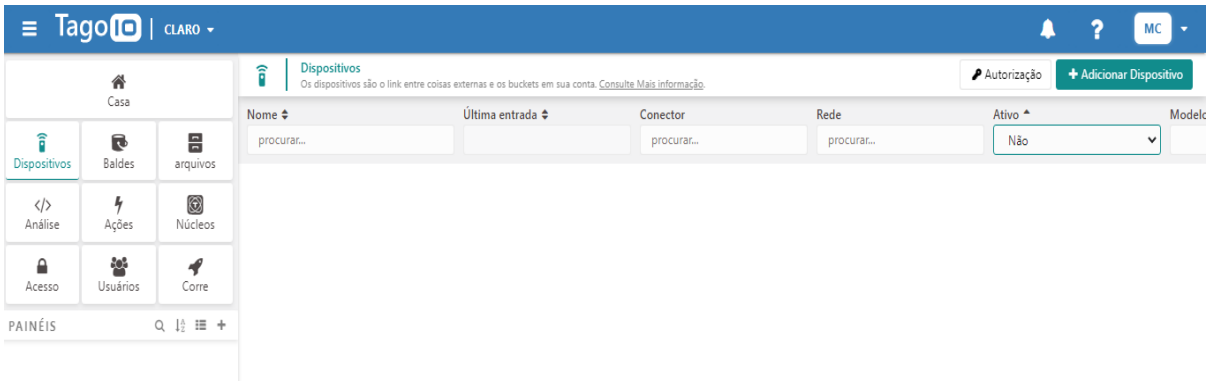

Figura 3.18 Acesso à plataforma TAGO.IO
Fonte: Autor, 2023.

Criando dispositivo device plataforma TAGO.IO.

Após logon de comunicação MQTT na plataforma TAGO.IO (Figura 3.19) que apresenta várias formas de conexão aos dispositivos baseados na camada de arquitetura IoT, mas como a nossa pesquisa está direcionada para o protocolo MQTT, então criaremos um device com essa característica, pertencentes a este protocolo conforme figura 3.19.

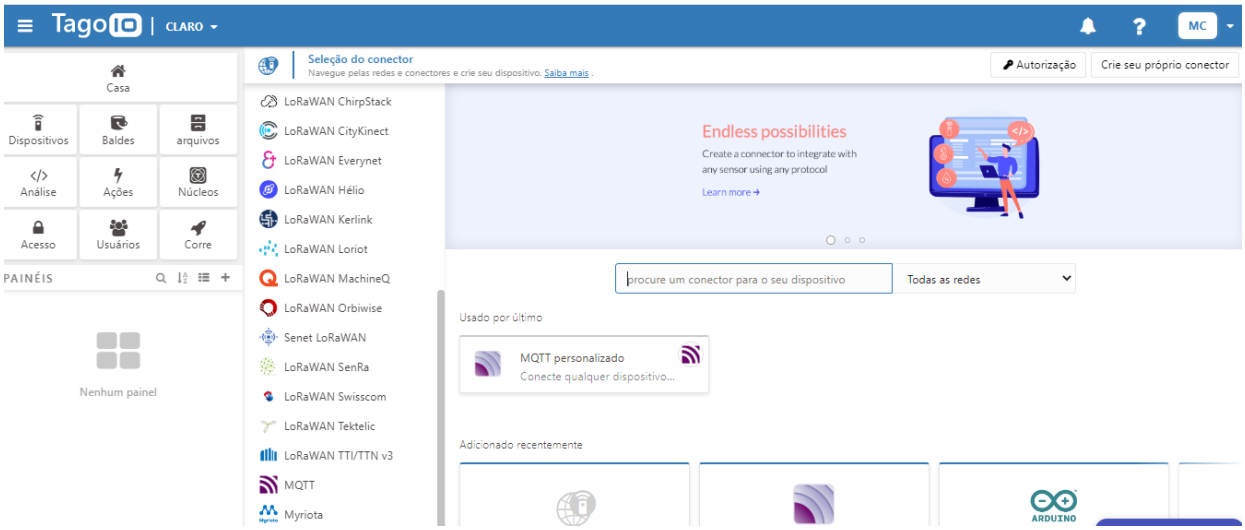

Figura3.19 Criação de Conexão dispositivo MQTT.
Fonte: Autor, 2023.

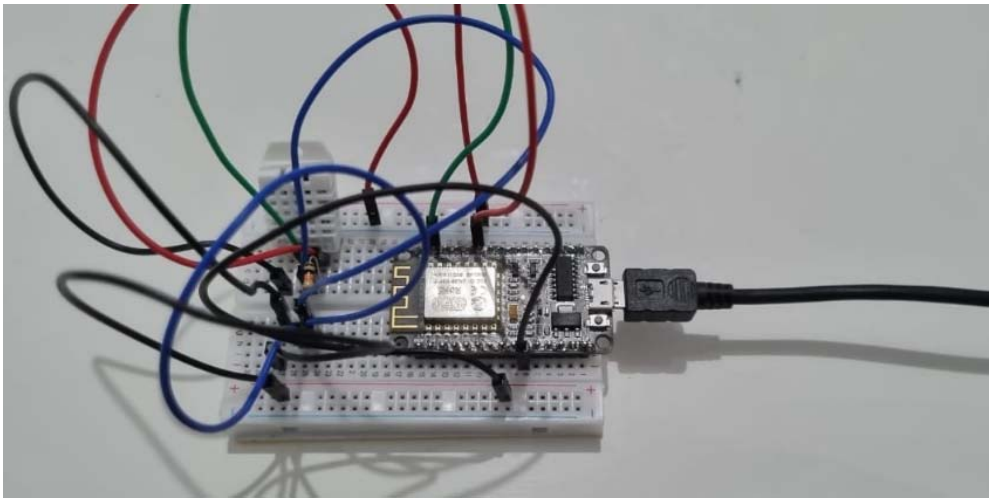

Figura 3.20, Imagem do dispositivo utilizado para este projeto Segurança IoT.
Fonte: Autor, 2023.

CAPÍTULO 4

Resultados Obtidos

4.1 - Demonstração dos Resultados

Os resultados obtidos serão demostrados abaixo, o protótipo realizou envio de informações sobre os dados coletados através do sensor DHT22 em processo de autenticação na plataforma TAGO.IO. No momento de acesso à plataforma, a ferramenta registrou 1.1k de informações durante o teste de conectividade, esses dados são referentes a temperatura e umidade do ar, conforme figura 4.1. As informações refletidas abaixo, confirmam que o dispositivo está ativo, e que o tráfego está sendo registrado na plataforma em nuvem através das evidências gráficas do dashboard, conforme figura 4.2, o qual realizou seu processo de autenticação, conforme figura 4.1, que executou a chamada do Token fornecido pela nuvem em background, seguido pela chamada do IP e porta, disparados pela execução do dispositivo IoT na residência, que iniciou as medições de temperatura e umidade do ar da residência. Após esta autenticação e início da coleta dos dados pelo dispositivo Iot, é que iniciaremos as tentativas de ataques ao meio de comunicação entre o dispositivo na residência e rede Wi-Fi local, descritos nos tipos de invasões.

