**PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL**
**ESCOLA POLITÉCNICA**
**BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

# TESTES DE INTRUSÃO EM DISPOSITIVOS IOT DOMÉSTICOS

**GUSTAVO GEYER ARRUSSUL**
**WINKLER DOS SANTOS**

Trabalho de Conclusão II apresentado como requisito parcial à obtenção do grau de Bacharel em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Ana Cristina Benso

**Porto Alegre**
**2024**

## AGRADECIMENTOS


Eduardo Geyer
Cynthia Geyer
Ana Benso
Avelino Zorzo
Tiago Ferreto
Equipe da WSS Security

Eduardo Winkler
Rafaela Favero
Daniel Dalalana
Roben Lunardi
Alexandre Agustini
Grupo CONSEG

“And so the geek inherited the earth.”
(Edward Snowden)

# TESTES DE INTRUSÃO EM DISPOSITIVOS IOT DOMÉSTICOS


## RESUMO

Com o crescente número de dispositivos de Internet das Coisas (IoT) utilizados em ambientes domésticos, onde a privacidade e segurança são imperativas, a vulnerabilidade a ataques cibernéticos tem se tornado uma preocupação crítica. Portanto, os requisitos de segurança dos dispositivos e dos serviços implantados por eles é intrinsecamente sujeito ao escrutínio. Apesar disso, conforme documentado na literatura da área, auditorias de segurança não são tão comuns no âmbito da IoT, especialmente quando se trata de avaliações do estado de segurança. Protocolos de comunicação sem fio e plataformas para IoT conhecidamente não possuíam implementações de segurança robustas por conta de limitações técnicas que são intrínsecas desses dispositivos, e apenas nos últimos anos são vistas arquiteturas mais expressivas em relação a privacidade e integridade, com a introdução de protocolos como o Matter. Contudo, muitos dos dispositivos e aplicações, amplamente disponíveis para o consumidor comum hoje em dia, como câmeras de segurança, lâmpadas e tomadas inteligentes, não utilizam tecnologias de estado da arte e, ainda, podem estar expostos a vulnerabilidades comuns. Então, este trabalho tem como objetivo utilizar teste de intrusão para avaliar sua eficácia como instrumento de análise de vulnerabilidades em dispositivos de IoT comuns, bem como avaliar os riscos que as possíveis descobertas representam para um ambiente doméstico conectado.



**Palavras-Chave:** iot, pentest.

# PENETRATION TESTING IN HOUSEHOLD IOT DEVICES

## ABSTRACT

With the growing number of Internet of Things (IoT) devices used in home environments, where privacy and security are imperative, vulnerability to cyberattacks has become a critical concern. Therefore, the security requirements of these devices and the services they deploy are intrinsically subject to scrutiny. Nevertheless, according to literature, security audits are not as common in the IoT realm, especially when it comes to evaluating the state of security. Wireless communication protocols and IoT platforms have historically lacked robust security implementations due to the technical limitations inherent to these devices. Only in recent years have we seen more expressive architectures regarding privacy and integrity, with the introduction of protocols such as Matter. However, many devices and applications widely available to the average consumer today, such as security cameras, smart lights, and smart plugs, do not employ state-of-the-art technologies and may still be exposed to common vulnerabilities. This study aims to utilize penetration testing to assess its effectiveness as a tool for vulnerability analysis in common IoT devices, as well as to evaluate the risks that potential findings pose to a connected home environment.

**Keywords:** iot, pentest.

# LISTA DE FIGURAS

Figura 6.1 – Aplicativo exibe o endereço de e-mail da vinculação anterior parcialmente ofuscado . . . . . 29
Figura 6.2 – Câmera desvinculada da conta sem confirmação . . . . . 30
Figura 6.3 – Câmera EZVIZ CS-C6N encontrada em local público . . . . . 31
Figura 6.4 – Câmera associada ao nome "HikvisionDig 31
Figura 6.5 – Endereço IP da câmera visível pela função de "LAN Live View 32
Figura 6.6 – *Prompt* de acesso ao controle remoto possui o usuário padrão "admin 34
Figura 6.7 – Aplicativo EZVIZ mostrando a versão do dispositivo como V5.3.0 build 220623. . . . . 35
Figura 6.8 – Resultado da execução com 1000 tentativas . . . . . 37
Figura 6.9 – Resultado da execução com 500 tentativas em cada máquina . . . . . 38
Figura 6.10 – Pacote criptografado do aplicativo para a câmera, capturado pelo Wireshark . . . . . 40
Figura 6.11 – Resposta em texto claro da câmera, como documentado em [37] . . 41
Figura 6.12 – Formato de requisição ofuscado . . . . . 41
Figura 6.13 – Comando `wget` para o *endpoint* faz *download* do arquivo binário .dav da última versão de *firmware* para o modelo de dispositivo especificado com sucesso . . . . . 42
Figura 6.14 – Software SADP, utilizado para gerenciamento de dispositivos . . . . . 43
Figura 6.15 – Tabela de respostas RTSP, como "401 Unauthorized 46
Figura 6.16 – Tabela GOT contida no arquivo binário de *firmware* . . . . . 46
Figura 6.17 – Slide explicando o *exploit* utilizado pelos pesquisadores . . . . . 47
Figura 6.18 – Chave de API do usuário acessível pelo arquivo `key.json` . . . . . 54
Figura 6.19 – Listagem de *endpoints* . . . . . 55
Figura 6.20 – Mensagem de erro de verificação do *hash* MD5 . . . . . 55

# LISTA DE TABELAS


6.1 Resumo descritivo da descoberta EZVIZ-01. . . . . 49
6.2 Resumo descritivo da descoberta EZVIZ-02. . . . . 50
6.3 Resumo descritivo da descoberta EZVIZ-03. . . . . 51
6.4 Resumo descritivo da descoberta EZVIZ-04. . . . . 51
6.5 Resumo descritivo da descoberta EZVIZ-05. . . . . 52
6.6 Resumo descritivo da descoberta EZVIZ-06. . . . . 53
6.7 Resumo descritivo da descoberta GM-01. . . . . 57
6.8 Resumo descritivo da descoberta GM-02. . . . . 58
6.9 Resumo descritivo da descoberta GM-03. . . . . 58

# LISTA DE SIGLAS

AP – Access Point

API – Application Programming Interface

APS – Application Support Sublayer

ASLR – Address Space Layout Randomization

BDI – Belief-Desire-Intention

BLE – Bluetooth Low Energy

CPE – Common Platform Enumeration

CPS – Cyber-Physical System

CVE – Common Vulnerabilities and Exposures

CVSS – Common Vulnerability Scoring System

CWE – Common Weakness Enumeration

DOS – Denial of Service

DDOS – Distributed Denial of Service

ECDSA – Elliptic Curve Digital Signature Algorithm

FTP – File Transfer Protocol

GCC – GNU Compiler Collection

GPS – Global Positioning System

ICMP – Internet Control Message Protocol

IOT – Internet of Things

IP – Internet Protocol

JPEG – Joint Photographics Experts Group

LAN – Local Area Network

MAC – Media Access Control

MAC – Message Authentication Code

MITM – Man-In-The-Middle

NFC – Near Field Communication

NIST – National Institute of Standards and Technology

NVD – National Vulnerability Database

OSINT – Open Source Intelligence

OTA – Over-The-Air

OWASP – Open Worldwide Application Security Project

PAN – Personal-Area Network

PIN – Personal Identification Number

POC – Proof of Concept

PSK – Pre-Shared Key

PTES – Penetration Testing Execution Standard

QR – Quick Response

RFID – Radio Frequency Identification

ROP – Return-Oriented Programming

RPA – Resolvable Private Address

RTSP – Real Time Streaming Protocol

SADP – Search Active Devices Protocol

SDK – Software Development Kit

TCP – Transmission Control Protocol

TFTP – Trivial File Transfer Protocol

TKIP – Temporal Key Integrity Protocol

TLS – Transport Layer Security

UDP – User Datagram Protocol

URL – Uniform Resource Locator

WEP – Wired Equivalent Privacy

WI-FI – Wireless Fidelity

WPA – Wi-Fi Protected Access

XML – Extensible Markup Language

XSS – Cross-Site Scripting

ZDO – Zigbee Device Objects

# SUMÁRIO


**1 INTRODUÇÃO** . . . . . **11**

**2 INTERNET DAS COISAS (IOT)** . . . . . **13**

2.1 ARQUITETURA DE DISPOSITIVOS IOT . . . . . 13

2.2 PROTOCOLOS COMUMENTE UTILIZADOS . . . . . 15

2.2.1 SEGURANÇA NOS PROTOCOLOS DE COMUNICAÇÃO . . . . . 16

2.3 TUYA . . . . . 17

2.3.1 OUTRAS PLATAFORMAS SIMILARES . . . . . 17

**3 TESTES DE INTRUSÃO** . . . . . **19**

3.1 TESTE DE INTRUSÃO NO CONTEXTO DE IOT . . . . . 21

**4 TRABALHOS RELACIONADOS** . . . . . **24**

**5 DEFINIÇÃO DO ESCOPO DE TESTES** . . . . . **26**

5.1 ESTRATÉGIA DE TESTES . . . . . 27

**6 RESUMO DOS TESTES** . . . . . **29**

6.1 CÂMERA DE SEGURANÇA EZVIZ C6N . . . . . 29

6.1.1 **SUMÁRIO TÉCNICO DAS DESCOBERTAS** . . . . . 49

6.2 GEEKMAGIC CLOCK SMALLTV-ULTRA . . . . . 53

6.2.1 SUMÁRIO TÉCNICO DAS DESCOBERTAS . . . . . 56

**7 CONCLUSÃO** . . . . . **59**

**REFERÊNCIAS** . . . . . **61**

# 1. INTRODUÇÃO

Com o avanço da "Internet das Coisas" ou *Internet of Things* (IoT), diversos dispositivos com conectividade *Bluetooth* e *Wi-Fi* estão se tornando elementos comuns no mercado de consumo em massa. Prometem maior conveniência e automação em ambientes residenciais e empresariais, variando de câmeras IP até eletrodomésticos inteligentes e fechaduras conectadas. Contudo, surge uma preocupação relevante quanto à segurança desses dispositivos, pois são, em geral, limitados em recursos de hardware e software e não possuem mecanismos apropriados para controle de segurança, mas possuem uma pilha de protocolos sujeita às vulnerabilidades e, ainda, são operados por usuários, em sua maioria leigos, sem maiores cuidados e conhecimento sobre segurança e/ou privacidade [40].

As limitações dos dispositivos de IoT, muitas vezes, estão ligadas à baixa disponibilidade de recursos computacionais, como limitações de processador e memória, os quais acabam sendo dedicados ao atendimento das funcionalidades principais do dispositivo em detrimento de outras características como a segurança, acarretando a ausência de mecanismos adequados para proteção. Isso pode expor os consumidores a diversas vulnerabilidades cibernéticas.

Além disso, esses dispositivos podem possuir configurações padrão, como credenciais de acesso remoto, que devem ser alteradas na instalação, dentre outras opções, de acordo com cada dispositivo e aplicação. Nesses casos, o usuário, ao adicionar o dispositivo ao seu ambiente, deve alterar tais padrões, a fim de não permitir a entrada de intrusos que exploram essa opção para ganhar acesso indevido aos dispositivos na Internet. Isso exige do usuário algum conhecimento e cuidado para a proteção de seu ambiente de rede. No entanto, uma boa parte dos usuários não tem conhecimento e não toma as ações necessárias na implantação dos dispositivos [42]. Em um caso recente, ocorrido em outubro de 2024 nos Estados Unidos, aspiradores-robô da marca Ecovacs foram invadidos por agentes maliciosos que tomaram controle das câmeras e alto-falantes embutidos nos dispositivos para observar os usuários e reproduzir ofensas racistas, como reportado pelo portal de notícias australiano ABC News [9]. Um ano antes, pesquisadores de segurança haviam reportado que o controle de autenticação para controle remoto do aspirador poderia ser ignorado pois era apenas implementado no aplicativo móvel da marca, e não no próprio aspirador, no entanto, a vulnerabilidade não foi propriamente corrigida pela empresa, o que provavelmente levou ao incidente posteriormente.

Assim, o objetivo deste estudo é examinar a segurança dos dispositivos de IoT disponíveis no mercado de consumo em massa por meio de testes de intrusão. Por exemplo, o quanto adicionar uma câmera de monitoração portátil com *Wi-Fi*, altamente disponível no mercado atual e com baixo custo, ou conectar uma lâmpada inteligente à rede doméstica pode expor o usuário? Esta é uma abordagem concreta para investigar as potenciais formas

pelas quais um agente malicioso pode prejudicar um usuário, essencialmente avaliando o nível de risco de uma pessoa comum através de diferentes dispositivos.

Além disso, também é objetivo deste trabalho explorar o uso de testes de intrusão como ferramenta para verificação das vulnerabilidade dos dispositivos IoT. Será importante observar as disparidades entre o modelo de rede em dispositivos IoT e as redes convencionais [31]. Com camadas de segurança distintas e frequentemente não alinhadas aos padrões da Internet tradicional, garantir a proteção contra ameaças cibernéticas torna-se uma tarefa desafiadora em relação a esses dispositivos.

Ao examinar o estado de segurança desses dispositivos, neste trabalho, é possível fornecer informações relevantes para consumidores, fabricantes e profissionais de segurança cibernética, destacando a importância de abordar as vulnerabilidades existentes e apresentando práticas de segurança que podem ser aplicadas para a proteção dos dispositivos e dos ambientes em que se inserem.

Para isso, foram avaliados quais dispositivos são popularmente utilizados em ambientes domésticos para, subsequentemente, estudar as tecnologias subjacentes desses dispositivos para identificar falhas comuns e possíveis vulnerabilidades atreladas a cada nível de um ambiente de IoT.

Assim, o conteúdo deste trabalho está organizado nos próximos capítulos da seguinte forma: o Capítulo 2 apresenta o conceito de Internet das Coisas (IoT), suas arquiteturas e tecnologias; o Capítulo 3 apresenta o conceito de teste de intrusão e sua aplicação na segurança em IoT; o Capítulo 4 apresenta trabalhos relacionados; o Capítulo 5 define o escopo de testes executado; o Capítulo 6 contém o resumo do procedimento dos testes, apresentando o processo de descoberta das vulnerabilidades; o Capítulo 7 apresenta a conclusão deste volume.

# 2. INTERNET DAS COISAS (IOT)

O termo "Internet das Coisas" ou *Internet of Things* (IoT) refere-se à interconexão de dispositivos físicos que possuem sensores, software e outras tecnologias de comunicação, permitindo que eles coletem e troquem dados com outros dispositivos e sistemas pela Internet de maneira autônoma. Esses dispositivos podem variar desde eletrodomésticos inteligentes, como geladeiras e termostatos, até sistemas complexos de monitoramento em indústrias e cidades inteligentes [38].

A IoT tem o potencial de revolucionar uma ampla gama de setores, desde a saúde e a indústria até o transporte e o ambiente doméstico, proporcionando maior eficiência, automação e conveniência. No entanto, o crescimento exponencial da IoT também traz consigo uma série de desafios, especialmente em termos de segurança cibernética. Uma das principais preocupações é o fato de que um ambiente IoT inerentemente proporciona uma vasta superfície de ataques - com um grande número de dispositivos conectados, muitos dos quais podem ter recursos limitados de segurança, sendo o potencial de vulnerabilidades abertas significativo. A relevância desse problema é amplificada pela rápida e crescente difusão dessa tecnologia em ambientes residenciais, instigando preocupações, também, quanto à privacidade.

Ataques mais sofisticados de interoperabilidade em IoT podem resultar no sequestro do poder computacional do dispositivo para estabelecer uma rede de diversos dispositivos distribuída que operam sob o controle do agente malicioso (*botnets*) para outros usos, como a criptomineração, conforme reportado pela empresa de software Avast em 2017 [24], e ataques de Negação de Serviço, como no caso do *malware* conhecido como "*Mirai*" [28].