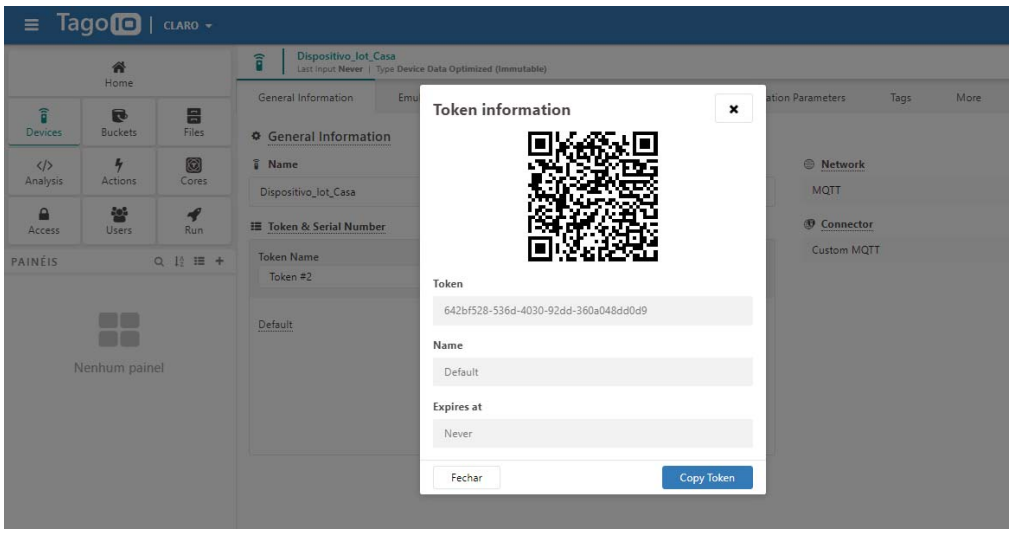

Figura 4.1 Token de autenticação do dispositivo IoT
Fonte: Autor, 2023.

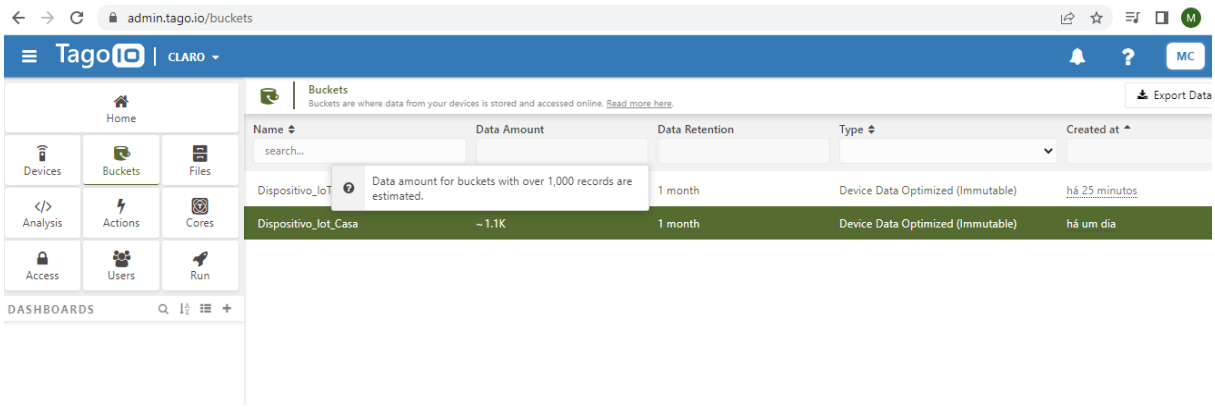

Figura 4.2 Informações sendo recebidas na plataforma TAGO.IO, após autenticação.
Fonte: Autor, 2023.

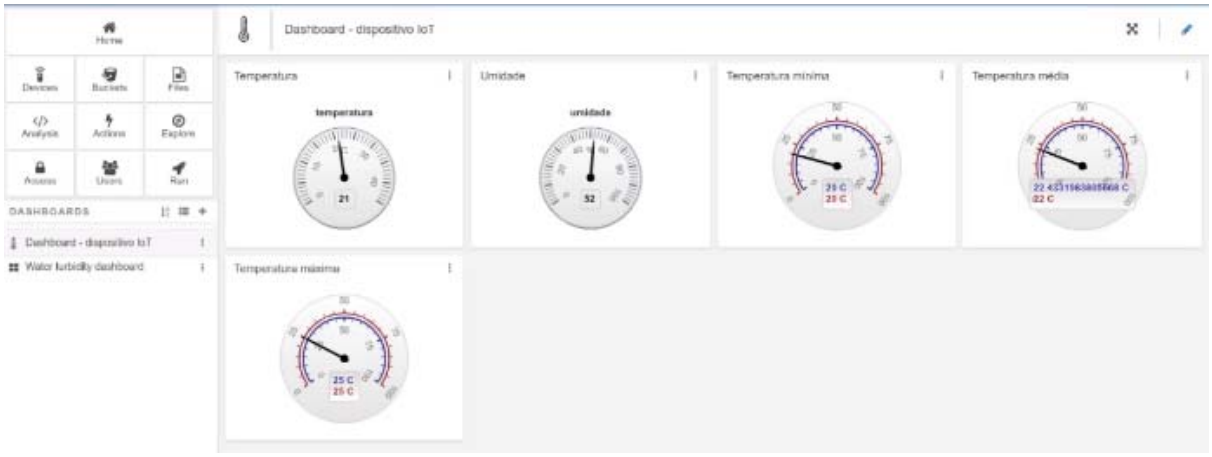

Figura 4.3 informações da temperatura e umidade do ar no dashboard.
Fonte: Autor, 2023.

4.2 - Ataque SYN Flood

Foi explorado um tipo de ataque muito utilizado em redes corporativas, como o SYN Flood, baseado em explorar a uma vulnerabilidade tradicional no processo de handshaking do protocolo TCP, comum a qualquer dispositivo que utiliza o protocolo TCP/IP para transmissão de dados pelo payload. A causa é impedir que o alvo faça conexões legítimas, e causando assim, negação de serviço. Sua estratégia principal é usar o TCP three-way handshake, como uma vulnerabilidade a ser explorada no protocolo MQTT, pois este protocolo é baseado em troca de mensagens por broker.

O three-way handshake do protocolo TCP é um processo que ocorre durante a fase de conexão entre duas máquinas, garantindo que ambas estão cientes da conexão e prontas para trocar mensagens (IETF RFC 793, 1981).

Há a troca de mensagens entre cliente e servidor:

SYN: Um pacote de sincronização é enviado do cliente para o servidor, sinalizando sua intenção de se conectar.

SYN/ACK: O servidor responde, reconhecendo a conexão.

ACK: O cliente avisa o servidor de que recebeu o pacote SYN/ACK e a conexão é estabelecida.

O SYN flood: Consiste em mandar uma série de pacotes SYN para um servidor e, ao receber os respectivos pacotes SYN/ACK, simplesmente ignorá-los. Dessa maneira, o servidor fica esperando pela resposta ACK que nunca chegará. Eventualmente o tempo limite de espera do servidor para cada conexão expira, mas, logo em seguida, o atacante inicia novas conexões, o que acaba por manter o servidor incapaz de receber conexões legítimas, caracterizando a negação de serviço do servidor.

4.2.1- PENTEST Syn Flood

O Pentest é a exemplificação em ambiente de laboratório sobre as evidências ocorridas em possíveis simulações de ataque. O primeiro Pentest de vulnerabilidade que será demostrado foi aplicado na camada de enlace, junto ao protocolo MQTT que também faz uso deste meio acesso. Realizaremos um ataque ao protocolo ICMP desta camada, que irá forçar uma queda de todos os serviços de envio e recebimento de arquivo na rede residencial, onde se encontra o dispositivo, ataque conhecido como SYN Flood.
Ataque de inundação de ICMP execução do comando:
hping3 -1 192.168.0.8 --flood --rand-source, conforme figura 4.4

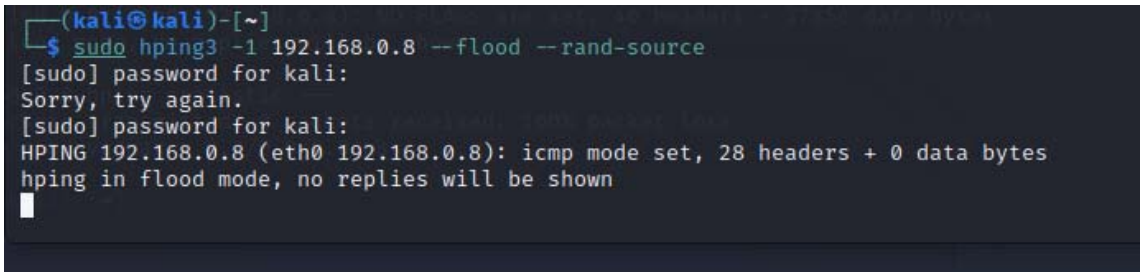

Figura 4.4 Evidência do ataque ao ICMP do IP.
Fonte: Autor, 2023.

Para visualizar aplicabilidade do ataque, monitoramos através da ferramenta *wireshark*. O ataque foi direcionado ao IP do dispositivo IoT, no teste IP atribuído ao

dispositivo 192.168.0.8, que teve o resultado apresentado na linha de comando, conforme figura 4.4 e 4.5.

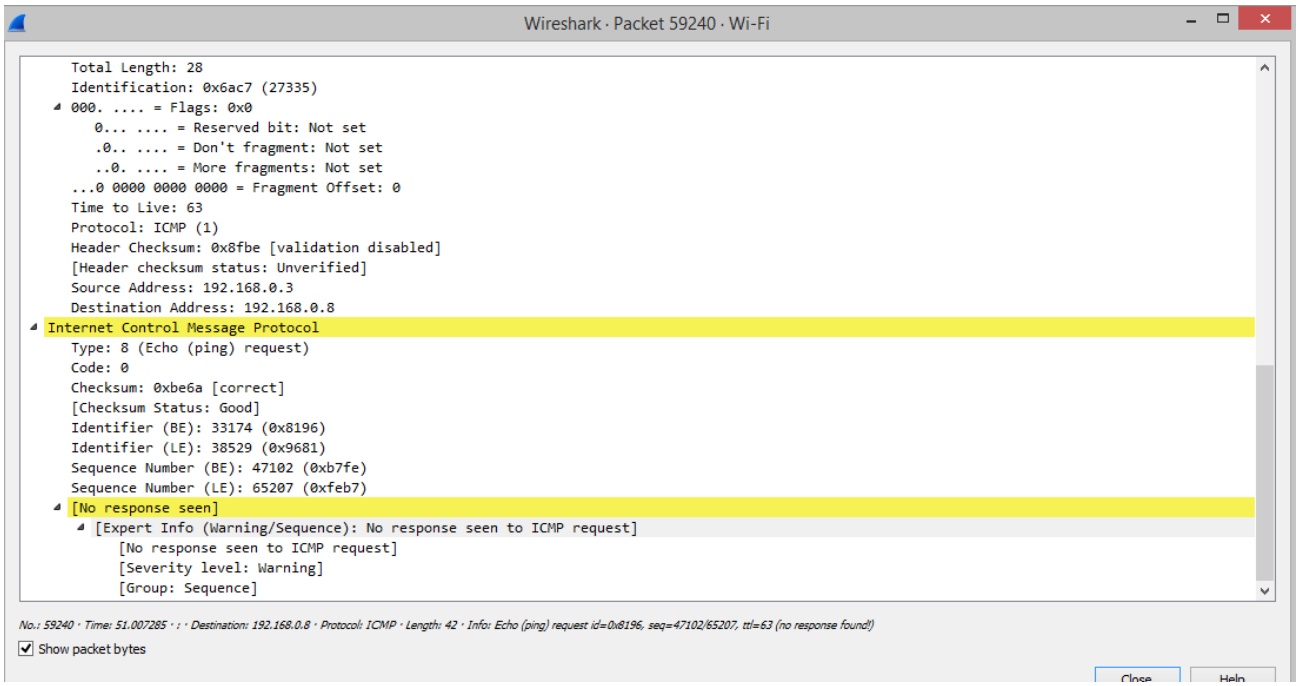

Figura 4.4 Evidência do ataque ao ICMP do dispositivo IoT por Wi-Fi.
Fonte: Autor, 2023.

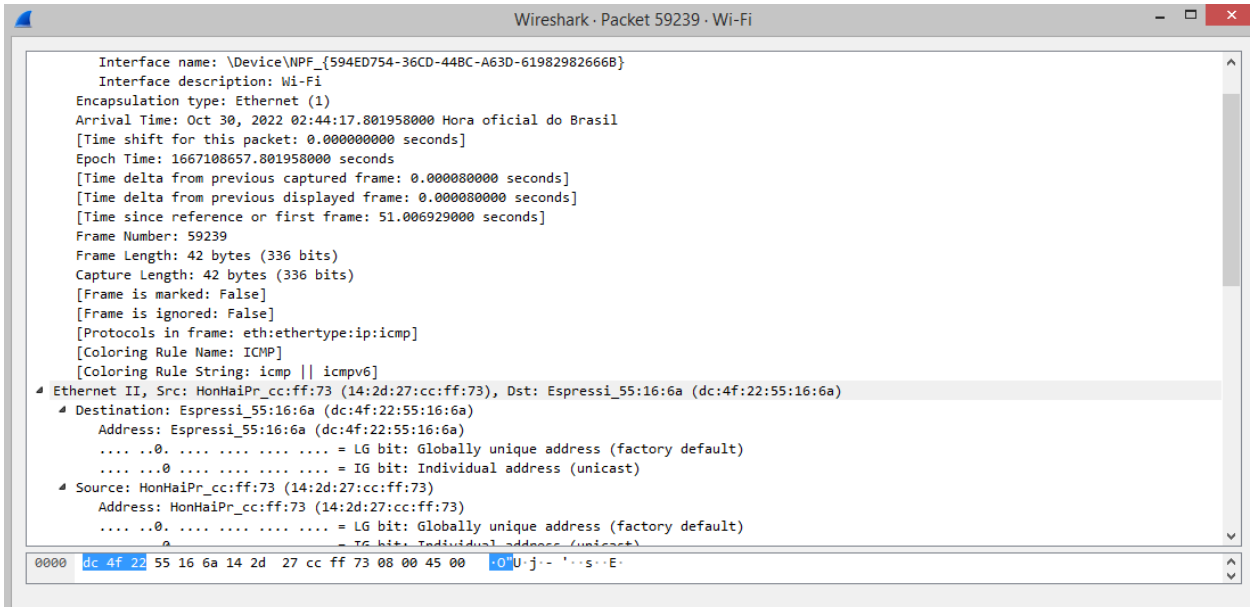

Figura 4.5 ataques ao ICMP do dispositivo IoT por Wi-Fi.
Fonte: Autor, 2023.

Analisando um dos dashboards de monitoração de temperatura da platafoma TAGOIO, observa-se que, na figura 4.6, o serviço de coletagem de dados parou de receber o que foi coletado pelo sensor DHT22 controlado pelo NodeMCU. O ataque foi

direcionado ao ip desse dispositivo através da rede Wi-Fi. O mesmo fato também ocorreu no dashboard de umidade do ar.

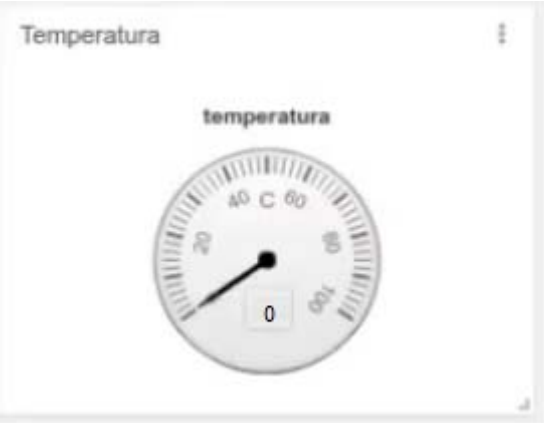

Figura 4.6 Dashboard de temperatura do Dispositivo IoT.
Fonte: Autor, 2023.