Para lidar com esses desafios de segurança, é consenso na literatura de segurança que é crucial que os fabricantes de dispositivos IoT incorporem recursos de segurança desde o design inicial, em vez de tratá-los como uma reflexão tardia. Isso inclui a implementação de criptografia robusta, autenticação forte e atualizações de segurança regulares. Além disso, os usuários e operadores de dispositivos IoT devem estar cientes dos riscos e adotar boas práticas de segurança, como alterar senhas padrão e manter os dispositivos atualizados com as últimas correções de segurança [42].

## 2.1 Arquitetura de dispositivos IoT

Abordar o problema da segurança em IoT como um todo não é simples, pois se trata de um paradigma que pode operar sobre estruturas específicas e diferentes de outros modelos computacionais. Embora muitos dispositivos implementarem a pilha TCP/IP, há uma diversidade de padrões e protocolos de comunicação entre dispositivos [27], com

uma definição de arquitetura base proeminente sendo a de IoT *Cyber-Physical Systems* (CPS) [35] [32].

#### TCP/IP

A pilha TCP/IP é o conjunto de protocolos que fundamenta a comunicação de sistemas na Internet, sendo composta de 4 camadas que ocupam funções específicas: Aplicação, Transporte, Internet e Acesso à rede.

1. **Camada de Acesso à rede (*Network Access Layer*)**
   Camada responsável pela transmissão física dos dados entre os dispositivos conectados na rede. Envolve aspectos de hardware e protocolos de acesso ao meio físico, como Ethernet e Wi-Fi, garantindo que os dados possam ser transmitidos e recebidos através de uma conexão física ou de uma rede local.

2. **Camada de Internet/Rede (*Internet/Network Layer*)**
   Camada onde é feito o endereçamento, encaminhamento e roteamento dos pacotes pela rede até seus destinos. O protocolo central desta camada é o IP (*Internet Protocol*), que especifica a origem e o destino de cada pacote para que seja encaminhado pelo roteador.

3. **Camada de Transporte (*Transport Layer*)**
   Responsável por gerenciar a transferência de dados através da desmontagem e remontagem dos pacotes e garantir a entrega correta, incorporando o TCP (*Transmission Control Protocol*) para transmissões orientadas a conexão e UDP (*User Datagram Protocol*) em transmissões rápidas e diretas mas sem garantia de entrega.

4. **Camada de Aplicação (*Application Layer*)**
   Camada de mais alto nível que serve como interface entre o usuário e os aplicativos de rede, através de protocolos como HTTP (*Hypertext Transfer Protocol*), FTP (*File Transfer Protocol*) e DNS (*Domain Name System*)

#### *Cyber-Physical Systems*

Uma arquitetura básica de 3 camadas que incorpora o aspecto da coleta de dados pelo ambiente físico com a transmissão em tempo real pela rede através da camada de percepção, utilizada em dispositivos de IoT mais simples.

1. **Camada de Percepção (*Perception Layer*)**
   A camada de Percepção é a que fundamentalmente diferencia a IoT de outros paradigmas. Ela é composta por sensores que são responsáveis por coletar dados do mundo físico, seja através de *Radio Frequency Identification* (RFID), *Global Positioning System* (GPS), câmeras ou demais sensores e dispositivos de *hardware*, e transmiti-los para outras camadas a fim de desempenhar as funções do dispositivo.

2. **Camada de Rede (*Network Layer*)**
   A camada de rede é responsável, primariamente, por transmitir os dados captados pelos sensores para a aplicação - porém, também tem a responsabilidade de realizar a interconexão entre dispositivos e redes. Portanto, requisitos de privacidade e segurança também são implementados por essa camada e, consequentemente, é uma camada com alta sensibilidade a ataques [15] [44].

3. **Camada de Aplicação (*Application Layer*)** A camada de aplicação compreende as aplicações que utilizam tecnologia IoT ou que são implantadas pela IoT. As aplicações implementam os sistemas inteligentes de alto nível como *Smart Homes* e *Smart Cities*, realizando o processamento dos dados recebidos, e podem ter ou não interação direta com o usuário.

## 2.2 Protocolos Comumente Utilizados

Como supracitado, há uma diversidade de arquiteturas e protocolos associados e implementados pela IoT. Para este trabalho, o foco é voltado ao estudo da estrutura tecnológica dos dispositivos IoT a fim de definir um escopo de testes que represente a maioria das aplicações da IoT em ambientes domésticos.

No caso de dispositivos domésticos, os protocolos de comunicação predominantes incluem Wi-Fi, Bluetooth, Zigbee e Matter [19] [20].

- Wi-Fi (*Wireless Fidelity*)
  O Wi-Fi é amplamente utilizado em redes domésticas, empresariais e públicas para conectar dispositivos como computadores, smartphones, tablets e uma ampla gama de dispositivos IoT, incluindo câmeras de segurança, termostatos inteligentes e eletrodomésticos conectados, através de ondas de rádio. Ele é baseado no padrão IEEE 802.11, que define redes locais sem fio (WLANs) e, hoje em dia, é utilizado principalmente nas faixas de frequência 2,4 GHz e 5 GHz, no entanto, muitos dispositivos de IoT suportam apenas redes de 2,4 GHz pelo maior alcance e maior capacidade de penetração por paredes [34].

- Bluetooth
  O Bluetooth já é amplamente utilizado em dispositivos pessoais como fones de ouvido, teclados, mouses, e também em dispositivos IoT como sensores de saúde e fitness, lâmpadas inteligentes, e sistemas de segurança residencial. Ele é construído sobre o padrão IEEE 802.15.1 que define uma PAN (*Personal-Area Network*) sem fio operável em áreas do tamanho de uma sala grande ou um *hall* [10], existindo também a variação *Bluetooth Low Energy (BLE)*, projetada para aplicações com baixo consumo de energia [36], amplamente empregada em dispositivos IoT [45].

- Zigbee
  Zigbee é um protocolo para comunicação sem fio baseado no padrão IEEE 802.15.4 e projetado para redes de baixo consumo de energia e custo [11]. É amplamente utilizado em sistemas de automação residencial na IoT [45], com sua arquitetura de rede do estabelecendo uma PAN que pode possuir entre 3 tipos de dispositivos: Coordenador, responsável por iniciar e gerenciar a rede; Roteadores, que encaminham os dados entre os dispositivos; e dispositivos finais, dispositivos simples que realizam tarefas específicas e se comunicam apenas com o coordenador ou roteadores, como sensores de movimento, temperatura e tomadas inteligentes.

- Matter
  O Matter, originalmente *Project Connected Home over IP (CHIP)*, é um protocolo de comunicação desenvolvido para melhorar a interoperabilidade entre dispositivos IoT de diferentes fabricantes, abrangendo aplicações como iluminação inteligente, termostatos, fechaduras de portas e assistentes virtuais. Ele foi projetado em 2019 pela Connectivity Standards Alliance (CSA), anteriormente conhecida como Zigbee Alliance, e é baseado em IPv6 [12]. Matter opera sobre Wi-Fi e Thread, um protocolo de rede projetado para dispositivos IoT de baixo consumo de energia e altamente escalável.

### 2.2.1 Segurança nos protocolos de comunicação

Os protocolos de comunicação precisam implementar mecanismos de segurança nativamente para proteger a troca de informações confidenciais entre dispositivos e redes além das medidas empregadas pelos protocolos a nível de aplicação. Assim, a utilização da criptografia é fundamental na comunicação em baixo nível.

- Wi-Fi
  A segurança do Wi-Fi é efetuada principalmente por protocolos de criptografia e autenticação, como o WPA2 (*Wi-Fi Protected Access 2*), que requer o uso de senha para se conectar à rede e utiliza cifras AES-CCMP para proteger os dados em trânsito. Apesar disso, não são incomuns redes domésticas e públicas que utilizam protocolos de segurança fracos como WEP (*Wired Equivalent Privacy*) e WPA (*Wi-Fi Protected Access*), que podem ser facilmente quebrados por atacantes [46].

- Bluetooth
  A segurança do Bluetooth advém principalmente da técnica de emparelhamento, em que os dispositivos devem estabelecer uma conexão direta entre eles utilizando códigos de verificação ou números de identificação pessoal (*PIN*). Para TLS (*Transport Layer Security*) na camada de rede, é utilizada a cifra AES-128-CCM e *Resolvable Private Addresses*, que são um tipo de endereço temporário e alterável gerado a partir

de um identificador público e uma chave de resolução. O RPA permite que dispositivos autorizados possam reconhecer e conectarem-se ao aparelho Bluetooth, e protege a privacidade dos dispositivos através da sua mudança periódica.

- Zigbee
  O Zigbee baseia a segurança na definição do protocolo 802.15.4, utilizando AES-128 para criptografar as informações com o modo de operação CTR e também validar os dados trafegados através do MAC (*Message Authentication Code*). Além disso, implementa camadas de segurança sobre as camadas de rede e de aplicação com 3 tipos de chaves de segurança - Chave de Rede, Chave de *Link* e Chave Mestra.

- Matter
  Para a segurança, o protocolo utiliza AES-128-CCM e emparelhamento através de NFC (*Near Field Communication*) e códigos *QR (Quick Response)* para a configuração inicial (*Onboarding*). Além disso, utiliza certificados baseados em ECDSA (*Elliptic Curve Digital Signature Algorithm*) para a verificação de identidade de dispositivos e TLS sobre a comunicação IP [12].

## 2.3 Tuya

Tuya, diferentemente dos protocolos de comunicação, é uma plataforma global mantida pela Tuya Inc. que oferece soluções integradas de Internet das Coisas, facilitando a transformação de dispositivos comuns em dispositivos inteligentes. A plataforma fornece uma ampla gama de serviços integrados, incluindo hardware, software e suporte à nuvem, permitindo que fabricantes de dispositivos criem produtos IoT rapidamente e com eficiência [17].

Dessa forma, a Tuya promete interoperabilidade através desses componentes; módulos e *chipsets* que incorporam a conectividade Tuya com suporte a múltiplos protocolos, incluindo Wi-Fi, Bluetooth e Zigbee; plataforma de desenvolvimento e configuração de dispositivos IoT e Tuya Smart App, um aplicativo móvel que permite o controle e monitorização de dispositivos inteligentes para o usuário final; e serviços de nuvem pela Tuya Cloud como APIs (*Application Programming Interface*) e ferramentas para análise de *Big Data* e inteligência artificial [18].

### 2.3.1 Outras plataformas similares

Existem algumas plataformas para IoT com a mesma proposta do Tuya e desenvolvidas junto de dispositivos *first-party*, mas que ainda podem integrar dispositivos de tercei-

ros e possuem suporte a diversos protocolos de comunicação. Esses incluem SmartThings (Samsung) [1], Google Nest [2], Alexa Smart Home (Amazon) [3], Apple HomeKit [4] e Xiaomi Mi Home [5].

# 3. TESTES DE INTRUSÃO

O teste de intrusão, conhecido como *penetration testing* ou simplesmente *pentesting*, é uma avaliação de segurança da informação que consiste em realizar ataques simulados em um sistema, aplicação ou rede para verificar sua postura de segurança cibernética [43]. O objetivo de um teste de intrusão é, primordialmente, validar a segurança de um sistema, por meio da identificação e exploração de quaisquer vulnerabilidades. Esse processo envolve o uso de ferramentas e técnicas avançadas para identificar pontos fracos e avaliar o potencial impacto de um ataque.

A estrutura de um teste de intrusão, de acordo com diversas metodologias formais como *Penetration Testing Execution Standard (PTES)* [16], *Open Source Security Testing Methodology Manual (OSSTMM)* [13] e *Open WorldWide Application Security Project (OWASP)* [14], consiste nas seguintes fases:

1. **Coleta de Informação**

   Após a conclusão de quaisquer interações de pré-engajamento, é realizado o reconhecimento do alvo. Isso pode incluir informações sobre a infraestrutura de rede, tecnologias utilizadas, procedimentos de segurança e demais conhecimento sobre o alvo que pode ser aplicado para facilitar o ataque. A coleta de informações pode ser feita através de técnicas de Open Source Intelligence (OSINT), ferramentas de escaneamento de rede e varreduras manuais ou automatizadas no sistema.

2. **Modelagem de Ameaça**

   A partir das descobertas provenientes da fase de reconhecimento, são analisadas as possíveis ameaças do sistema, tanto externas quanto internas, com base nos requerimentos de segurança, tais como a identificação de potenciais agentes de ameaça e motivações para um ataque. O principal objetivo dessa fase é apontar as principais ameaças ao sistema para que se possa priorizar, então, os recursos a serem protegidos e as estratégias de defesa a serem implementadas.

3. **Análise de Vulnerabilidade**

   Esta fase tem o propósito de identificar e avaliar, de maneira técnica, as vulnerabilidades presentes no sistema. Isso pode ser feito através de técnicas como varredura de portas, identificação de serviços e versões, análise de configurações inadequadas, revisão de código-fonte, entre outras. As vulnerabilidades encontradas são categorizadas e classificadas de acordo com sua gravidade e probabilidade de exploração pelo agente de ameaça.

4. **Exploração**

   Nesta fase, as vulnerabilidades identificadas na fase anterior são exploradas para determinar se elas podem ser utilizadas para comprometer a segurança do sistema. Isso pode envolver a utilização de ferramentas automatizadas ou técnicas manuais para tentar explorar as vulnerabilidades e comprometer ou obter acesso não autorizado ao sistema. O objetivo é simular o comportamento de um atacante real e determinar o impacto potencial de uma exploração bem-sucedida.

5. **Pós-Exploração**

   Uma vez que uma vulnerabilidade é explorada com sucesso, esta fase envolve a avaliação do acesso obtido e a realização de atividades adicionais de exploração para determinar a extensão do comprometimento do sistema. Isso pode incluir a escalada de privilégios, o acesso a informações sensíveis, a instalação de *backdoors* ou a persistência no sistema. O objetivo é entender completamente as consequências de um ataque bem-sucedido e identificar as medidas necessárias para mitigar o impacto.

6. **Relatório**

   Finalmente, o resultado de todo o processo de teste de intrusão é documentado em um relatório detalhado. O relatório deve incluir uma descrição completa das atividades realizadas, as vulnerabilidades identificadas, os métodos de exploração utilizados, os resultados obtidos e as recomendações para mitigar os riscos de segurança identificados. O relatório é geralmente apresentado aos responsáveis pela segurança da informação ou à administração da organização para que possam tomar medidas corretivas apropriadas. Ao relatório também cabem classificações, informações adicionais e referências relevantes para dar apoio e embasamento aos testes e descobertas enunciadas. As informações de classificação e criticidade geralmente são sustentadas por métricas conhecidas no cenário de avaliação e testes de segurança, como CVE (Common Vulnerabilities and Exposures), CVSS (Common Vulnerability Scoring System) e CWE (Common Weakness Enumeration).

Essa estratégia permite um melhor entendimento dos riscos associados ao usuário e à organização ao diretamente expor os vetores de ataques do sistema, estabelecendo uma perspectiva clara das potenciais situações de incidentes de segurança. O emprego de testes de segurança, especialmente em conjunto do planejamento da segurança da informação, solidifica a segurança ofensiva como um processo fundamental para a mitigação efetiva dos riscos. Deve ser ressaltado, também, que o teste de intrusão é caracterizado

pela sua agressividade - deve haver a tentativa de exploração das vulnerabilidades encontradas no decorrer do teste, diferente de um teste de análise de vulnerabilidade em que a criticidade é apenas estimada a partir de sistemas de escore padronizados, como o CVSS.

## 3.1 Teste de intrusão no contexto de IoT

Considerando o escopo de Internet das Coisas, embora os métodos tradicionais de teste de intrusão possam ser adaptados para os dispositivos, é importante ressaltar a distinção entre a avaliação dos serviços implantados pela IoT e o teste nos próprios dispositivos.