4.3 - Ataque ao MQTT Broker

O SYN flood é um ataque que explora uma vulnerabilidade do protocolo TCP, mas, como o protocolo MQTT frequentemente depende do TCP para criar conexões e trocar mensagens, toda e qualquer vulnerabilidade encontrada no protocolo TCP pode também ser utilizada para atacar sistemas baseados em MQTT. O ambiente utilizado para realizar o ataque foi:

Máquina atacante: (Kali Linux), também Linux, na mesma rede local, Local Área Network (LAN), do broker.

hping3: Ferramenta de ataque desenvolvida pela Offensive Security Organization para a criação, envio e análise de pacotes. O hping3 foi escolhido em detrimento de outras implementações do SYN flood por se tratar de uma ferramenta consolidada tanto no meio acadêmico quanto na indústria, de fácil uso e grande versatilidade. (Liang et al., 2016) [39].

4.3.1 - PENTEST Mqtt Broker

Realizado uma PENTEST de ataque ao MQTT Broker por inundação por pacotes grandes, o comando executado foi o: *hping3 -S -p 1883 --flood --rand-source 192.168.0.8*, com o objetivo de direcionar a porta do dispositivo 1883. Conforme

evidencia do comando no *kali Linux*. Figuras 4.7, 4.8, 4.9, o que implicou na parada do dispositivo IoT.

O comando executado no comandline do Kali Linux, para o PENTEST foi:

 # hping3 -c 15000 −d 120 -S -p 1883 -f lood 192.168.17.1

Os parâmetros especificados são:

- -c. Número de pacotes a serem enviados.
- -d. Tamanho, em bytes, de cada pacote.
- -S: Especifica que os pacotes contêm a flag SYN, ou seja, que são pacotes SYN.
- -p. Porta a receber o ataque. O servidor não conseguirá receber outras conexões nessa porta. A porta 1883 é a padrão para o protocolo MQTT.
- -flood. Especifica que se trata de um ataque de flooding, hping3 enviará os pacotes o mais rápido possível.
- 192.168.0.8: Endereço IP do servidor (máquina alvo).

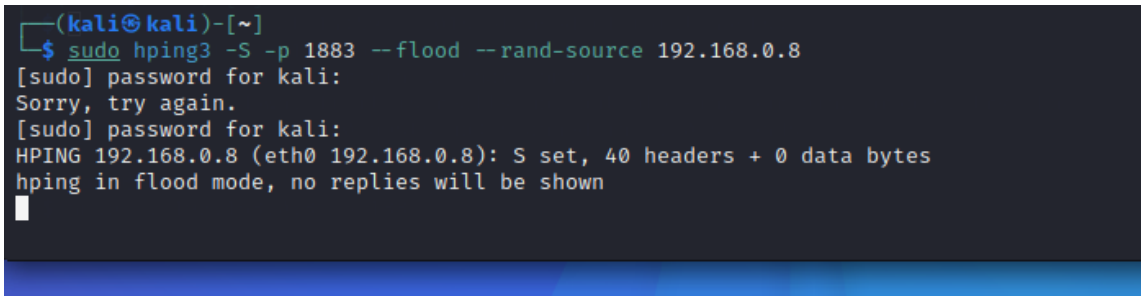

Figura 4.7, ataque MQTT broker direcionado a range do IP 192.168.0.0/24 - porta 1883
Fonte: Autor, 2023.

.

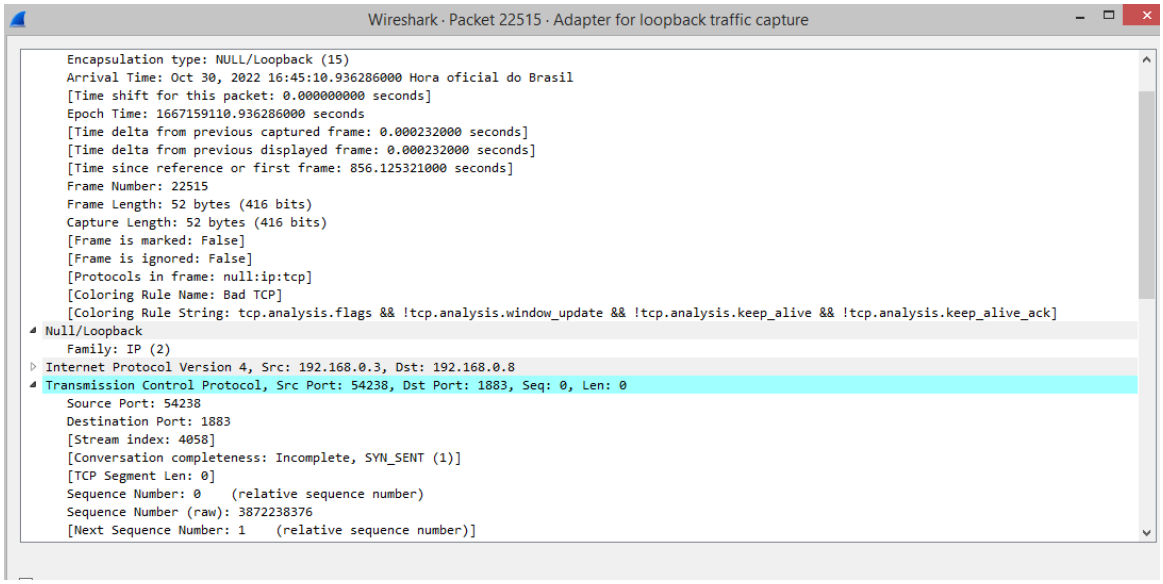

Figura 4.8 Evidência de ação no Wireshark do ataque MQTT broker.
Fonte: Autor, 2023.

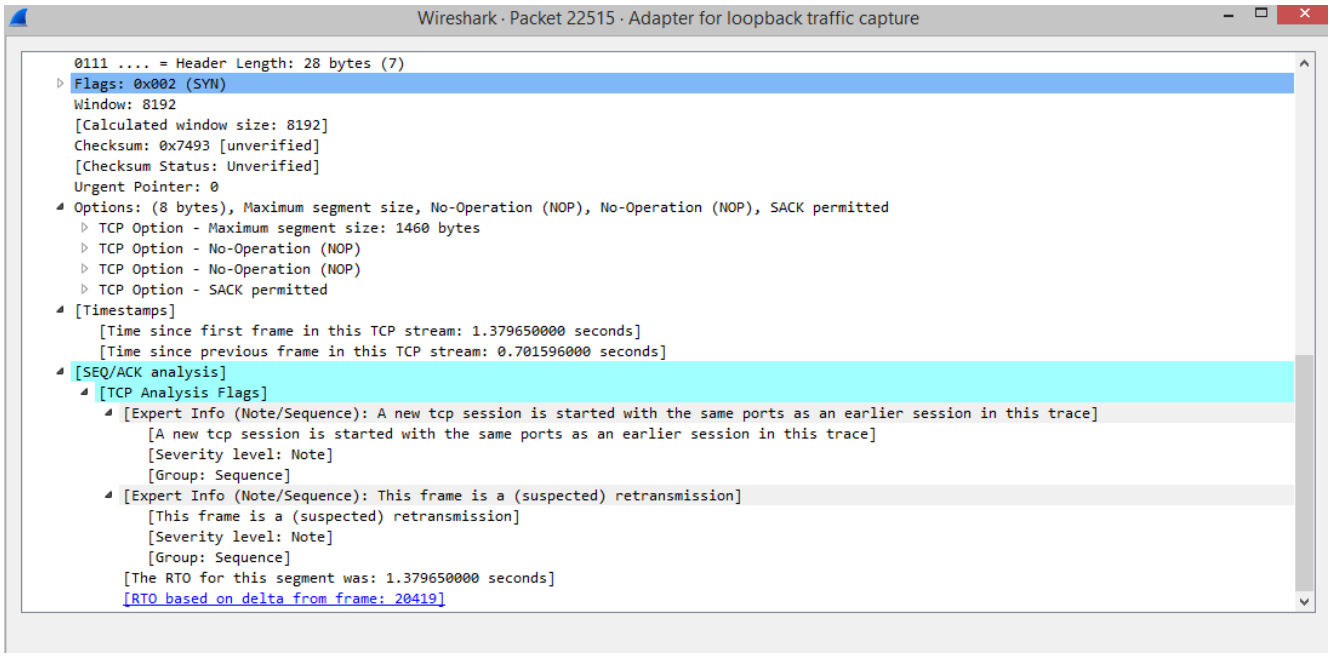

Figura 4.9 Captura no Wireshark do ataque MQTT Broker.
Fonte Autor, 2023.

O resultado deste ataque ocasionou uma pane geral em todos os processos de transmissão da rede interna da residência, ao qual foi direcionada somente a um dispositivo IoT, afetando todo o ambiente interno da residência conectado à internet, conforme figura 4.10.

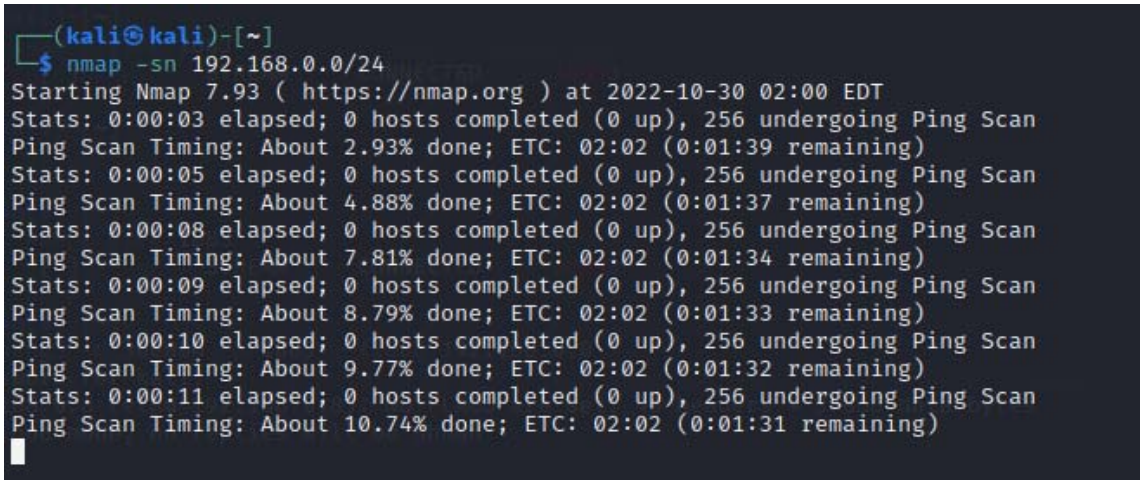

Figura 4.10 Nmap não consegue mais fazer varredura no range 192.168.0.0/24.
Fonte: Autor, 2023.

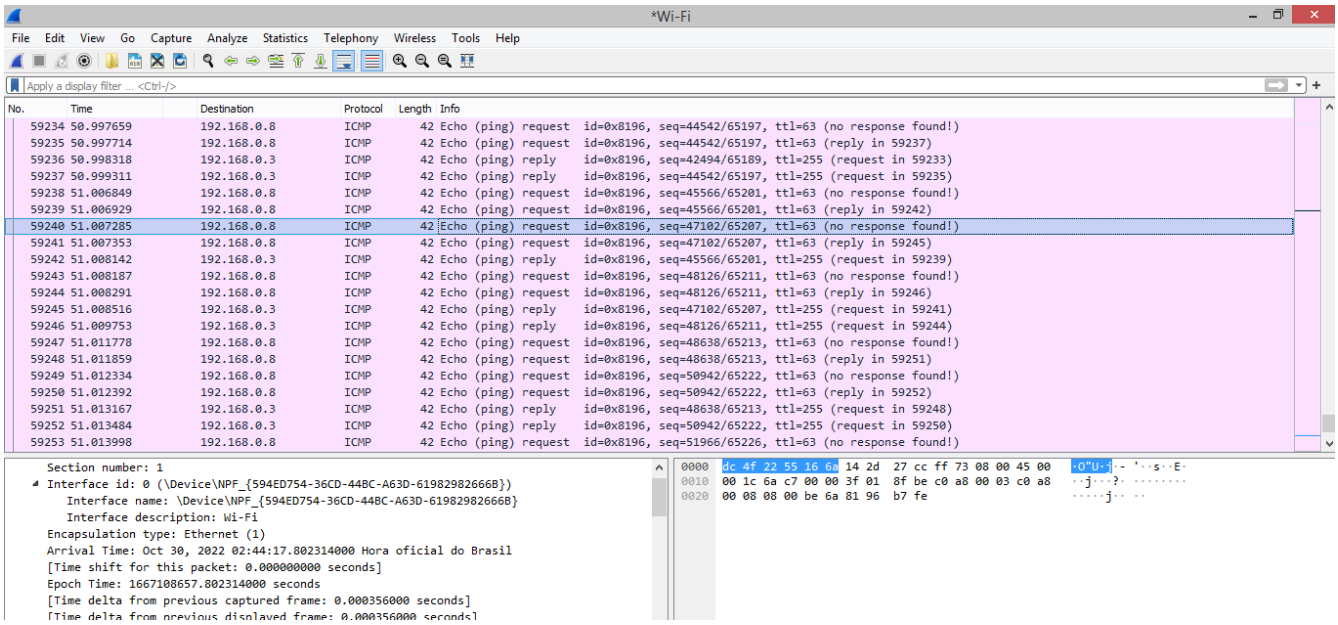

Figura 4.11 Sem acesso ao do range 192.168.0.0/24 pelo wireshark.
Fonte: Autor, 2023.

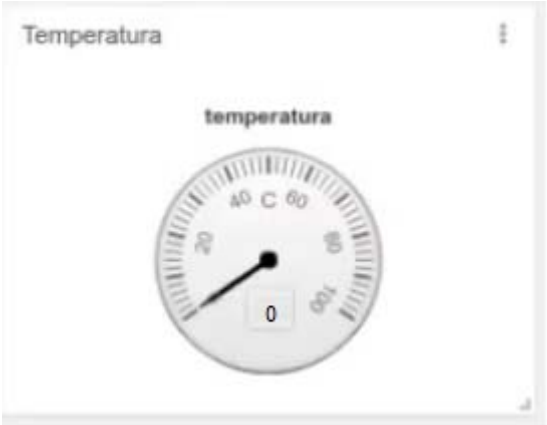

Figura 4.12 Parou de coletar dados do Dispositivo IoT.
Fonte: Autor, 2023.

Após interrupção de todos esses ataques aplicados a partir do servidor Kali, as retransmissões retornaram a sua normalidade, conforme figura 4.14 na simulação do PENTEST, os pacotes de dados começaram a ser retransmitidos enviados novamente para porta 1883, conforme figura 4.13, e dados voltaram a ser coletados pelo dashboard da plataforma em nuvem.

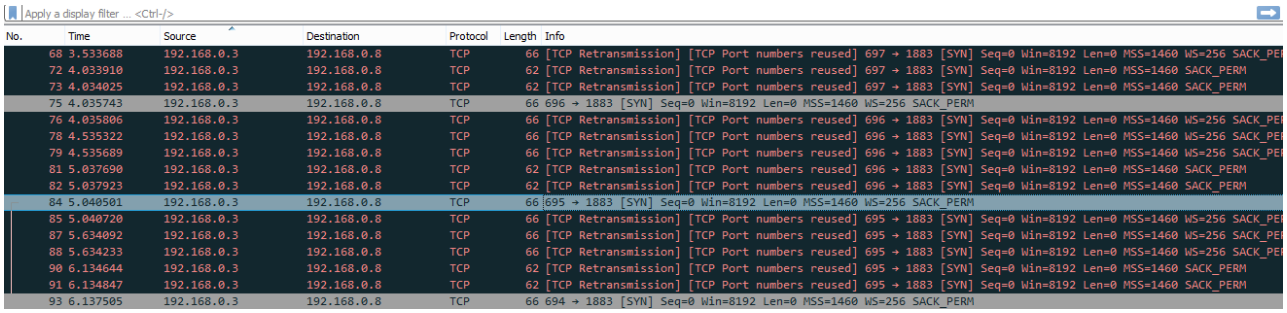

Figura 4.13 Processos de transmissão sendo retomados.
Fonte: Autor, 2023.