A avaliação de segurança dos dispositivos IoT envolve a análise de seus componentes físicos e software embarcado, como mecanismos de proteção contra adulteração, *firmware* potencialmente desatualizado e testes de resiliência contra ataques de força bruta e negação de serviço. A segurança dos dispositivos é essencial porque qualquer comprometimento a esse nível pode levar ao controle não autorizado do dispositivo, permitindo que atacantes executem comandos maliciosos, acessem dados sensíveis ou usem o dispositivo como ponto de entrada para atacar outros componentes da rede. Considerando a definição basilar da IoT como a arquitetura de 3 camadas, podemos delimitar alguns ataques comuns de cada camada:

1. **Camada de Percepção**

   - *Eavesdropping*

     O agente malicioso intercepta, em tempo real, comunicações privadas, como chamadas de celular, chamadas de vídeo ou mensagens de texto, através de transmissões inseguras.

   - *Node Capture*

     O agente malicioso captura o controle de um nó importante, como o nó de *gateway*. Este ataque permite acessar informações cruciais das comunicações do dispositivo como a chave usada para estabelecer as conexões seguras do dispositivo ou informações armazenadas nele.

   - *Fake Node*

     O ataque de Nó Falso consiste no atacante inserir um nó na rede, na qual o dispositivo IoT está conectado, se passando por um nó legítimo. O objetivo é enganar o sistema central para que o invasor colete informações ou controle a rede.

   - *Replay Attack*

O agente malicioso intercepta e captura a comunicação legítima entre dispositivos IoT e, em outro momento, consegue retransmitir a mesma comunicação para enganar os dispositivos e conseguir algum acesso. Notoriamente, existem dispositivos físicos como o ***Flipper Zero*** que são capazes de capturar comunicações de RFID e retransmiti-las para o alvo, providenciando acesso para o usuário não autorizado.

- *Timing Attack*

  Este ataque geralmente é direcionado a dispositivos com menor capacidade de computação e se caracteriza pela inferência de informações relevantes do sistema a partir da análise dos tempos de resposta do dispositivo a diferentes solicitações ou operações.

2. **Camada de Rede**

   - Negação de Serviço (*DoS*)

     Ataque em que os recursos da rede são exaustados ao ponto de ficarem indisponíveis para usuários legítimos. Geralmente consiste em inundar a rede com pacotes redundantes e perpetuado por um, ou no caso de Negação de Serviço Distribuída (*DDoS*), diversos agentes de ameaça.

   - *Man-In-The-Middle (MITM)*

     O agente malicioso intercepta a comunicação entre duas entidades na rede, conseguindo ler e/ou alterar os dados sendo trafegados.

   - Ataques ao *Gateway*

     Ataques direcionados ao *gateway* da rede a fim de obter acessos não-autorizados ou simplesmente comprometer a comunicação entre dispositivos.

   - *Storage Attack*

     Ataque em que o agente de ameaça compromete as informações armazenadas no dispositivo através da leitura e/ou modificação dos dados. Essas informações podem conter dados sensíveis do usuário ou do próprio dispositivo [44] [39].

3. **Camada de Aplicação**

   - *Cross-Site Scripting (XSS)*

     Um agente malicioso consegue inserir um *script* próprio *client-side* em uma página web legítima, alterando o conteúdo da aplicação e podendo levar a Execução de Código Remota (*RCE*), em que o agente consegue executar código malicioso no *host* da aplicação.

   - *SQL Injection*

Ataque que consiste em introduzir código SQL em algum *endpoint* da aplicação que não propriamente sanitiza entradas de usuário, podendo levar à exposição ou manipulação dos dados armazenados no banco de dados.

- *Broken Access Control*

  Qualquer tipo de falha no controle de acesso da aplicação que possibilita um usuário obter acessos indevidos. A exploração pode acarretar desde a exfiltração de dados até a escalada de privilégio, e geralmente está ligada a vulnerabilidades de *IDOR (Insecure Direct Object Reference*), em que um objeto é diretamente referenciado pela aplicação sem as verificações de autenticação apropriadas, e técnicas de *bypass*, em que o agente malicioso essencialmente burla as medidas de controle de acesso.

Além dos dispositivos em si, é crucial entender e avaliar a segurança dos serviços que eles implementam e fornecem. Esses serviços podem implementar funcionalidades de controle remoto, armazenamento e transmissão de dados, atualizações de firmware *over-the-air (OTA)*, e integração com plataformas de nuvem e assistentes virtuais. A segurança desses serviços depende de vários fatores, como a segurança de APIs, a proteção de dados em trânsito e em repouso, e a autenticidade e integridade das atualizações de software. A avaliação de segurança desses serviços envolve a análise de possíveis vulnerabilidades nas interfaces de comunicação, a verificação da gestão de chaves de criptografia e a implementação de mecanismos de autenticação e autorização adequados. Neste âmbito, entende-se que as estratégias para testes de segurança são mais próximas das tradicionais, como em testes de infraestrutura de rede, por exemplo.

Por fim, no que se refere aos dispositivos na esfera da Internet das Coisas, entende-se que novos desafios são introduzidos no ambiente de redes, devido à sua natureza distribuída, a diversidade de protocolos de comunicação e, muitas vezes, a falta de atualizações de segurança regulares. Portanto, isso também os torna alvos potenciais para ataques cibernéticos. Contudo, há uma falta de abordagens para avaliar a segurança geral de ambientes IoT [32] [25]. Logo, a proposta deste trabalho também consiste em explorar a aplicação de avaliações de segurança, por meio de testes de intrusão, para identificar como gerar informações e orientações sobre as melhores práticas para a proteção dos dispositivos e do ambiente.

# 4. TRABALHOS RELACIONADOS

Algumas pesquisas foram feitas em torno da segurança em Internet das Coisas e testes de intrusão como uma abordagem às demandas de verificação da segurança em dispositivos IoT:

- Dalalana e Zorzo [25] abordam questões em aberto e direções futuras para pesquisas em testes de intrusão a partir de um estudo de mapeamento sistemático baseado em diversas pesquisas realizadas sobre pentesting.
- Lunardi et al. [41] propõe uma estratégia para melhorar a segurança em redes IoT utilizando *blockchains* e destaca a importância dos *gateways* na gerência de controle de acesso a informações dos dispositivos e controle da comunicação na rede.
- Leite et al. [29] apresenta um sumário das principais vulnerabilidades encontradas em dispositivos IoT baseado no OWASP IoT Project, a partir de rotinas de teste pré-definidas.
- Chu et al. [32] propõe a condução e automação de testes de intrusão com base no modelo *belief-desire-intention (BDI)* para IoT, uma arquitetura lógica que descreve a convicção, objetivo e intenção do agente malicioso como fundamento do seu curso de ação para atacar um dispositivo.
- IoT-PEN [33] é uma *framework End-to-End* para testes de intrusão em IoT com base em uma arquitetura cliente-servidor, com os nós IoT sendo clientes de um sistema central que age como o servidor. Dessa forma, pode ser feita a análise do grafo do alvo para identificar nós e caminhos críticos da rede IoT e, subsequentemente, avaliar a criticidade das vulnerabilidades presentes baseado na *Common Platform Enumeration (CPE)*.
- Burhan et al. [27] apresenta uma pesquisa sobre os elementos, arquiteturas e problemas de segurança em IoT, relevando pilhas de 3, 4 e 5 camadas em arquiteturas IoT.
- Ali et al. [40] e Hameed et al. [35] enunciam os diversos problemas de segurança em IoT, apresentando uma descrição compreensiva dos tipos de ataques e seus respectivos vetores.
- Keersmaeker et al. [30] apresenta um estudo em *datasets* públicos de IoT, extensivamente documentando topologias, arquiteturas, modelos e tecnologias utilizadas em IoT.

Esses trabalhos representam o principal embasamento teórico para as motivações e estratégias enunciadas para este trabalho, que tem o foco de avaliar o estado de segurança de dispositivos de IoT comuns ao nível do amplo mercado através de testes de intrusão especialmente direcionados à arquitetura única da IoT.

# 5. DEFINIÇÃO DO ESCOPO DE TESTES

Dado o entendimento das questões supracitadas nos capítulos anteriores, foi definido um escopo de testes que busca representar os ambientes domésticos conectados em relação aos tipos de dispositivos e tecnologias empregados, e que também possa ser estudado e cumprido no decorrer do semestre. Assim, para a realização do trabalho foram adquiridos e testados os seguintes dispositivos:

- **Câmera Residencial Inteligente EZVIZ CS-C6N**
  Câmera de segurança residencial da marca EZVIZ que se conecta em uma rede Wi-Fi e é gerenciada através do aplicativo de celular EZVIZ. Adquirido pelo Mercado Livre.
- **GeekMagic Clock Smalltv-Ultra**
  Relógio digital de mesa com diversas funcionalidades, conectado em uma rede Wi-Fi. Adquirido pelo Mercado Livre.

Por conta do modo de funcionamento desses dispositivos, que são conectados a uma rede Wi-Fi e suas funcionalidades em maior parte são limitadas à ela, destaca-se que o objetivo dos testes é avaliar a segurança embutida nos dispositivos e nos serviços implementados por eles ou adjacente a eles; não foram executadas técnicas para invasão da rede doméstica, os cenários desenvolvidos são delimitados pela presença inicial de um agente malicioso na rede em que os dispositivos se encontram. Dessa forma, o laboratório de testes montado para este trabalho é composto por:

- Rede Wi-Fi doméstica isolada, operando exclusivamente na frequência de banda 2,4 GHz;
- Dispositivo alvo;
- Máquina virtual com instalação padrão do Kali Linux para Oracle VirtualBox, configurada com 8GB de memória base, 2 processadores e placa de rede em modo *Bridge*.

Deve ser ressaltado que a rede configurada com as definições padrões através do painel de roteador da CLARO possui segurança WPA2 (TKIP). Esse protocolo de criptografia (*Wi-Fi Protected Access 2* com *Temporal Key Integrity Protocol*) é considerado inseguro, pois utiliza os mesmos mecanismos de segurança que o WEP (*Wired Equivalent Privacy*). O WEP é vulnerável ao ataque de recuperação da chave MIC (*Message Integrity Code*) que, se executado, permite o atacante a transmitir e descriptografar pacotes arbitrários na rede [21]. Esse fato fortalece a factibilidade de uma cadeia de ataque a partir de, entre outros fatores, configurações padrão e a possível desatenção do usuário em relação a essas.

## 5.1 Estratégia de testes

A partir do laboratório construído para a cobertura do escopo elencado, os testes e análises realizados para este trabalho são como descrito:

1. Análise de *firmware*
   Extração e análise do *firmware* para detectar vulnerabilidades conhecidas e informações sensíveis, como *backdoors*, credenciais ou chaves de API embutidas;
   Verificação da presença de proteções contra ataques comuns.

2. Testes de comunicação de rede
   Captura e análise de tráfego entre dispositivos para avaliar a criptografia e identificar dados que podem ser exfiltrados;
   Identificar possibilidades de ataques de *replay* e potenciais redes mal configuradas;

3. Testes de resiliência
   Avaliar a capacidade dos dispositivos e serviços de resistirem a ataques de negação de serviço (*DoS*);
   Identificar quaisquer falhas que possam levar à indisponibilidade ou degradação do desempenho do dispositivo.

4. Testes de configuração e gerenciamento de dispositivos
   Avaliação das configurações padrão e práticas de segurança recomendadas;
   Identificação de credenciais padrão e possibilidade de configuração insegura.

5. Testes de segurança de aplicações
   Testes voltados a estratégias *web* e *AppSec (Application Security)* nas aplicações, se existirem, que se comunicam com os dispositivos.

Este escopo de testes de intrusão visa cobrir uma gama abrangente de aspectos de segurança dos dispositivos IoT que utilizam, em alguma capacidade, as tecnologias estudadas. Ao abordar desde a análise de *firmware* até a segurança das comunicações e aplicações envolvidas, pretende-se fornecer uma visão holística das vulnerabilidades e medidas de mitigação necessárias para proteger esses sistemas complexos e interconectados.

Finalmente, é salientado que um teste de segurança deve considerar um recorte temporal do estado de segurança de um ativo. Isso significa que as análises feitas refletem os ativos avaliados em seu estado momentâneo - entendendo que mudanças são constantes e novas vulnerabilidades surgem com o passar do tempo. Assim, as conclusões e recomendações refletem as informações coletadas durante o período de execução do trabalho. Sabe-se também que muitas informações ou descobertas apresentadas podem

servir como base para ataques que envolvem a esfera das relações humanas - sendo assim, é relevante que tenha-se um olhar crítico e atencioso sobre toda e qualquer avaliação e teste feitos. Adicionalmente, as avaliações, por serem com tempo limitado e determinado, não permitem uma avaliação completa de todos os controles de segurança. Foi priorizada a avaliação para identificar os controles de segurança mais fracos que um agente de ameaça exploraria. Nesse sentido, é proposta a realização de avaliações semelhantes como trabalhos futuros, visando constantemente desenvolver a segurança na área de Internet das Coisas.

# 6. RESUMO DOS TESTES

Este capítulo descreverá o funcionamento básico e configuração dos dispositivos testados, o processo de testes realizado, bem como enunciação das ferramentas utilizadas nesse processo, e a síntese das descobertas feitas no decorrer do trabalho.

## 6.1 Câmera de Segurança EZVIZ C6N

A análise do dispositivo iniciou-se com o processo de vinculação e configuração da câmera. Para isso, foi necessário instalar o aplicativo EZVIZ para smartphone. Pelo aplicativo, o dispositivo é adicionado escaneando o código QR em um adesivo embaixo da câmera, este que também contém o número de série e o código de verificação de 6 letras do dispositivo. O código QR armazena 4 informações: "ezvizlife", número de série, código de verificação e modelo da câmera.

Após o escaneamento, o aplicativo informou que o dispositivo já estava vinculado a outra conta, exibindo o endereço de e-mail parcialmente ofuscado da orientadora, que previamente havia realizado o emparelhamento a fim de averiguar o funcionamento do dispositivo.


Figura 6.1 – Aplicativo exibe o endereço de e-mail da vinculação anterior parcialmente ofuscado

Ao fazer a "solicitação de desvinculação", o aplicativo exige apenas que o usuário assinale que é o dono do dispositivo. Após isso, a desvinculação é concluída e o dispositivo pode ser adicionado novamente. Não houve interação necessária ou notificação para o endereço de e-mail da orientadora.

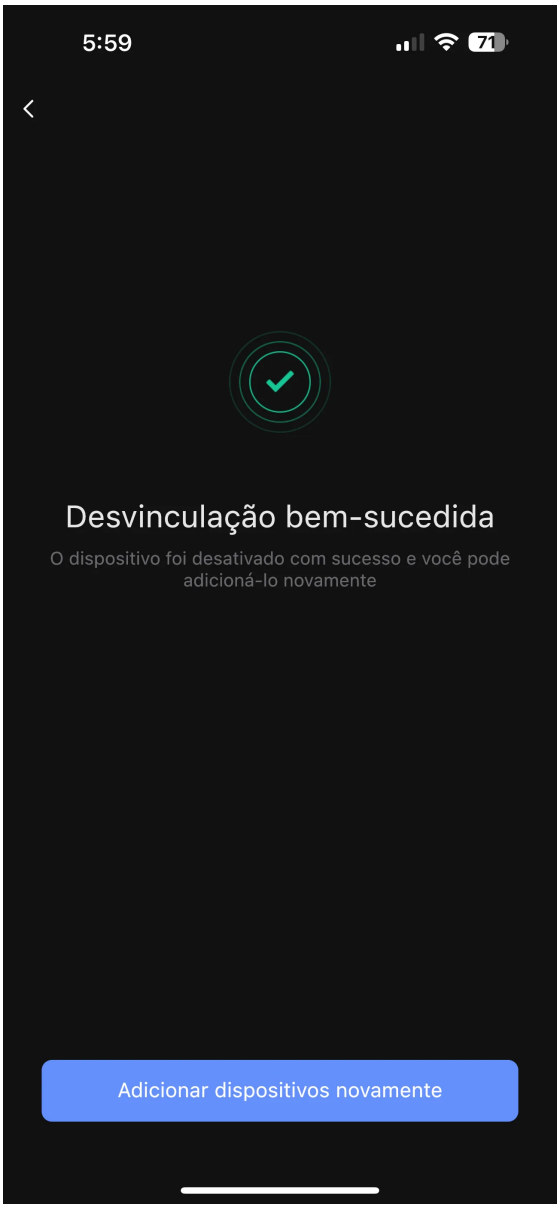


Figura 6.2 – Câmera desvinculada da conta sem confirmação

Dado esse comportamento e as informações contidas no QR code, o comprometimento das informações no adesivo na base da câmera pode levar ao comprometimento total do acesso. É necessário cuidado por parte do usuário especialmente em contextos não domésticos onde pode haver acesso físico ao dispositivo pelo ambiente, como é o caso em alguns estabelecimentos.