4.4 - Ataque Sybil

O ataque Sybil consiste fundamentalmente em simular uma abundância de identidades falsas, fazendo com que um cliente sybil, ou nó sybil no contexto de redes ponto a ponto, se passe por vários clientes. Isso pode ser atingido usando uma série de diferentes técnicas, como, por exemplo, a criação de identidades falsas ou o roubo de identidades verdadeiras, fazendo com que a autoridade, frequentemente um servidor, ou, no caso da topologia no protocolo MQTT, o broker, acredita que as identidades roubadas pelo nó sybil são os clientes legítimos (DOUCEUR, 2002) [38], conforme figura 4.14.

Esse tipo de ataque se mostra muito efetivo no contexto de redes MQTT, uma vez que na grande maioria das implementações o broker responsável não limita o número de clientes que podem operar sob um mesmo endereço IP. Para explorar a vulnerabilidade foi desenvolvido um programa na linguagem Python que utiliza a biblioteca Paho do projeto Mosquitto para criar e instanciar clientes, abrir conexões e realizar inscrições e publicações. Para demostrar esse tipo de ataque, o qual é voltando para cliente servidor precisaria de um servidor com o MQTT instalado, como o servidor que usamos está na internet, achamos prudente não aplicar um ataque ao ambiente em nuvem.

```
import paho.mqtt.client as mqtt
import random,string,sys

def gen_cli_id (size):

" " "
Generates random client ID o f 'size' characters
" " "
return ' '.join([ random.choice(string.ascii_letters + \
string.digits) fornin range(size)])

if __name__ == ' __main__ ' :
if len ( sys . argv ) != 5 :
print ('usage:python3 bad_sub.py HOST\
PORT TOPIC SYBIL_NODES ' )
exit(0)

host = sys.argv[1]
port = int ( sys.argv[2])
topic = sys.argv[3]
sybil_nodes = int(sys.argv[4])
clients = [ ]

print('Subscribing sybil nodes ...')

for i in range( sybil_nodes):

clients.append(mqtt.Client(gen_cli_id(10)))
clients[i].connect(host,port)
clients[i].subscribe(topic)
print('Publishing messages,flooding broker . . . ')
i = 0
while True:
clients[i].publish(topic, 'flooding topic')

i= (i + 1) % sybil_nodes
```

Figura 4.14, código de programação para ataque ao MQTT.
Fonte: Autor, 2023.

4.5 - Mitigando os ataques aplicados e como evitá-los

Como podemos resolver esses problemas de ataques descritos neste trabalho? Através de orientações de segurança, que podem mitigar os ataques exemplificados neste ambiente de projeto, voltado para IoT residencial, que utiliza plataforma de troca de mensagens pelo protocolo MQTT.

Para se usar dispositivo IoT em uma rede residencial é necessário seguir algumas orientações de segurança, relacionado aos tipos de ataques Sybil, Syn Flood, MQTT broker, sempre se atentando ao manual do fabricante Iot, no que tangem segurança da informação:

- Se faz necessário que o dispositivo utilize sempre conexões autenticadas e com certificado digital, tais como: MQTT com TLS/SSL, HTTPS (MQTT broker), antes de implementar Iot, verificar a compatibilidade com esses protocolos no dispositivo Iot.

- Se possível, faça criptografia do payload das mensagens entre dispositivo e plataforma IoT, verificar se o dispositivo aceita esse tipo de configuração para ataques como Syn Flood e MQTT broker.

- Sempre que possível, utilize criptografia assimétrica para evitar os ataques Sybil e MQTT broker.

- Além da conexão autenticada para acesso ao dispositivo Iot, utilize login e senha personalizados, pois o protocolo MQTT suporta essa funcionalidade, e assim poderemos mitigar ataques Sybil e MQTT broker.

- Nunca deixe o software de seu dispositivo em cartões SD / micro-SD ou em chips eMMC sem proteção, isso para dispositivos IoT, que possui esse tipo de entrada para gravações de atualização ou rotinas de backup, por ser suscetível a ataques Sybil, Syn Flood e MQTT broker.

- Deixe o mínimo possível da inteligência do seu negócio programada no seu dispositivo Iot. Assim, se ele for invadido, não será possível copiar toda a inteligência do seu dispositivo e o processo de automatização criado nele.

CAPÍTULO 5

Conclusão

Esse projeto teve sua análise sobre a construção de um protótipo IoT em arduino IDE, para medição de temperatura e umidade do ar residencial, tendo como objetivo explorar a vulnerabilidade do protocolo de mensageria MQTT, que realiza a troca de mensagens por meio deste protocolo. Tendo em vista que conexão realizada ocorreu entre a camada física e a camada de aplicação, para realizar ataques neste meio de comunicação, o qual a conexão se dá por rede Wi-Fi, utilizando o PENTEST para analisar e efetuar ataque no microcontrolador NodeMCU. O processo de autenticação ocorre no ambiente em nuvem que se dá via token gerado pela própria plataforma em nuvem onde ocorre a inserção dos dados referente as informações coletadas pelo sensor do dispositivo. Foram feitos testes de vulnerabilidade via ataques de negação de serviços realizados a partir de um servidor Kali Linux situado na mesma rede que o dispositivo IoT, o qual explorou a vulnerabilidade tanto da rede interna como do dispositivo IoT através de uma porta específica do protocolo MQTT. Após os ataques explorados podemos perceber que qualquer dispositivo IoT está sucessível a qualquer tipo de ataque, aqui foram explorados ataque simples destinados a um dispositivo IoT residencial. Como possíveis soluções de mitigação para os ataques apresentados neste trabalho podemos considerar, que se utilize sempre conexões autenticadas e com certificado digital, se possível, faça criptografia do payload das mensagens entre dispositivo e plataforma IoT, agora se considerarmos esses e outros tipos de ataque aplicados em dispositivo IoT indústrias ou em usina nuclear, se faz necessário que busque-se normas de segurança da informação para implantação desse dispositivos de forma segura , podemos destacar as normas ISO 27001 voltada para segurança da informação e a 27002 voltada para gestão de risco o qual irá informar várias formas de mitigação de segurança da informação, que poderão ser aplicadas também para dispositivos Iot de qualquer natureza, que utilizam a internet com meio de comunicação.

5.1-Trabalhos Futuros

Efetuar mais PENTEST de teste de conexão aplicável a outros protocolos como Lora, e se possível, realizar teste de conexão via vários dispositivos Arduinos como Raspberry pi, banana pi e NodeMCU, aplicando tentativa de invasão explorando a vulnerabilidade do hardware desses dispositivos, e corrigir essas vulnerabilidades mediante aplicação de fix nos dispositivos internos do microcontrolador desse Arduinos, evitando assim outros tipos de ataques não mencionados neste trabalho.

Referência Bibliográfica

[1] GALEGALE, G. P. et al. **Internet das coisas aplicada a negócios - um estudo bibliométrico**. Journal of Information Systems and Technology Management, TECSI, v. 13, n. 3, dec 2016. Citado na página 1.

[2] NESHEIM, M. B.; ROSNES, K. S. A **smarter home, the smarter choice?** Dissertação (Mestrado) - Universitetet i Stavanger, 2016. Citado na página 1.

[3] MUKHERJEE, B. et al**. Flexible IoT security middleware for end-to-end cloud–fog communication**. Future Generation Computer Systems, Elsevier BV, v. 87, p. 688–703, oct 2018. Citado na página 1.

[4]"Disponível em: <**https://tecnoblog.net/noticias/2019/06/25/hacker-invade-nasa-com-raspberry-pi/** >". Acesso em:26 jun.2023.

[5]"Disponível em: <**https://blog.avast.com/hacker-creates-seven-new-variants-of-the-mirai-botnet**>". Acesso em:27 maio.2023.

[6]"Disponível em: <**https://iot-analytics.com/iot-2021-in-review/>".** Acesso em:26 jun.2023.

[7] ATZORI, L.; IERA, A.; MORABITO, G. **The internet of things: A survey**. Computer Networks, Elsevier BV, v. 54, n.15, p. 2787 – 2805, oct 2010.TraduçãoNossa.Citado na página 4.

[8] RIBEIRO, R. M. O. S**egurançsa em iot simulação de ataque em uma rede rpl utilizando contiki**. Universidade Federal de Uberlândia, 2018. Citado 2 vezes nas páginas 4 e 6.

[9] Eduard Kovacs, ED kovacs. **70 Percent of IoT Devices Vulnerable to Cyberattacks**. Securityweek, ano 2014, "disponível em: <https://www.securityweek.com/70-iot-devices-vulnerable-cyberattacks-hp/>", acesso em: 30 maio. 2023.

[10] MAGRANI, E. **A internet das coisas.** Rio de Janeiro: FGV Editora, 2018.

[11] PACHECO, L. A. B. **Arquitetura para privacidade na integração de internet das coisas e computação em nuvem**. Dissertação (Mestrado em Informática) Departamento de Ciências da Computação, Universidade de Brasília, Brasília, 2018. "Disponível em: < https://repositorio.unb.br/handle/10482/32137>". Acesso em: 23 jun.2023, de Luís Alberto Belém Pacheco.

[12] SANTOS, B. P. et al. **Internet das coisas: da teoria à prática**. 2016. Disponível em: < https://homepages.dcc.ufmg.br/~mmvieira/cc/papers/internet-das-coisas.pdf >. Acesso em: 26 jun.2023.

[13] Luís Gustavo Machietto; Anderson Luiz Nogueira Viera; Fernando Esquírio Torres **ARQUITETURA E INFRAESTRUTURA DE IOT** et al.

[14] LEITE, J. R. E.; MARTINS, P. S.; URSINI, E. L. **Segurança e gerenciamento da rede IoT**. In: BRAZILIAN TECHNOLOGY SYMPOSIUM, 2019, Campinas. Anais eletrônicos[...]. "Disponível em: <https://lcv.fee.unicamp.br/images/BTSym-19/Papers/057.pdf >". Acesso em: 26 jun.2023.

[15] **A INTERNET das COISAS (IoT): Tecnologias e Aplicações**, São Paulo, 2017" Disponível em: <https://lcv.fee.unicamp.br/images/BTSym-17/Papers/76926.pdf>". Acesso em 27 maio.2023.

[16] RIPATHY, B. K.; ANURADHA, J. **Internet of things (IoT): technologies, applications, chal-lenges, and solutions**. Boca Raton: CRC Press, 2018

[17] DELICATO, F. C.; PIRES, P. F.; BATISTA, T. **Middleware solutions for the internet of things**. London: Springer, 201

[18] TAKABI, H.; JOSHI, J. B. D.; AHN, G.-J. **Security and privacy challenges in cloud compu-ting environments**. IEEE Security and Privacy, v. 8, no. 6, nov. /dez. 2010. "Disponível em <https://www.cs.ru.nl/~jhh/pub/secsem/takabi2012security-privacy-cloud-challenges.pdf >". Acesso em:26 jun 2023.

[19] PALARMINI, L. F. **Conheça os principais protocolos para IoT**. Ano 2020. "Disponível em:<https://www.makerhero.com/blog/conheca-os-principais-protocolos-para-iot/> ". Acesso em: 26 jun 2023.

[20] SALMAN, T. **Internet ofthings protocols and standards**. 2015. "Disponível em: <https://www.cse.wustl.edu/~jain/cse570-15/ftp/iot_prot.pdf>." Acesso em: 26 jun 2023.

[21] N. Zhang, S. Demetriou, X. Mi,W. Diao, K. Yuan, P. Zong, F. Qian, X.Wang, K. Chen,Y. Tian, C. A. Gunter, K. Zhang, P. Tague e Y.-H. Lin. "**Understanding IoT SecurityThrough the Data Crystal Ball: Where We Are Now and Where We Are Going toBe**". Em: (2017). arXiv: 1703.09809. url: http://arxiv.org/abs/1703.09809, Acesso em 20 maio 2023.

[22] M. A. Khan e K. Salah. "**IoT security: Review, blockchain solutions, and open challenges**". Em: *Future Generation Computer Systems* 82 (2018), pp. 395–411. issn:0167739X. doi: 10.1016/j.future.2017.11.022. Url: https://doi.org/10.1016/j.future.2017.11.022.(https://www.sciencedirect.com/science/article/abs/pii/S0167739X17315765?via%3Dihub). Acesso em 20 maio 2023.

[23] SOFIA, R. C. An overview on the evolution ofIoT communication approaches. Ano 2018. Disponível em: **https://www.researchgate.net/profile/Rute-Sofia/publication/327645202_An_Overview_on_the_Evolution_of_IoT_Communication_Approaches/links/5b9f829145851574f7d1c945/An-Overview-on-the-Evolution-of-IoT-Communication-Approaches.pdf**. Acesso em: 21 março. 2023.

[24] AKRAM, H.; KONSTANTAS, D.; MAHYOUB, M. A **comprehensive IoT attacks survey based on a building-blocked reference model**. International Journal of Advanced Computer Science and Applications, The Science and Information Organization, v. 9, n. 3, 2018. Citado na página 9

[25] KHAN, W. Z. et al. **Detection and mitigation of node replication attacks in wireless sensor networks: A survey**. International Journal of Distributed Sensor Networks, SAGE Publications, v. 9, n. 5, p. 149023, jan 2013. Citado na página 10.

[26] ROONEY, C.; SEEAM, A.; BELLEKENS, X. **Creation and detection of hardware trojans using non invasive o the shelf technologies.** Electronics, MDPI AG, v. 7, n. 7, p. 124, jul 2018. Citado na página 10.

[27] CHAKRABORTY, R. S.; NARASIMHAN, S.; BHUNIA, S. **Hardware trojan: threatsand emerging solutions.** In: 2009 IEEE International High Level Design Validationand Test Workshop. [S.l.]: IEEE, 2009. Citado na página 10.

[28] OSANAIYE, O.; ALFA, A.; HANCKE, G. A **statistical approach to detect jamming attacks in wireless sensor networks**. Sensors, MDPI AG, v. 18, n. 6, p. 1691, may 2018. Citado na página 10.

[29] GROVER, K.; LIM, A.; YANG, Q. **Jamming and anti-jamming techniques in wireless networks: a survey**. International Journal of Ad Hoc and Ubiquitous Computing, Inderscience Publishers, v. 17, n. 4, p. 197, 2014. Citado na página 10.

[30] SALAHDINE, F.; KAABOUCH, N. **Social engineering attacks: A survey**. Future Internet, MDPI AG, v. 11, n. 4, p. 89, apr 2019. Citado na página 10.

[31] CANEILL, M.; GILIS, J.-L. **Attacks against the Wi-Fi protocols wep and wpa**. Journal, no. December, 2010. Citado na página 12.CANEILL, M.; GILIS, J.-L. Attacks against the Wi-Fi

[32]JEBA, S.; PARAMASIVAN, B. "**False data injection attack and its countermeasures in wireless sensor networks**". European Journal of Scientic Research, v. 82, 07 2012. Citado na página 13.