Figura 6.3 – Câmera EZVIZ CS-C6N encontrada em local público

Com a câmera conectada na rede, foi iniciada a fase de reconhecimento através de varreduras manuais e automatizadas.

## Mapeamento da Superfície de Ataque

Enumerar o endereço IP do dispositivo é trivial; utilizando um *sniffer* de rede como o Wireshark para analisar o tráfego, é possível ver múltiplos pacotes TCP originários da câmera, que é confirmado pela resolução de endereço físico do programa. Além disso, o aplicativo EZVIZ possui uma função de "exibição em tempo real de LAN", que ativamente procura dispositivos na mesma rede através da porta 8000, que hospeda o servidor de comando do SDK (*Software Development Kit*) da câmera e permite que um usuário acesse a câmera.

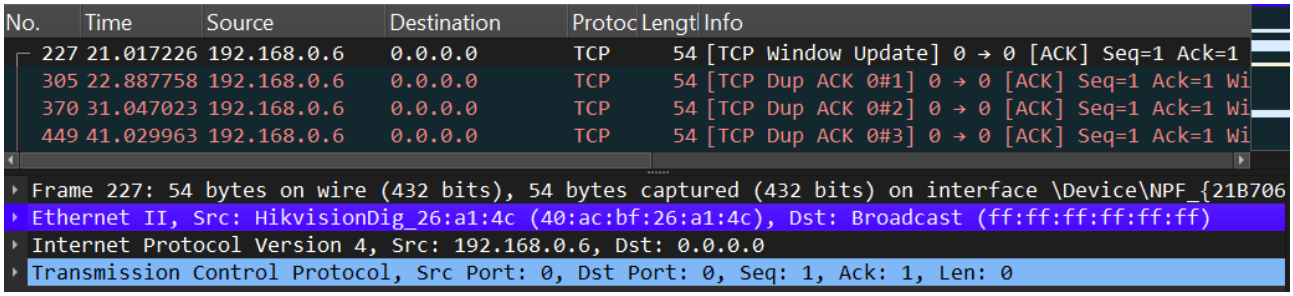


Figura 6.4 – Câmera associada ao nome "HikvisionDig"

Deve ser notado que, apesar de a câmera não ser da marca Hikvision, a EZVIZ é uma subsidiária da Hikvision que foca em dispositivos inteligentes residenciais. Por isso, as tecnologias subjacentes dos dispositivos EZVIZ são proprietárias da Hikvision e mantém os mesmos nomes.

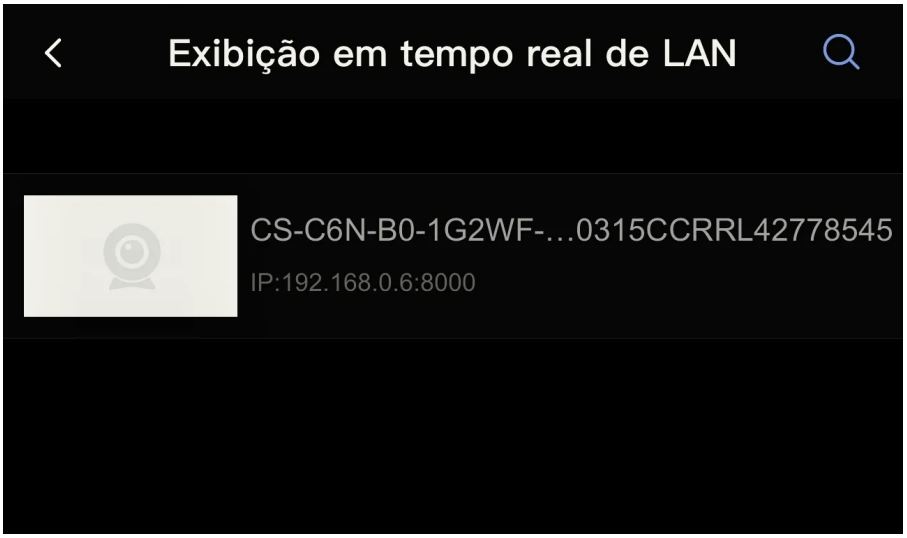


Figura 6.5 – Endereço IP da câmera visível pela função de "LAN Live View"

Sabendo o endereço IP da câmera, podemos executar uma varredura de portas usando o nmap. Com o comando `nmap -T4 -sV -sC -Pn 192.168.0.6`, é executada uma varredura nas 1000 portas mais comuns, com as *flags* `-T4` para maior agressividade resultando em uma varredura mais rápida, `-sV` para enumeração de versão dos serviços, `-sC` para usar *scripts* de enumeração comuns e *-Pn* para ignorar testes de *ping*, então assumindo que o *host* está ativo. Com isso, obtemos o seguinte resultado:

```
Host is up (0.022s latency).
Not shown: 996 filtered tcp ports (no-response)
PORT     STATE SERVICE   VERSION
554/tcp  open  rtsp      Hikvision 7513 POE IP camera rtspd
8000/tcp open  http-alt?
8200/tcp open  trivnet1?
9010/tcp open  sdr?
1 service unrecognized despite returning data. If you know the service/
   version, please submit the following fingerprint at https://nmap.org
   /cgi-bin/submit.cgi?new-service :
SF-Port8000-TCP:V=7.94%I=7%D=10/12%Time=670B2751%P=x86_64-pc-linuxgnu%
   SF:r(Kerberos,10,"\0\0\0\x10\0\0\0\r\0\0\0\r\0\0\0\0")%r(SMBProgNeg
   ,10,"\0\0\0\x10\0\0\0\r\0\0\0\r\0\0\0\0");
Service Info: Device: webcam
```

```
Service detection performed. Please report any incorrect results at
   https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 176.45 seconds
```

Listing 6.1 – Saída da varredura com `nmap`

Com isso, descobre-se que a câmera possui um serviço de RTSP (*Real Time Streaming Protocol*) na porta 554 para integração com outros programas para *streaming* de vídeo, como o *VLC Media Player*. O RTSP é um protocolo que opera na camada de aplicação e transmite áudio e vídeo em tempo real pela rede, assim permitindo o usuário a acessar o *feed* da câmera de outros dispositivos, no entanto, sem poder controlar a câmera. Após buscas na Internet, constata-se que a porta 8200 serve para acesso remoto à câmera, conjuntamente à porta 8000, e a porta 9010 é utilizada na comunicação entre o aplicativo móvel e a câmera. Também foi encontrada uma página de suporte da EZVIZ instruindo sobre a utilização de uma câmera como *webcam*, em que consta o formato da URL (*Uniform Resource Identifier*) para utilização do RTSP: `rtsp://admin:verification code@device ip:554/ch1/main`

Assim, são identificadas as credenciais padrão para acesso ao *feed* da câmera, com o usuário sendo `admin` e a senha sendo o código de verificação do dispositivo, encontrado no adesivo embaixo da câmera e no código QR. No aplicativo, não foi encontrada nenhuma opção que permitisse alterar essas credenciais, e após buscas na Internet, constatou-se que não há nenhuma maneira de alterá-las [6]. O aplicativo possui uma opção para habilitar a criptografia de vídeo no dispositivo, que requer a criação de uma senha que serve de chave pré-compartilhada (*PSK - pre-shared key*) para criptografar os pacotes de vídeo na rede, no entanto, não altera as credenciais de acesso para o RTSP ou controle remoto pela funcionalidade LAN Live View. Observa-se também que a configuração estava desabilitada por padrão.

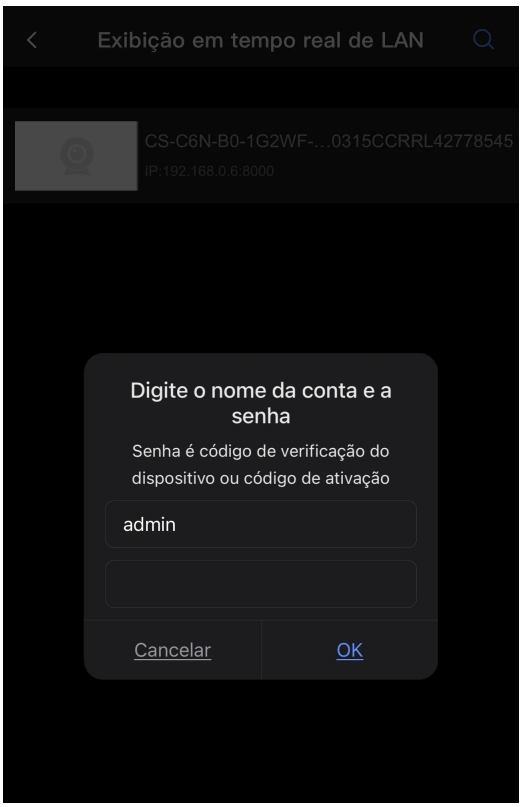


Figura 6.6 – *Prompt* de acesso ao controle remoto possui o usuário padrão "admin"

Ainda, foi confirmado que o código de verificação sempre consiste de 6 letras maiúsculas [6], o que configura uma entropia razoavelmente baixa. Com alguns experimentos, também foi observado que não há qualquer bloqueio após múltiplas tentativas falhas.

Por fim, pelo aplicativo EZVIZ, ao acessar as configurações da câmera, é possível ver que a versão do *firmware* instalado é a V5.3.0 build 220623. Essa versão possui algumas vulnerabilidades conhecidas, como as CVEs CVE-2023-34552 [22] e CVE-2023-48121 [23], conforme documentado na NVD (*National Vulnerability Database*), um banco de dados que registra vulnerabilidades conhecidas de diversas tecnologias, mantido pela NIST (*National Institute of Standards and Technology*) dos Estados Unidos. A partir de então, os testes foram direcionados à modelagem de ameaça e subsequente exploração dos vetores de ataque identificados.

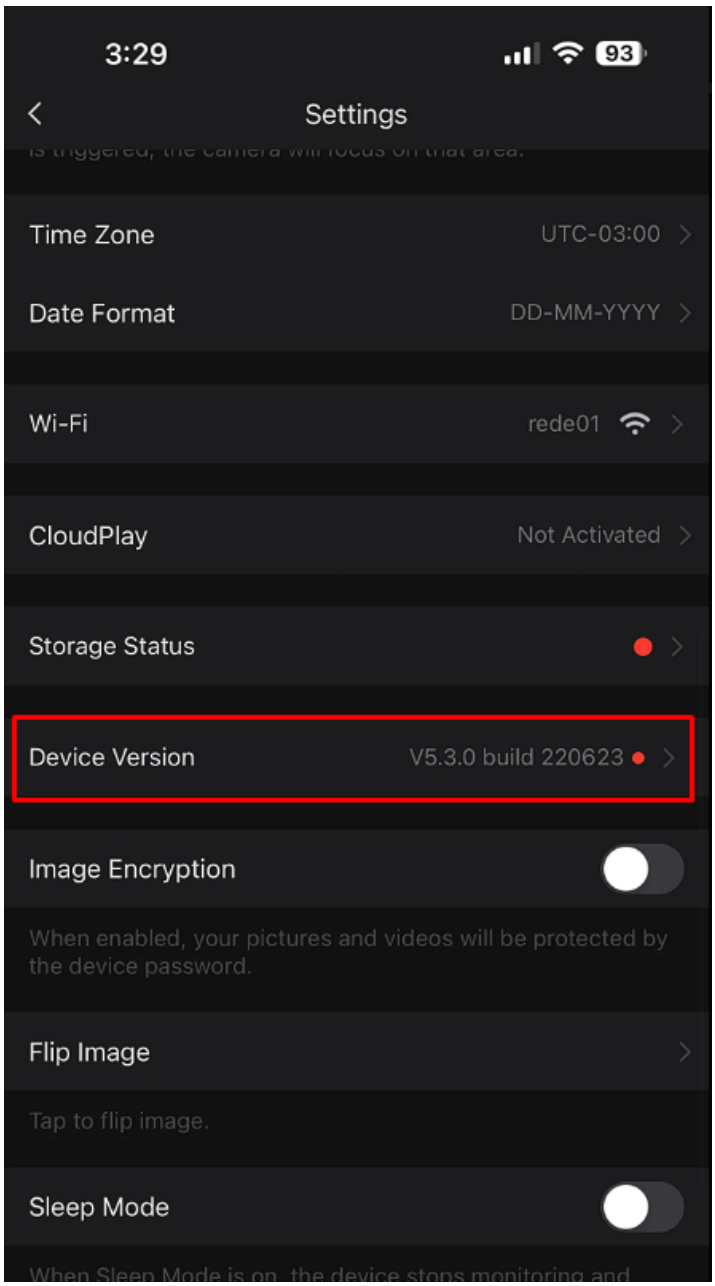


Figura 6.7 – Aplicativo EZVIZ mostrando a versão do dispositivo como V5.3.0 build 220623.

**Enumeração do código de verificação pelo RTSP**

Como um possível caminho para obter um acesso indevido, foi estudada a viabilidade da enumeração do código de verificação via força bruta para acessar a câmera. Se há apenas uma possibilidade para o nome de usuário e a senha é composta de 6 letras maiúsculas de A a Z podendo haver repetições, o total de combinações a ser testado em uma câmera é de $26^6$ = 308.915.776. Para a execução do ataque, foram encontradas algumas ferramentas no Github para enumerar credenciais por RTSP que funcionam de maneira similar; são enviadas múltiplas requisições para a porta 554 no endereço IP do alvo utilizando o método *DESCRIBE* do RTSP, que recebe uma URL e retorna diversos dados da

*stream* de vídeo. Analisando a resposta *HTTP*, um código *401 Unauthorized* significa que as credenciais são incorretas e o código *200 OK* sinaliza credenciais válidas.

Assim, foi elencada a ferramenta *rtsp_authgrinder* para a execução do teste, que recebe uma combinação de usuários e senhas para realizar o ataque em um endereço especificado, e escrito um código em Python para gerar todas as combinações de senha possíveis e escrevê-las para um arquivo-texto. O código da ferramenta foi alterado apenas para mostrar uma contagem do tempo de execução, em segundos, após encontrar uma credencial válida.

```
import itertools

def generate_combinations(filename):
    letters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    # Progress tracking
    total_combinations = len(letters) ** 6
    progress_interval = total_combinations // 20

    with open(filename, 'w') as file:
        for i, pin in enumerate(itertools.product(letters, repeat=6)):
            file.write(''.join(pin) + '\n')

            if (i + 1) % progress_interval == 0:
                progress = ((i + 1) / total_combinations) * 100
                print(f"Progress: {progress:.0f}%")

generate_combinations("pins.txt")
print(f"Combinations written to pins.txt")
```

Listing 6.2 – Código em Python para a geração e escrita de todas as combinações possíveis de códigos de verificação para câmeras EZVIZ.

Na primeira execução do ataque, a câmera foi tirada do ar em alguns segundos, resultando em erros na ferramenta e o aplicativo EZVIZ mostrando que a câmera estava *offline*. Foi necessário reiniciar a câmera para tentar o ataque novamente. Após isso, a fim de realizar um teste em menor volume, foi criada uma lista menor contendo apenas 1000 combinações, com a 1000$^a$ sendo a correta. Iniciando o ataque novamente, a credencial foi obtida em 115.72 segundos.