[33] KUMARASAMY, S.; GOWRISHANKAR, A. **An active defense mechanism for tcp synooding attacks**. arXiv preprint arXiv:1201.2103, 2012. Citado 2 vezes nas páginas 13 e 14.

[34] ZHENG, O.; POON, J.; BEZNOSOV, K. **Application-based TCP hijacking**. In: Proceedings of the Second European Workshop on System Security - EUROSEC.
[S.l.]: ACM Press, 2009. Citado na página 13.

[35] MENEZES, A. R. de F. et al. **Internet das coisas e os principais protocolos**. Revista Expressão Científica, v. 2 n. 2, p. 43–56, 2017. Disponível em: < https://repositorio.ifs.edu.br/biblioteca/handle/123456789/779>. Acesso em: 11 maio. 2023.

[36]STACK OVERFLOW, P. **MQTT-Broker meets WebServer with Database**. [S. l.: s. n.], 2020. "Disponível em: <https://stackoverflow.com/questions/44924983/mqtt-broker-meets--webserver-with-database?rq=1>". Acesso em: 26 jun.2023.

[37]MOTA, L. da C. **Uma análise comparativa dos protocolos SNMP, Zabbix e MQTT, no contexto de aplicações de internet das coisas**. 2017. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Sergipe, São Cristovão, 2017. "Disponível em: https://ri.ufs.br/bitstream/riufs/10776/2/LEVI_COSTA_MOTA.pdf". Acesso em: 26 jun.2023.

[38]DOUCEUR, J. R. **The sybil attack**. In: DRUSCHEL, P.; KAASHOEK, F.; ROWSTRON, A. (Ed.). Peer-to-Peer Systems. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. p. 251–260. ISBN 978-3-540-45748-0. Citado na página 30.

[39]Liang, L.; Zheng, K.; Sheng, Q.; Huang, X**. A denial of service attack method for an iot system**. In:2016 8th International Conference on Information Technology in Medicine and Education (ITME). [S.l.: s.n.], 2016. p. 360–364. ISSN 2474-3828. Citado na página 28.

[40] SILVA, M. R. B.; LEDEL, L. C. **RaspBene IoT: uma plataforma de hardware e software aplicada ao monitoramento e controle de ambientes físicos**. [S. l.: s. n.], 2019 Disponível em: https://hto.ifsp.edu.br/portal/images/thumbnails/images/IFSP/Cursos/Coord_ADS/Arquivos/TCCs/2019/TCC_MarcioRicardoBenetassoDaSilva.pdf. Acesso em: 11 fev. 2023.

[41]"Disponível em: <https://www.3glteinfo.com/lora/lora-architecture/>". Acesso em:18 jun.2023.

[42]"Disponível em: < https://www.arduino.cc/en/software >". Acesso em:18 maio 2023.

[43] "Disponível em: < https://fritzing.org/ >". Acesso em:18 fev 2023.

[44]”Disponível em: https://www.makerhero.com/blog/tag/blynk/>”.Acesso em:18 março 2023.

Anexo 1

```
/*
 * Programa: código-fonte do dispositivo IoT prototipo
 * Autor: Marcelo Cavalcanti da Costa
 */
#include <stdio.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <DHT.h>

/* Definicoes gerais */
#define TEMPO_ENVIO_INFORMACOES    5000 //ms

/* Definicoes do sensor de temperatura */
#define DHTPIN  D1   /* GPIO que o pino 2 do sensor é conectado */

/* A biblioteca serve para os sensores DHT11, DHT22 e DHT21.
   No nosso caso, usaremos o DHT22, porém se você desejar utilizar
   algum dos outros disponíveis, basta descomentar a linha correspondente.
*/
//#define DHTTYPE DHT11   // DHT 11
#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321
//#define DHTTYPE DHT21   // DHT 21 (AM2301)

void connect_MQTT(void)
{
    char mqtt_id_randomico[5] = {0};

    while (!MQTT.connected())
    {
        Serial.print("* Tentando se conectar ao broker MQTT: ");
        Serial.println(broker_mqtt);

        /* gera id mqtt randomico */
        randomSeed(random(9999));
        sprintf(mqtt_id_randomico, "%ld", random(9999));

        if (MQTT.connect(mqtt_id_randomico, MQTT_USERNAME, MQTT_PASSWORD))
        {
            Serial.println("Conectado ao broker MQTT com sucesso!");
        }
        else
        {
            Serial.println("Falha na tentativa de conexao com broker MQTT.");
            Serial.println("Nova tentativa em 2s...");
            delay(2000);
        }
    }
}
```

```
/* Funcao: envia informacoes para plataforma IoT (Tago.io) via MQTT
 * Parametros: nenhum
 * Retorno: nenhum
 */

/*
JSON a ser enviado para Tago.io:

{
    "variable": "nome_da_variavel",
    "unit"    : "unidade",
    "value"   : valor
}
*/
void send_data_iot_platform(void)
{
   StaticJsonDocument<250> tago_json_temperature;
   StaticJsonDocument<250> tago_json_humidity;
   char json_string[250] = {0};
   float temperatura_lida = dht.readTemperature();
   float umidade_lida = dht.readHumidity();
   int i;


void connect_MQTT(void)
{
    char mqtt_id_randomico[5] = {0};

    while (!MQTT.connected())
    {
        Serial.print("* Tentando se conectar ao broker MQTT: ");
        Serial.println(broker_mqtt);

        /* gera id mqtt randomico */
        randomSeed(random(9999));
        sprintf(mqtt_id_randomico, "%ld", random(9999));

        if (MQTT.connect(mqtt_id_randomico, MQTT_USERNAME, MQTT_PASSWORD))
        {
            Serial.println("Conectado ao broker MQTT com sucesso!");
        }
        else
        {
            Serial.println("Falha na tentativa de conexao com broker MQTT.");
            Serial.println("Nova tentativa em 2s...");
            delay(2000);
        }
    }
}
```

```
    /* Imprime medicoes de temperatura e umidade (para debug) */
    for (i=0; i<80; i++)
        Serial.println(" ");

    Serial.println("----------");
    Serial.print("Temperatura: ");
    Serial.print(temperatura_lida);
    Serial.println("C");
    Serial.print("Umidade: ");
    Serial.print(umidade_lida);
    Serial.println("%");

    /* Envio da temperatura */
    tago_json_temperature["variable"] = "temperatura";
    tago_json_temperature["unit"] = "C";
    tago_json_temperature["value"] = temperatura_lida;
    memset(json_string, 0, sizeof(json_string));
    serializeJson(tago_json_temperature, json_string);
    MQTT.publish(MQTT_PUB_TOPIC, json_string);

    /* Envio da umidade */
    tago_json_humidity["variable"] = "umidade";
    tago_json_humidity["unit"] = "%";
    tago_json_humidity["value"] = umidade_lida;
    memset(json_string, 0, sizeof(json_string));
    serializeJson(tago_json_humidity, json_string);
    MQTT.publish(MQTT_PUB_TOPIC, json_string);
}

void setup()
{
    /* UARTs setup */
    Serial.begin(DEBUG_UART_BAUDRATE);

    /* Inicializa comunicacao com sensor DHT22 */
    dht.begin();

    /* Inicializa wi-fi */
    init_wifi();

    /* Inicializa MQTT e faz conexao ao broker MQTT */
    init_MQTT();
    connect_MQTT();
}

void loop()
{
    /* Verifica e garante conectividades wi-fi e MQTT */
    verify_wifi_connection();
    verify_mqtt_connection();

    /* Faz o envio da temperatura e umidade para a plataforma IoT (Tago.io) */
    send_data_iot_platform();

    /* Aguarda o tempo definido em TEMPO_ENVIO_INFORMACOES para o proximo envio */
    delay(TEMPO_ENVIO_INFORMACOES);
}
```