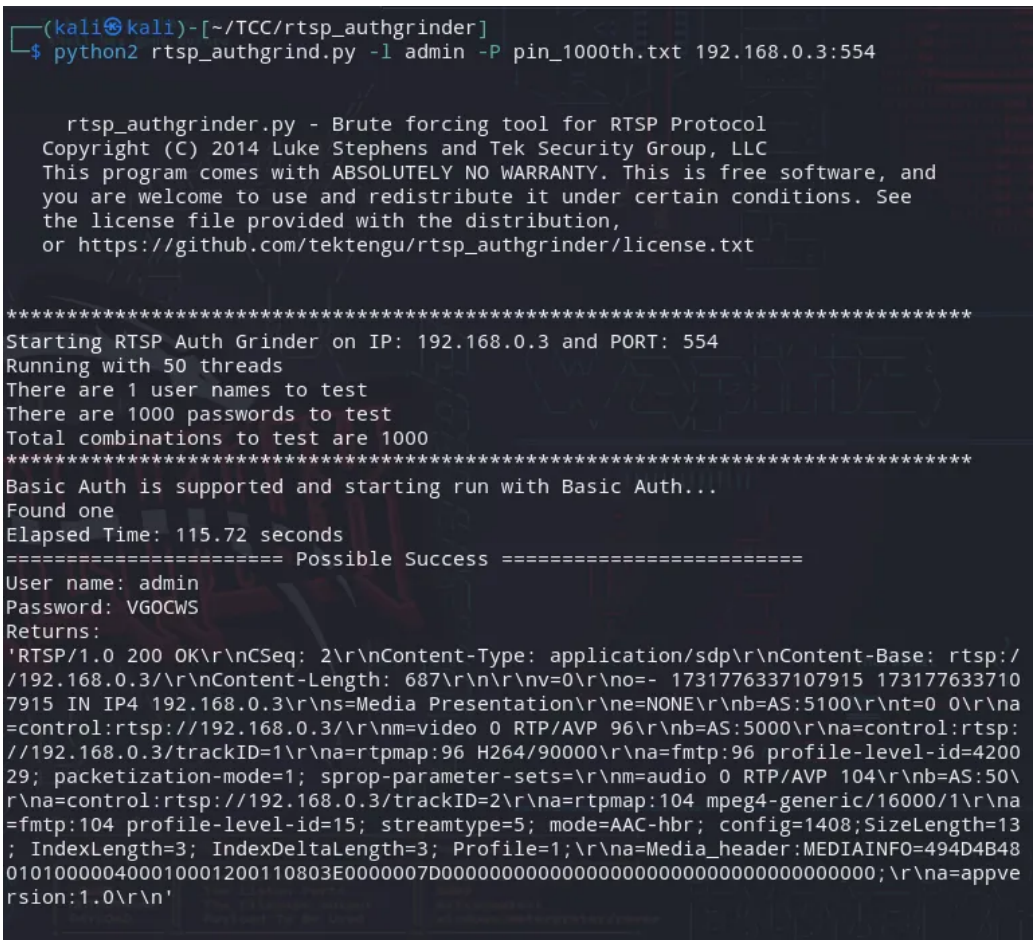


Figura 6.8 – Resultado da execução com 1000 tentativas

Considerado o comportamento da câmera no primeiro teste e o tempo para realizar o ataque com apenas 1000 entradas, foi postulado que há uma limitação na capacidade de processamento da câmera em si, em oposição à capacidade da máquina virtual, o que impõe uma restrição no tempo de execução do ataque. Então, foi testada a execução do mesmo ataque com mais uma máquina simultaneamente, dividindo 500 combinações entre as duas. A máquina virtual, que possuía a parte da lista com a senha correta, obteve 109.95 segundos no teste, uma melhoria marginal, o que indica que de fato há um gargalo na capacidade computacional da câmera.

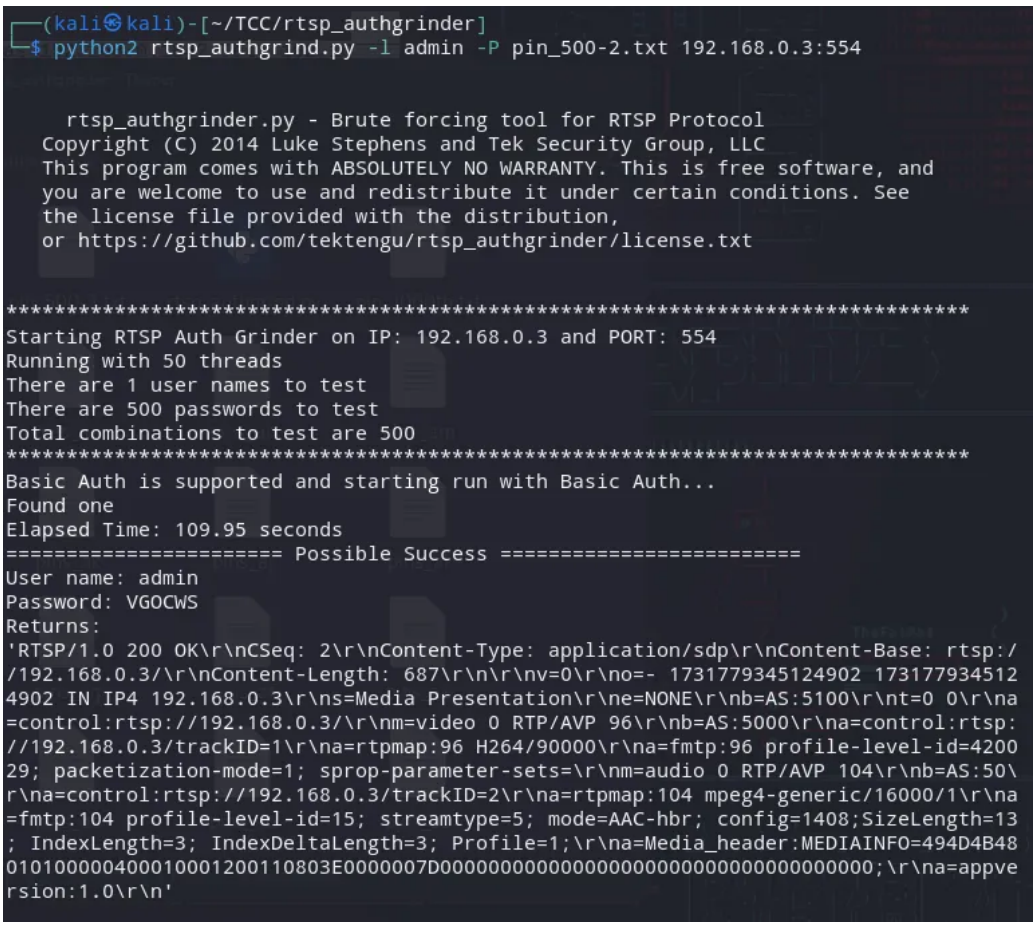


Figura 6.9 – Resultado da execução com 500 tentativas em cada máquina

Com esse desempenho, o tempo de execução médio para o pior caso de execução desse ataque seria em torno de 393 dias, porém, a instabilidade do dispositivo poderia requerer uma limitação a mais na taxa de requisições, aumentando ainda mais o tempo, o que faz essa estratégia ser pouco viável. No entanto, ressalta-se que isso se dá pela dependência do poder de processamento do dispositivo em si; o aplicativo EZVIZ conta com um serviço pago de armazenamento em nuvem, em que as imagens são armazenadas remotamente e criptografadas. Em um *whitepaper* publicado pela empresa de segurança Bitdefender em 2022 [26], é relatado que foi possível obter imagens da câmera através de uma vulnerabilidade de IDOR em um *endpoint* da API desse serviço e, como a PSK padrão utilizada para criptografar as imagens é o próprio código de verificação, realizar o ataque de força bruta localmente eliminaria o problema de gargalo imposto pela câmera.

Esse vetor de ataque pode ser mitigado habilitando a funcionalidade de criptografia de vídeo, em que o usuário pode definir uma senha forte para criptografia. Apesar disso, o mesmo *whitepaper* também descreve a possibilidade de obter essa chave de criptografia através de outro *endpoint* da API apenas tendo o número de série da câmera. No entanto, para o escopo deste trabalho, serviços pagos como o armazenamento em nuvem não foram testados. Em suma, deve ser observado que a entropia na geração de um dado sensível como o código de verificação é muito baixa e, portanto, suscetível a ser quebrada por ata-

ques de força bruta - mas o meio disponível para executar o ataque impõe uma restrição tecnológica na sua eficiência.

**Negação de Serviço**

Baseado no comportamento observado com o último teste, foi postulado que ataques de negação de serviço contra o dispositivo possam ser bastante eficazes. Assim, é considerado o seguinte cenário: Um agente de ameaça que tem o conhecimento da câmera e a intenção de eliminar a segurança providenciada por ela poderia adotar essa estratégia para isso? Ou seja, a resposta desse questionamento deve avaliar se a negação de serviço pode, consistentemente, degradar a operação da câmera o suficiente para que a segurança fornecida por ela seja obsoletada. Caso positivo, esse ataque poderia ser coordenado para capacitar outras ações maliciosas, como invasão do ambiente originalmente protegido pela câmera.

Para atender a esse teste, foi elencada a estratégia de *flooding*, em que o atacante envia um grande volume de tráfego a fim de sobrecarregar a rede ou sistema alvo e, assim, impedir que tráfego legítimo seja processado pelo alvo. Utilizando a ferramenta hping3, foram executados ataques de *flooding* de pacotes UDP, ICMP (*Internet Control Message Protocol*), SYN e *ping* da morte, em que são enviados pacotes ICMP extremamente grandes fragmentados para contornar limites, porém ao serem reagrupados pelo sistema alvo, excedem o tamanho máximo permitido pelo protocolo, provocando a negação de serviço.

Para executar o ataque, foram utilizados os seguintes comandos:

1. ***SYN flood*** - `hping3 -c 15000 -d 120 -S -w 64 --flood 192.168.0.3`
   Envia 15000 pacotes (`-c`) de 120 bytes cada (`-d`), especificando a *flag* SYN (`-S`) com tamanho da janela TCP de 64 (`-w`) e `--flood` para enviar os pacotes o mais rápido possível sem esperar resposta do alvo.

2. ***UDP flood*** - `hping3 --udp --flood -f 192.168.0.3`
   Pacotes udp (`--udp`) com fragmentação (`-f`).

3. ***ICMP flood*** - `hping3 -S --icmp --flood 192.168.0.3`
   Pacotes ICMP (`--icmp`), ping genérico.

4. **Ping da morte** - `hping3 -d 65538 -S -f --flood 192.168.0.3`
   Pacotes de 65538 bytes fragmentados.

Após diversas execuções, é possível afirmar que todos os ataques foram eficientes para cumprir o objetivo do cenário proposto. Em especial, os ataques de SYN *flood* e ping da morte foram particularmente efetivos em rapidamente travar a transmissão do dispositivo, algumas vezes o deixando *offline*. Nos testes de ping e UDP *flooding*, as imagens ao

vivo da câmera, tanto pela visualização no aplicativo quanto pelo serviço de RTSP, foram gradualmente atrasando a um ponto de parada total. Ao parar o ataque, ainda é necessário reconectar ao *feed* da câmera para que as imagens voltem ao normal. Esse comportamento também é visível nos vídeos gerados pela função de gravação do aplicativo. Nos demais ataques, a imagem da câmera imediatamente congela e a gravação é terminada. Em uma instância de SYN *flood*, a câmera apareceu como *offline* pelo aplicativo e precisou ser manualmente reinicializada depois de concluído o ataque. Assim, concluiu-se que ataques de negação de serviço podem consistentemente inutilizar a câmera, potencialmente fornecendo uma abertura para outras atividades maliciosas.

### Investigação de vulnerabilidades conhecidas

Foram estudadas algumas CVEs que estão atreladas ao mesmo modelo e versão de *firmware* do dispositivo e que possuem boa documentação disponível online a fim de reproduzir as vulnerabilidades. A CVE-2023-48121 afeta diversos modelos de câmeras Hikvision e EZVIZ, e foi descoberta a partir de testes em um dispositivo de modelo CS-C6N-B0-1G2WF nas versão de *firmware* V5.3.0 build 230215 e inferiores - a mesma situação do dispositivo desta pesquisa. A vulnerabilidade envolve um *bypass* de autenticação que permite que um atacante execute funções da câmera sem fornecer credenciais ou chaves de segurança; isso é realizado a partir das funcionalidades executadas pelo aplicativo EZVIZ através do protocolo proprietário na porta TCP 9010, como citado anteriormente, chamado de *Ezviz Command Port*. Normalmente, as requisições do aplicativo no celular para a câmera são criptografadas com AES-128 utilizando uma PSK, no entanto, o cabeçalho da mensagem não é criptografado e contém alguns metadados como o identificador do comando e uma *flag* para que seja criptografado o corpo da mensagem, que contém atributos para a execução do comando.

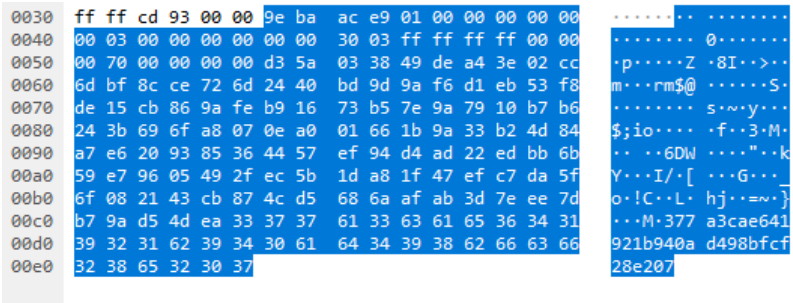


Figura 6.10 – Pacote criptografado do aplicativo para a câmera, capturado pelo Wireshark

Foi descoberto, porém, que a criptografia não é obrigatória para que a mensagem seja aceita pela câmera, permitindo que uma requisição em texto claro sem autenticação seja enviada e processada pela câmera. Isso resulta na câmera enviar respostas das requisições em texto claro também. O mesmo esquema de comunicação é utilizado para a

função de *snapshot*, que captura uma imagem da câmera e a envia. Essa função conta com uma opção adicional de criptografia para proteger a confidencialidade da imagem, porém essa opção também não é imposta na requisição, permitindo que um atacante construa uma requisição XML para obter uma imagem da câmera em formato JPEG (*Joint Photographic Experts Group*) em texto claro sem autenticação.

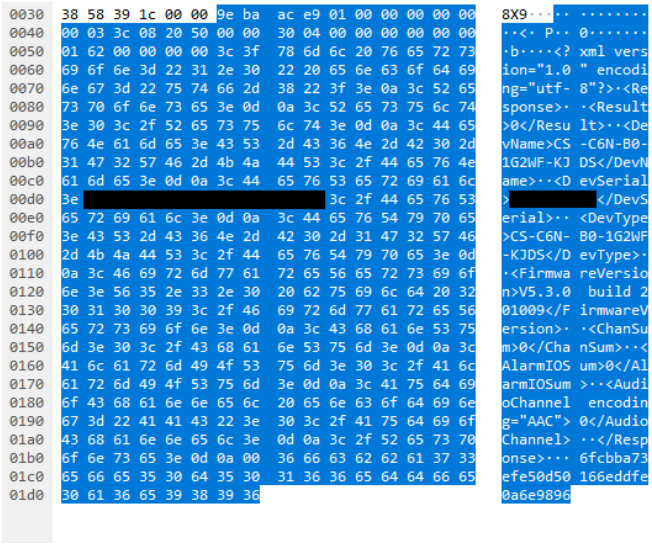


Figura 6.11 – Resposta em texto claro da câmera, como documentado em [37]

Por fim, o pesquisador responsável por essas descobertas, em acordo com a EZ-VIZ, não divulgou as *PoCs* (*Proof of Concept*) das vulnerabilidades e ofuscou o formato das requisições vulneráveis [37]. Após diversas buscas, também não foi possível encontrar a requisição na Internet, portanto mesmo com um dispositivo supostamente vulnerável em mãos, não foi possível testar essa exploração.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <Request>
3      <[redacted]>
4  </Request>
```

Figura 6.12 – Formato de requisição ofuscado

A outra vulnerabilidade conhecida, CVE-2023-34552, descoberta pelos pesquisadores Octavio Gianatiempo e Javier Aguinaga, se trata de uma vulnerabilidade de *stack buffer overflow* que levam à execução de código remota na câmera. Ela está atrelada a diversos modelos, incluindo o CS-C6N-B0-1G2WF com *firmware* em versões inferiores à V5.3.0 build 230215, como é o caso deste dispositivo, e possui criticidade alta de acordo com o CVSS V3.x [8].

De acordo com os pesquisadores, o intuito inicial da pesquisa era obter o *firmware* da câmera a fim de extraí-lo e realizar uma análise, porém ele não é disponibilizado publicamente. Então, ao interceptar o tráfego de rede durante uma atualização do dispositivo pelo

aplicativo EZVIZ, foi possível capturar uma requisição HTTP para um *endpoint* de *download* de *firmware*: `http://(sa|us)download.ezvizlife.com/device/[model]/2.0/[model].dav`. Ao realizar o mesmo teste no laboratório utilizando o *Wireshark*, não foi possível obter essa requisição, no entanto, pode-se confirmar que o *endpoint* continua ativo após conseguir fazer o *download* do *firmware* para o modelo de câmera adquirido pela URL `http://download.ezvizlife.com/device/CS-C6N-B0-1G2WF/2.0/CS-C6N-B0-1G2WF.dav`


Figura 6.13 – Comando `wget` para o *endpoint* faz *download* do arquivo binário .dav da última versão de *firmware* para o modelo de dispositivo especificado com sucesso

Assim, utilizando a ferramenta *Binwalk*, é possível extrair diversas informações do arquivo binário - porém o arquivo não contém símbolos, ou seja, não há metadados que descrevem nomes de funções, variáveis ou outras estruturas internas do programa, o que dificulta a depuração do código-fonte para engenharia reversa. Para isso, os pesquisadores enumeraram diversos *endpoints* para *download* de outras versões de *firmware* e, com a ferramenta *Bindiff*, compararam as diferenças entre todas as versões para parear funções e aplicar símbolos para a versão desejada a partir de padrões encontrados. Dessa forma, foi possível reconstruir o código-fonte utilizado no *firmware*. A partir da análise do código, foram encontrados pontos com chamadas inseguras em uma função relacionada ao protocolo proprietário da Hikvision, o SADP (*Search Active Devices Protocol*).

O SADP é um protocolo não documentado que utiliza primariamente comunicações com XML (Extensible Markup Language) que permite ativar câmeras, configurar redes, entre outras funcionalidades. Normalmente, essas funções são realizadas através do software de mesmo nome disponibilizado pela Hikvision.

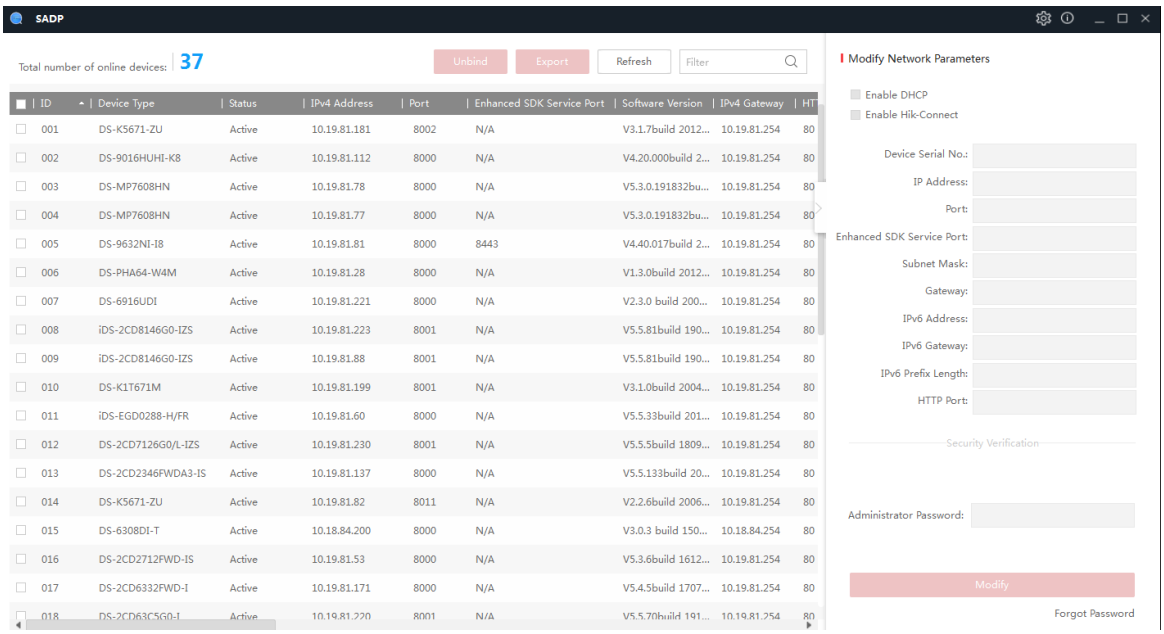


Figura 6.14 – Software SADP, utilizado para gerenciamento de dispositivos

Analisando o tráfego na rede, pode-se observar que a comunicação é iniciada pela ferramenta enviando um *payload* do tipo "*inquiry*"para o endereço multicast 239.255.255.250 na porta UDP 37020, ao qual a câmera responde com diversas informações sobre si mesma, incluindo modelo, número de série, endereço MAC, versão de *firmwmare* e data de inicialização, essas que, então, constam na listagem de dispositivos na ferramenta.

```
<?xml version="1.0" encoding="utf-8"?><Probe><Uuid>4EC8EADC-E4A1-5449-
   A941-19487CA60F4F</Uuid><Types>inquiry</Types></Probe>
```

Listing 6.3 – Conteúdo do pacote de *discovery* SADP

```
<?xml version="1.0" encoding="UTF-8" ?>
<ProbeMatch>
<Uuid>4EC8EADC-E4A1-5449-A941-19487CA60F4F</Uuid>
<Types>inquiry</Types>
<DeviceType>798982</DeviceType>
<DeviceDescription>CS-C6N-B0-1G2WF-BR</DeviceDescription>
<DeviceSN>CS-C6N-B0-1G2WF-BR0120230315CCRRL42778545</DeviceSN>
<CommandPort>8000</CommandPort>
<HttpPort>80</HttpPort>
<MAC>40-ac-bf-26-a1-4c</MAC>
<IPv4Address>192.168.0.6</IPv4Address>
<IPv4SubnetMask>255.255.255.0</IPv4SubnetMask>
<IPv4Gateway>192.168.0.1</IPv4Gateway>
<DHCP>true</DHCP>
<AnalogChannelNum>0</AnalogChannelNum>
<DigitalChannelNum>1</DigitalChannelNum>
<SoftwareVersion>V5.3.0build 220623</SoftwareVersion>
```

```
<DSPVersion>V7.0 build 200519</DSPVersion>
<BootTime>1970-01-01 00:00:09</BootTime>
<OEMInfo>N/A</OEMInfo>
<EZVIZCode>true</EZVIZCode>
<manufacturer>CSWSHC</manufacturer>
<Activated>true</Activated>
<ResetAbility>false</ResetAbility>
<PasswordResetAbility>true</PasswordResetAbility>
<PasswordResetModeSecond>true</PasswordResetModeSecond>
</ProbeMatch>
```

Listing 6.4 – Resposta da câmera

Essa mesma comunicação é observada entre a câmera e o aplicativo EZVIZ no modo *LAN Live View*.

Com noção do funcionamento desse protocolo e munidos do seu código-fonte, descobriu-se que durante a comunicação, quando o protocolo recebe conteúdo com a *tag* `<MAC>`, é inicializado um *buffer* com tamanho fixo de 64 bytes, como o esperado para um endereço MAC; porém, não há tratamento caso a *string* recebida tem um tamanho maior que o esperado. Assim, a *string* malformada é passada como parâmetro para a função `convertMac` para *parsing*. Essa função, por sua vez, lê o endereço, converte cada *char* para hexadecimal e armazena em outro *buffer*. Caso seja encontrado um separador ('-', ':' ou ' '), é adicionado um 0 e o índice do *buffer* é incrementado - no entanto, não há uma verificação da quantidade de separadores, assim, se a função continuamente encontra separadores, o índice continua sendo incrementado até que ocorra um erro de *stack buffer overflow*.

```
int convertMac (char *dst, char *src) {
 dst_idx = 0;
 dst[0] = 0;
 src_idx = 0;
 while ( 1 ) {
 src_char = src[src_idx];
 if ( !src[src_idx] )
 break ;
 if (is_mac_sep (src_char)) { // -, : or space
 // Write 0 and increase index
 dst[++dst_idx] = 0;
 ++src_idx;
 } else {
 converted_char = from_hex(src_char);
 // Buffer overflow #2
 dst[dst_idx] = converted_char + 16 * dst[dst_idx];
 ++src_idx;
```

```
 }
 }
}
```

Listing 6.5 – Função `convertMac` vulnerável

Assim, ao enviar um *payload* XML contendo a *tag* `<MAC>` e uma *string* grande o bastante para o mesmo endereço de multicast, pode resultar em um *crash* do dispositivo. Porém, em algumas execuções do ataque, não ocorreram *crashes* e a fim de entender o comportamento do dispositivo nesses casos, os pesquisadores construíram uma imagem de *docker* a partir de uma *distro* de Linux com um *kernel* e versão de *gcc (GNU Compiler Collection)* similares para que seja compilado estaticamente e inserido manualmente no dispositivo, a fim de criar um *debugger* para depuração do comportamento.

O objetivo, então, passa a ser a criação de uma *toolchain* para compilar executáveis dinâmicos que já são utilizados pelo dispositivo, a fim de executar código arbitrário na câmera. Porém, o *firmware* implementa *Address Space Layout Randomization (ASLR)* para mitigar esse vetor de ataque. Essa técnica consiste na randomização do espaço de endereçamento em memória a cada execução, dificultando a localização do endereço de memória das bibliotecas necessárias. Assim, ideia é utilizar a superfície de ataque fornecida pelo *stack buffer overflow* para causar um vazamento de dados que forneça os endereços de memória necessários.

Com o *debugger* instalado na câmera, foi observado que ao fim da função vulnerável `convertMac`, é executada uma instrução `POP {R4-R11, PC}`. Isso significa que ao causar o *buffer overflow*, regulando o tamanho de entrada corretamente, é possível manipular os registradores R4 a R11 e o *program counter*. Após o retorno dessa função, a função anterior que havia a instanciado executa uma instrução `STR R4, [R5, R3]`, armazenando o conteúdo do registrador R4 nos endereços de memória apontados por R5 e R3. Ou seja, se os endereços na pilha que são usados para carregar os valores nos registradores forem sobrescritos com o *buffer overflow* mas sem alterar o PC, ao retornar para a função anterior, deve ser possível realizar uma escrita arbitrária, pois pode-se especificar o conteúdo de R4 e o direcionamento com R3 e R5.

Outra informação adquirida na análise do *firmware* é a existência de uma tabela de *structs* contendo as respostas para requisições do RTSP, com ponteiros para o código de status e a mensagem que o acompanha.

```
.data:002E4F94                DCD a200                ; "200"
.data:002E4F98                DCD aOk                 ; "OK"
.data:002E4F9C                DCD a201                ; "201"
.data:002E4FA0                DCD aCreated            ; "Created"
.data:002E4FA4                DCD a239255255250+0xC   ; "250"
.data:002E4FA8                DCD aLowOnStorageSp     ; "Low On Storage Space"
.data:002E4FAC                DCD a400300+4           ; "300"
.data:002E4FB0                DCD aMultipleChoice     ; "Multiple Choices"
.data:002E4FB4                DCD a301                ; "301"
.data:002E4FB8                DCD aMovedPermanent     ; "Moved Permanently"
.data:002E4FBC                DCD a302                ; "302"
.data:002E4FC0                DCD aMovedTemporari     ; "Moved Temporarily"
.data:002E4FC4                DCD a303                ; "303"
.data:002E4FC8                DCD aSeeOther           ; "See Other"
.data:002E4FCC                DCD a304                ; "304"
.data:002E4FD0                DCD aNotModified        ; "Not Modified"
.data:002E4FD4                DCD a305                ; "305"
.data:002E4FD8                DCD aUseProxy           ; "Use Proxy"
.data:002E4FDC                DCD off_277B50          ; "09"
.data:002E4FE0                DCD aBadRequest         ; "Bad Request"
.data:002E4FE4                DCD a401                ; "401"
.data:002E4FE8                DCD aUnauthorized       ; "Unauthorized"
```

Figura 6.15 – Tabela de respostas RTSP, como "401 Unauthorized"

Também foi identificada a *Global Offset Table (GOT)*, uma tabela que contém todos os ponteiros para as funções das bibliotecas em tempo de execução, já que os endereços são aleatorizados em cada execução.

```
.got:002E072C strcat_ptr      DCD __imp_strcat        ; DATA XREF: strcat+8↑r
.got:002E0730 _ZNSt8ios_base4InitC1Ev_ptr DCD __imp__ZNSt8ios_base4InitC1Ev
.got:002E0730                                         ; DATA XREF: std::ios_base::Init:
.got:002E0730                                         ; std::ios_base::Init::Init(void)
.got:002E0734 prctl_ptr       DCD __imp_prctl         ; DATA XREF: prctl+8↑r
.got:002E0738 feof_ptr        DCD __imp_feof          ; DATA XREF: feof+8↑r
```

Figura 6.16 – Tabela GOT contida no arquivo binário de *firmware*

Com isso, deve ser possível escrever um endereço que aponta para a tabela GOT na tabela de respostas RTSP, e assim, ao mandar uma requisição RTSP, os endereços de memória seriam vazados pela resposta da câmera. Assim, pode ser criada uma cadeia de ataque baseada em *Return Oriented Programming (ROP chain)* para enviar um *payload* em XML contendo um endereço MAC malformado e uma linha de código que o sistema executará como comando. O comando, por sua vez, faz a transferência de um arquivo malicioso para o dispositivo utilizando o cliente de TFTP (*Trivial File Transfer Protocol*) já contido na câmera. O TFTP é um protocolo para transferência de arquivos similar ao FTP (*File Transfer Protocol)* porém utiliza UDP, e é utilizado no dispositivo como funcionalidade do SDK disponibilizado para usuários avançados automatizarem algumas funções da câmera utilizando a linguagem C ou C#. Assim, é gerado um *payload* como:

```
tftp -r x -g 192.0.0.3 9069;chmod +x x;./x
#<?xml version="1.0"encoding="utf-8"?>
<Probe>
<Uuid>aaaaaa</Uuid>
<Types>reset</Types>
```

```
<MAC>----00-00-00-f8-3a-25-4c-9d-33-b
0-25-17---c8-e7-----</MAC>
</Probe>
```

O ataque obteve sucesso para os pesquisadores, que a partir disso desejavam criar um *script* de pós-exploração que levasse a interceptar o vídeo da câmera. Então, estudando o binário do *firmware*, foi identificado que cada funcionalidade da câmera, como o serviço de RTSP, opera em *threads* diferentes. Assim, esse processo poderia ser finalizado forçadamente sem afetar os demais processos da câmera e substituído por um outro que fornece uma *stream* de vídeo *poisoned*, controlada pelo atacante. Para isso, foi construído um *exploit* que transfere um executável malicioso que essencialmente derruba o serviço de RTSP temporariamente, estabelece um túnel entre a câmera e a máquina do atacante, essa que hospeda um servidor para transmitir uma *stream* qualquer, e então reinicia o RTSP para transmitir a *stream* do atacante.

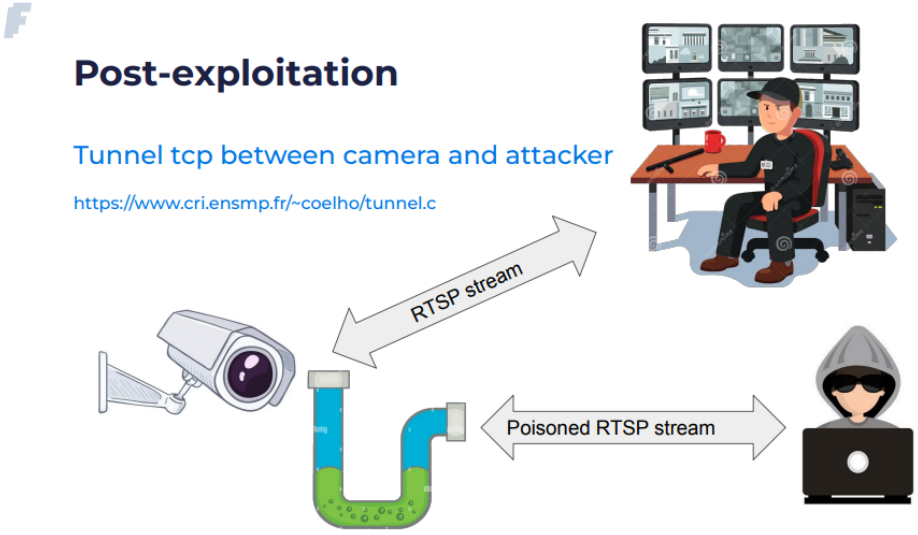


Figura 6.17 – Slide explicando o *exploit* utilizado pelos pesquisadores

O ataque também obteve sucesso, com os pesquisadores demonstrando uma prova de conceito em que é iniciada uma conexão com o serviço de RTSP do dispositivo normalmente, porém é exibido um clipe da música "Never Gonna Give You Up"de Rick Astley no lugar do *feed* da câmera. Logo, o foco dos testes para este trabalho foram direcionados a replicar os feitos encontrados nessa pesquisa, visto que os pesquisadores disponibilizaram um repositório no *Github* contendo os *scripts* de *PoC* dos *exploits* e, de acordo com a descrição da CVE-2023-34552 [22], a versão de *firmware* instalada no dispositivo para testes é vulnerável a esses ataques.

Foram realizados os mesmos passos para a preparação da exploração: foi executado o *build* da imagem de *docker* disponibilizada que cria uma instância de *gdbserver* compatível com o *firmware* da câmera e os arquivos binários utilizados para a exploração

e pós-exploração. Esses insumos, então, são extraídos para a máquina *host* do *docker* para que sejam fornecidas por TFTP para a câmera. Para isso, foi estabelecido um servidor utilizando a ferramenta py3tftp que serve todos os arquivos presentes no diretório corrente.

Para o *exploit* de *poisoning* da *stream* RTSP, foi utilizada uma imagem *docker* `rtsp-simple-server` [7] indicada no repositório, que inicia um servidor de RTSP em que, utilizando a ferramenta ffmpeg, serve uma *stream* a partir de um arquivo de vídeo com o comando `ffmpeg -re -stream_loop -1 -i <vídeo>.mp4 -c copy -f rtsp rtsp://localhost:8554/H.264`. No entanto, ao executar o *exploit*, não ocorreu o comportamento esperado; não houve transferências no servidor de TFTP e, analisando o tráfego de rede, a câmera não responde o pacote de multicast enviado. Mesmo após um extenso processo de depuração, não foi possível observar o comportamento esperado da câmera. Então, foi aberto um *issue* no repositório a fim de buscar ajuda sobre a execução do ataque. Após alguns dias, o pesquisador Octavio Gianatiempo respondeu explicando que os *exploits* no repositório foram feitos como prova de conceito para uma câmera do modelo CS-C6N-A0-1C2WFR com *firmware* V5.3.0 build 22 e, por consequência, os *offsets* de endereçamento utilizados são válidos para o binário presente naquele *firmware* e não é esperado que funcione diretamente em outros *firmwares*. Além disso, em resposta a um e-mail enviado também para Gianatiempo, ele elabora que o *exploit* se trata de uma escrita arbitrária que possivelmente não está presente no binário que está sendo testado.

Isso significa que, nesse caso, seria necessário realizar um estudo de baixo nível do *firmware* sendo utilizado na câmera a fim de averiguar a presença das vulnerabilidades descritas e adaptar os códigos-fonte para aquele binário; no entanto, o *firmware* obtido pelo *endpoint* de *download* supracitado é sempre a última versão disponível e, portanto, certamente não contém as mesmas vulnerabilidades pois foram tratadas pela EZVIZ [8]. Com isso, esse caminho de testes foi encerrado para o escopo deste trabalho, no entanto, o estudo dessa vulnerabilidade em outros modelos de câmera é proposto como trabalho futuro, visto que mesmo um dispositivo EZVIZ adquirido recentemente possui *firmware* defasado e possivelmente vulnerável, mantendo a relevância dessas vulnerabilidades. Finalmente, os estudos e testes realizados em torno desse dispositivo trazem à luz as práticas de segurança da implementação da IoT por parte da EZVIZ. Por mais que alguns dos vetores de ataque mais sofisticados não puderam ser exploradas com sucesso, certamente existe uma superfície de ataque a ser explorada por um potencial agente de ameaça com acesso à rede doméstica. Entre a enumeração do dispositivo ser relativamente trivial, com o software SADP que exibe todas as informações pertinentes à câmera com exceção do código de verificação, a suscetibilidade da câmera a ataques de negação de serviço e a falta de controles de segurança nas funcionalidades adjacentes do dispositivo, como as credenciais fixas do serviço de RTSP, há diversas práticas de segurança que poderiam ser solidificadas a fim de limitar as opções de um agente malicioso, especialmente no que tange a um dispositivo de segurança crítico como uma câmera.

### 6.1.1 Sumário Técnico das Descobertas

Esta seção apresenta um sumário técnico das descobertas que representam comportamentos inesperados para um dispositivo seguro, com níveis de criticidade modelados para o contexto postulado no escopo de testes, bem como a mitigação recomendada para o fabricante do dispositivo.

**EZVIZ-01. Negação de Serviço**

| **Criticidade** | **Crítica** |
|---|---|
| **Descrição** | O *streaming* de vídeo da câmera é completamente comprometido ao enviar um grande volume de tráfego para o dispositivo, sendo possível também resultar em um *crash*. |
| **Informações gerais de probabilidade e impacto** | • **Probabilidade:** Alta, pois o ataque é simples de ser executado e requer pouco conhecimento prévio sobre o dispositivo.<br>• **Impacto:** Alto, pois resulta na indisponibilidade da câmera. |
| **Mitigação** | É recomendada a implementação de mecanismos para limitar ou ignorar requisições repetidas em um intervalo de tempo muito curto e restringir certos tipos de tráfego, como *pings*. |

Tabela 6.1: Resumo descritivo da descoberta EZVIZ-01.

**EZVIZ-02. Credenciais de RTSP enumeráveis**

| **Criticidade** | **Alta** |
|---|---|

| | |
|---|---|
| **Descrição** | Foram identificadas credenciais de acesso padrões para o serviço de RTSP que não podem ser alteradas pelo usuário, com o nome de usuário sendo "admin"e uma senha fraca composta apenas de 6 letras maiúsculas. Isso possibilita a enumeração das credenciais por força bruta, visto que o dispositivo não implementa proteções contra múltiplas tentativas de acesso. Ressalta-se que o RTSP também é habilitado por padrão apesar de ser uma funcionalidade secundária. |
| **Informações gerais de probabilidade e impacto** | • **Probabilidade:** Média, pois apesar de haver um número relativamente pequeno de combinações a serem testadas e a execução do ataque ser simples, a capacidade de processamento limitada da câmera drasticamente impacta a eficiência do ataque.<br>• **Impacto:** Alto, pois as credenciais concedem acesso ao vídeo e áudio da câmera. |
| **Mitigação** | Recomenda-se que o serviço de RTSP seja desativado por padrão, requerendo que o próprio usuário configure a funcionalidade manualmente e defina credenciais de acesso com uma política de senha forte. |

Tabela 6.2: Resumo descritivo da descoberta EZVIZ-02.

**EZVIZ-03. Firmware defasado com múltiplas CVEs**

| **Criticidade** | **Média** |
|---|---|
| **Descrição** | O *firmware* que já estava instalado na câmera é a versão V5.3.0 build 220623, que é vulnerável a diversas CVEs como a CVE-2023-34551, CVE-2023-34552 e CVE-2023-48121, que podem levar ao comprometimento das imagens da câmera. |

| | |
|---|---|
| **Informações gerais de probabilidade e impacto** | • **Probabilidade:** Baixa, pois os ataques são complexos e dependem de alguns fatores que não puderam ser verificados para a câmera testada.<br>• **Impacto:** Alto, pois a exploração das vulnerabilidades pode levar ao vazamento de imagens da câmera e *poisoning* do serviço de RTSP. |
| **Mitigação** | É recomendado que a atualização de *firmware* seja obrigatória durante a primeira configuração do dispositivo. |

Tabela 6.3: Resumo descritivo da descoberta EZVIZ-03.

**EZVIZ-04. Desvinculação do dispositivo sem necessidade de autorização**

| **Criticidade** | **Baixa** |
|---|---|
| **Descrição** | A câmera pode ser desvinculada de uma conta e vinculada a outra sem necessidade de confirmação da conta original a partir das informações contidas no código QR. Ressalta-se que essas informações podem ser obtidas através da rede local, com exceção do código de verificação, utilizando o *software* SADP ou manualmente enviando um pacote de descoberta do protocolo SADP em *broadcast*. |
| **Informações gerais de probabilidade e impacto** | • **Probabilidade:** Baixa, pois o atacante precisaria ter conhecimento prévio do código de verificação.<br>• **Impacto:** Médio, apesar de potencialmente conceder acesso à câmera, esse movimento pode ser facilmente detectado pois o usuário legítimo perderia acesso ao seu dispositivo. |
| **Mitigação** | É recomendado que exista alguma autorização necessária por parte da conta já vinculada para que o dispositivo seja vinculado a outra conta. |

Tabela 6.4: Resumo descritivo da descoberta EZVIZ-04.

**EZVIZ-05. *Endpoint* para *download* de *firmware* acessível**

| **Criticidade** | **Baixa** |
|---|---|
| **Descrição** | O *endpoint* que fornece a última versão de *firmware* para a atualização da câmera é acessível pela Internet sem necessidade de autenticação. |
| **Informações gerais de probabilidade e impacto** | • **Probabilidade:** Média, pois o formato da URL para o *endpoint* pode ser encontrado a partir de buscas na Internet.<br>• **Impacto:** Baixo, pois não gera um risco direto ao usuário, mas pode ser utilizado por agentes maliciosos para descobrir vulnerabilidades no *firmware*. |
| **Mitigação** | É recomendado que o *endpoint* implemente algum mecanismo de autenticação. |

Tabela 6.5: Resumo descritivo da descoberta EZVIZ-05.

**EZVIZ-06. Criptografia de vídeo desabilitada por padrão**

| **Criticidade** | **Baixa** |
|---|---|
| **Descrição** | A funcionalidade de criptografia de vídeo esta desabilitada por padrão, fazendo com que a transmissão de vídeo seja criptografada utilizando apenas o código de verificação como chave pré-compartilhada, que possui baixa entropia em sua geração. |
| **Informações gerais de probabilidade e impacto** | • **Probabilidade:** Baixa, pois o agente malicioso precisa obter uma imagem criptografada da câmera em um ataque relativamente complexo.<br>• **Impacto:** Alto, pois pode levar ao comprometimento do código de verificação e acesso a imagens da câmera. |

| | |
|---|---|
| **Mitigação** | Recomenda-se que essa funcionalidade seja habilitada por padrão, como sugerido também para a descoberta EZVIZ-02. |

Tabela 6.6: Resumo descritivo da descoberta EZVIZ-06.

## 6.2 GeekMagic Clock Smalltv-Ultra

Este dispositivo se trata de um relógio de mesa digital com uma tela OLED que se conecta a uma rede Wi-Fi para desempenhar algumas funções, como exibir condições climáticas, fotos e animações. A configuração inicial do dispositivo consiste em ligá-lo e conectar em uma rede Wi-Fi estabelecida por ele, e visitar o endereço fixo 192.168.4.1 em um navegador. Na página web, o usuário vê uma lista de redes Wi-Fi disponíveis e conecta o dispositivo na rede escolhida, inserindo a senha. Após isso, o dispositivo reinicia e automaticamente se conecta na rede Wi-Fi escolhida, exibindo o seu próprio endereço IP na tela. O usuário deve, então, acessar o endereço IP em um navegador para configurar o relógio.

A página web hospedada pelo dispositivo contém todos os controles do relógio, como definição de data, horário, brilho, *upload* de imagens, reinicialização do dispositivo e atualizações. Para atualizar o dispositivo, o usuário deve acessar um link na página que leva para um repositório no Github contendo diversas versões de *firmware*, baixar a versão desejada e fazer o *upload* para o relógio pelo painel web.

Peculiarmente, a página de configuração do dispositivo não possui medidas de segurança como senha de acesso ou criptografia, ou seja, qualquer usuário na rede local pode acessar e modificar as configurações do dispositivo. Isso significa que uma ação maliciosa como o *upload* de uma imagem para ser exibida no relógio com o intuito de perturbar a vítima, reminiscente do caso da invasão de aspiradores-robô mencionado anteriormente [9], é trivial para um agente malicioso presente na rede.

Todos os dados que precisam ser armazenados no dispositivos também não são criptografados. O painel de configuração conta com uma funcionalidade para usuários avançados para obter atualizações climáticas mais rapidamente a partir de uma API. O usuário pode colocar a sua chave de API para que o dispositivo realize as chamadas ativamente, no entanto, ela é armazenada em texto claro. O mesmo comportamento é observado ao acessar o *endpoint* `wifi.json`, que exibe o SSID (*Service Set Identifier*) e senha da rede Wi-Fi à qual o dispositivo conecta.

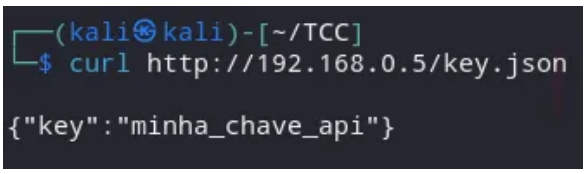


Figura 6.18 – Chave de API do usuário acessível pelo arquivo `key.json`

No entanto, foi necessário estudar o risco associado à funcionalidade de *upload* de *firmware*, pois a inserção de um binário malicioso pode representar uma gama de ataques potenciais contra o dispositivo e possivelmente comprometer a segurança da rede. Assim, como o *firmware* é disponibilizado publicamente, foi possível fazer o *download* do arquivo `FW-Smalltv-Ultra-V9.0.19.bin`. Ressalta-se que apenas o arquivo binário é disponibilizado, e não o código-fonte do *firmware*, por isso, foi necessário utilizar técnicas de engenharia reversa para tentar extrair informações sobre o *firmware*. Para essa finalidade, foi utilizada a ferramenta Binwalk.

**Análise de *firmware***

O comando `binwalk -e <file.bin>` tenta extrair arquivos ocultos contidos no arquivo binário com extensões conhecidas, como LZO (Lempel-Ziv-Oberhum, algoritmo de compressão de dados *lossless*), no entanto, o comando não gerou nenhuma saída. Então, para tentar extrair arquivos de qualquer tipo, foi executado o comando `binwalk --dd=".*"FW-Smalltv-Ultra-V9.0.19.bin --run-as=root`. Esta execução resultou em 4 arquivos, que então foram filtrados com a ferramenta strings, que extrai texto legível dos arquivos, com comandos `strings file > file_strings.txt`.

Analisando as saídas finais, foram feitas algumas descobertas relevantes; o arquivo `6DCAB` contém uma listagem de todos os *endpoints* do painel de controle, como `key.json` e `wifi.json`, e funções como `doUpload` e `delete`, o que permite a enumeração de todas as funcionalidades e informações contidas no dispositivo; o arquivo `6D9B1` contém menções de diversos erros, incluindo um erro "*MD5 verification failed*". Isso implica que há um verificação de integridade sobre o arquivo binário, provavelmente durante a instalação a fim de prevenir que seja instalado um *firmware* incompatível no dispositivo. Isso é corroborado pela presença de um arquivo-texto contendo a *hash* MD5 do arquivo juntamente a ele em cada versão no repositório do *Github*. No entanto, o algoritmo MD5 é suscetível a ataques de colisão, o tornando fraco como medida de segurança. Um agente malicioso poderia gerar um arquivo binário malicioso com o mesmo *hash* do arquivo original presente no repositório em tempo viável, a fim de que a sua instalação seja aceita pelo relógio. Assim, acredita-se que o emprego do MD5 para verificação de integridade seja uma medida para evitar mau uso por parte do usuário, mas não para prevenir a ação de um agente malicioso.

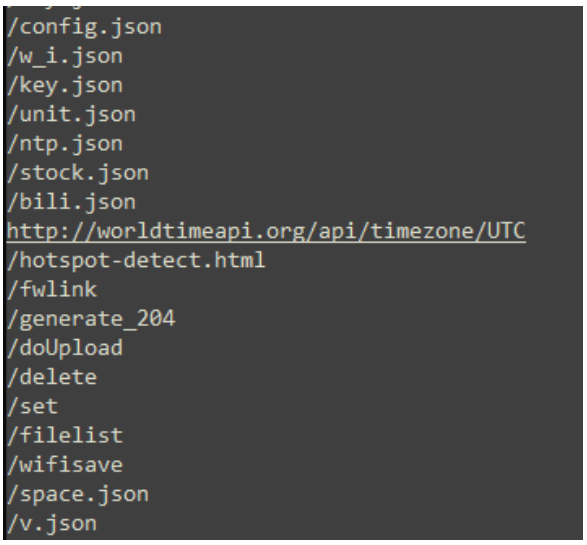


Figura 6.19 – Listagem de *endpoints*

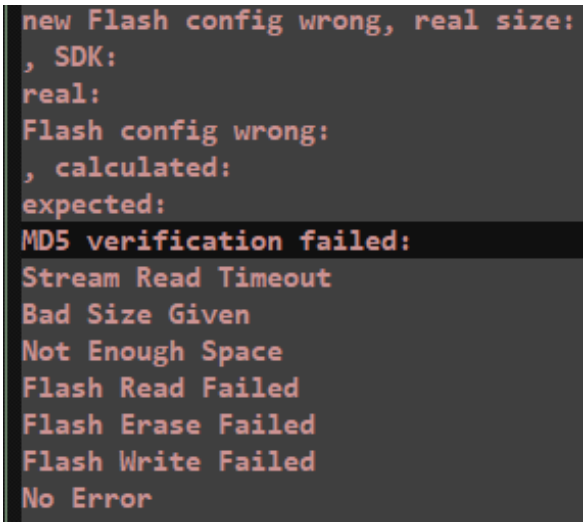


Figura 6.20 – Mensagem de erro de verificação do *hash* MD5

Uma vez que um *firmware* construído por um agente malicioso é instalado no relógio, ele poderia solidificar a sua persistência no ambiente a partir do relógio modificando diversos elementos no painel de configuração, como por exemplo, modificando o *link* para o repositório de atualizações para um endereço controlado por ele. Também poderia ser instalado um *backdoor* para executar comandos remotos a partir do dispositivo, assim, mesmo o atacante não estando diretamente inserido na rede, ele ainda poderia estar conectado remotamente através do relógio e até obter as credenciais da rede Wi-Fi novamente.

### Negação de Serviço

Por razão de completude, também foram executados testes de Negação de Serviço utilizando a ferramenta `hping3`, com os seguintes comandos:

1. ***SYN flood*** - `hping3 -c 15000 -d 120 -S -w 64 --flood 192.168.0.5`
   Envia 15000 pacotes (`-c`) de 120 bytes cada (`-d`), especificando a *flag* SYN (`-S`) com tamanho da janela TCP de 64 (`-w`) e `--flood` para enviar os pacotes o mais rápido possível sem esperar resposta do alvo.

2. ***UDP flood*** - `hping3 --udp --flood -f 192.168.0.5`
   Pacotes udp (`--udp`) com fragmentação (`-f`).

3. ***ICMP flood*** - `hping3 -S --icmp --flood 192.168.0.5`
   Pacotes ICMP (`--icmp`), ping genérico.

4. **Ping da morte** - `hping3 -d 65538 -S -f --flood 192.168.0.5`
   Pacotes de 65538 bytes fragmentados.

O comportamento foi conforme o esperado, com a página web hospedada pelo dispositivo sendo afetada de forma que o acesso às configurações não fosse possível. No entanto, o dispositivo continuou ligado e conectado, sem danos à sua operação normal.

Dessa forma, entende-se que se existe a possibilidade de um agente de ameaças estar presente na rede, não há nenhuma forma de mitigação por parte do usuário para evitar que o seu relógio seja comprometido a não ser o mantendo desligado. É evidente que o fabricante responsável pela manutenção do dispositivo não implementa medidas de segurança e, apesar de não se tratar de um alvo de maior valor como uma câmera de segurança, os riscos associados às funções do relógio podem introduzir uma ameaça persistente e imperceptível ao ambiente em que se insere.

### 6.2.1 Sumário Técnico das Descobertas

Esta seção apresenta um sumário técnico das descobertas que representam comportamentos inesperados para um dispositivo seguro, com níveis de criticidade modelados para o contexto postulado no escopo de testes, bem como a mitigação recomendada para o fabricante do dispositivo.

**GM-01. Ausência de mecanismos de autenticação**

| Criticidade | Crítica |
|---|---|

| | |
|---|---|
| **Descrição** | O painel de configuração do relógio é acessível sem nenhuma autenticação ou senha, permitindo que qualquer um na rede possa alterar as configurações e fazer *upload* de arquivos e *firmware* para o dispositivo. |
| **Informações gerais de probabilidade e impacto** | • **Probabilidade:** Alta, pois o atacante precisa apenas do endereço IP do relógio.<br>• **Impacto:** Alto, pois pode levar ao comprometimento do dispositivo. |
| **Mitigação** | Recomenda-se o uso de um sistema de autenticação para modificar as configurações e uma senha de acesso para o painel de configuração. |

Tabela 6.7: Resumo descritivo da descoberta GM-01.

**GM-02. Algoritmo de *hashing* fraco**

| **Criticidade** | **Alta** |
|---|---|
| **Descrição** | A verificação de integridade do *firmware* é feita através da comparação de *hashes* MD5. Esse algoritmo é considerado fraco por ser particularmente vulnerável a ataques de colisão, possibilitando que seja criado um *firmware* malicioso que gera a mesma *hash* de um *firmware* legítimo, assim contornando essa medida de proteção. |
| **Informações gerais de probabilidade e impacto** | • **Probabilidade:** Média, pois as versões de *firmware* são disponibilizadas em um repositório público, no entanto o vetor de ataque possui maior complexidade.<br>• **Impacto:** Alto, pois um *firmware* malicioso instalado no dispositivo leva a uma gama de outros ataques, como a implantação de um *backdoor* para a rede do usuário. |

| | |
|---|---|
| **Mitigação** | Recomenda-se o uso de um algoritmo de *hashing* forte para a verificação de integridade, como SHA-256. |

Tabela 6.8: Resumo descritivo da descoberta GM-02.

**GM-03. Dados não criptografados armazenados no dispositivo**

| | |
|---|---|
| **Criticidade** | **Baixa** |
| **Descrição** | Os dados armazenados diretamente no dispositivo, como chave de API para serviços de previsão de tempo e senha da rede Wi-Fi, não são criptografados. Um atacante com pode ler esses dados em texto claro ao acessar o painel de configuração. |
| **Informações gerais de probabilidade e impacto** | • **Probabilidade:** Alta, os dados são exibidos diretamente no painel ou obtidos através de *endpoints* dele.<br>• **Impacto:** Baixo, pois apenas usuários avançados irão utilizar a funcionalidade de API e a falta de segurança no acesso ao painel precede esta vulnerabilidade. |
| **Mitigação** | Recomenda-se criptografar dados sensíveis armazenados no dispositivo. |

Tabela 6.9: Resumo descritivo da descoberta GM-03.

# 7. CONCLUSÃO

Os testes obtiveram uma conclusão satisfatória para as proposições deste trabalho, cobrindo o escopo previamente definido e fornecendo um entendimento mais claro do atual cenário de segurança na esfera da Internet das Coisas, tanto em relação ao estado de segurança quanto aos estudos e trabalhos que são realizados nesse âmbito.

Com base nos comportamentos observados e os mecanismos presentes nos dispositivos, contata-se que a segurança ainda é um ponto que necessita da atenção de usuários e desenvolvedores. Enquanto que os usuários devem adotar boas práticas ao adquirir um novo dispositivo, como alterar senhas padrão, ativar configurações de segurança adicionais e atualizar o *firwmare* do dispositivo, que provavelmente estará desatualizado em sua primeira inicialização, cabe aos desenvolvedores e fabricantes prescrever essas boas práticas, padronizar configurações seguras e assumir uma postura proativa em relação a segurança, constantemente identificando possíveis fraquezas e pontos de melhoria nos dispositivos e nos serviços relacionados a eles. Essa é uma tarefa razoavelmente complexa, especialmente quando se trata de múltiplos vetores de ataque identificados para um alvo. Uma estratégia intuitiva é priorizar o tratamento das vulnerabilidades de maior impacto, no entanto, isso não necessariamente irá corresponder linearmente à redução dos riscos. Para isso, a abordagem dos testes de intrusão se faz instrumental ao permitir uma análise holística da superfície de ataque de um determinado alvo, resultado em uma avaliação de risco mais acurada.

Nesse sentido, entende-se que os riscos de segurança relacionados à IoT, em grande parte, são atrelados a uma cadeia de riscos em um grau maior do que em outros paradigmas. A criticidade de uma vulnerabilidade quando avaliada dentro de todo o seu contexto, não isolada, é fortemente influenciada por outros fatores que precedem e sucedem a posição do dispositivo na cadeia em si. Por exemplo, em um cenário prático, a segurança da rede muitas vezes é a primeira camada de proteção contra um agente de ameaça e poderia ser considerada mais relevante do que as vulnerabilidades presentes em um determinado dispositivo inserido na rede, essas vulnerabilidades que, por sua vez, podem não resultar em uma pós-exploração frutífera para o atacante naquele contexto. Com os resultados obtidos nos caminhos de exploração seguidos nos testes, é possível traçar um paralelo com os estudos de segurança realizados em dispositivos; a complexidade de algumas vulnerabilidades conhecidas e reveladas por esses estudos, por mais que publicamente documentadas, exigem maior conhecimento e sofisticação por parte de um agente de ameaças para serem exploradas do que ataques comuns que podem ser executados com relativa facilidade e possuem menos pré-condições. Isso significa que é necessário também estimar a probabilidade de exploração das vulnerabilidades a fim de propriamente avaliar o risco relacionado a ela e, para isso, um teste de intrusão proporciona uma aná-

lise de vulnerabilidade mais concreta ao tentar de fato explorar as vulnerabilidades, ainda mais no contexto de um ambiente doméstico conectado onde há uma cadeia de riscos mais acentuada.

Portanto, conclui-se que a aplicação sistemática de testes de intrusão, tanto por auditores externos quanto internos, pode ser incorporada como prática essencial para avaliar o estado de segurança de dispositivos IoT e idealmente deve constituir da análise detalhada de seus componentes em diversos níveis, além dos serviços e tecnologias adjacentes e implantados pelos dispositivos, se existirem, a fim de assegurar uma avaliação compreensiva dos riscos. Foi evidenciado ao longo deste volume que há uma considerável superfície de ataque nos dispositivos testados, com a presença de vulnerabilidades que envolvem a exploração de código vulnerável, ausência de mecanismos de *rate limit*, configurações padrão inseguras, *bypass* de mecanismos para criptografia, geração de credenciais fracas e até a total ausência de mecanismos de segurança, reforçando a importância da avaliação desses dispositivos que frequentemente são adotados em residências e, portanto, introduzem riscos de segurança e privacidade a seus usuários. Acredita-se, ainda, que somente através de uma abordagem proativa e contínua é possível assegurar que os ambientes domésticos inteligentes, que dependem cada vez mais de dispositivos conectados, permaneçam protegidos contra as crescentes ameaças cibernéticas, com a implementação de políticas robustas de segurança e a conscientização dos usuários sendo passos fundamentais para alcançar um ecossistema de IoT seguro e confiável.

# REFERÊNCIAS BIBLIOGRÁFICAS


[1] Capturado em: https://www.samsung.com/br/smartthings/, 20/11/2024.

[2] Capturado em: https://support.google.com/googlenest/answer/7029281?hl=pt-BR, 20/11/2024.

[3] Capturado em: https://www.amazon.com/alexa-smart-home/b?ie=UTF8&node=21442899011, 20/11/2024.

[4] Capturado em: https://support.apple.com/pt-br/guide/security/sec3a881ccb1/web, 20/11/2024.

[5] Capturado em: https://www.tudocelular.com/android/noticias/n168200/xiaomi-mi-home-v60-atualizacao-novidades.html, 20/11/2024.

[6] Capturado em: https://m-support.ezviz.com/faq/article/What-should-I-do-if-I-have-lost-my-EZVIZ-device-verification-code, 5/10/2024.

[7] Capturado em: https://hub.docker.com/r/aler9/rtsp-simple-server, 5/10/2024.

[8] “Ezviz - security notice 20230726”. Capturado em: https://www.ezviz.com/data-security/security-notice/detail/827?srsltid=AfmBOoof8FKcTBu0mqu8wGrsHaEZX3XkGPGjaLOgtjLT5bFZWEKPBejc, 22/09/2024.

[9] “Hackers take control of robot vacuums in multiple cities, yell racial slurs”. Capturado em: https://www.abc.net.au/news/2024-10-11/robot-vacuum-yells-racial-slurs-at-family-after-being-hacked/104445408, 3/11/2024.

[10] “Ieee std 802.15.1-2005”. Capturado em: https://standards.ieee.org/ieee/802.15.1/3513/, 15/04/2024.

[11] “Ieee std 802.15.4-2024”. Capturado em: https://www.ieee802.org/15/pub/TG4.html, 15/04/2024.

[12] “Introduction to matter”. Capturado em: https://docs.silabs.com/matter/2.1.1/matter-fundamentals-introduction/, 15/04/2024.

[13] “Open Source Security Testing Methodology Manual”. Capturado em: https://www.isecom.org/research.html#content5-9d.

[14] “Open WorldWide Application Security Project”. Capturado em: https://owasp.org/www-project-web-security-testing-guide/.

[15] “Owasp internet of things project”. Capturado em: https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project, 12/04/2024.

[16] “Penetration Testing Execution Standard”. Capturado em: http://www.pentest-standard.org/index.php/Main_Page.

[17] “Tuya developer platform”. Capturado em: https://developer.tuya.com/en/docs/iot, 15/04/2024.

[18] “Tuya smart cloud platform overview”. Capturado em: https://developer.tuya.com/en/docs/iot/tuya-smart-cloud-platform-overview?id=K914joiyhhf7r, 15/04/2024.

[19] “What is zigbee? does it matter for smart homes in 2023?” Capturado em: https://smartgeekhome.com/what-is-zigbee/, 15/04/2024.

[20] “Zigbee vs z-wave: Comprehensive comparison of smart home protocols in 2024”. Capturado em: https://www.smarthomeperfected.com/zigbee-vs-z-wave/, 15/04/2024.

[21] “Asia ccs ’13: Proceedings of the 8th acm sigsac symposium on information, computer and communications security”, 2013.

[22] “Cve-2023-34552”, Relatório Técnico, NIST, 2023.

[23] “Cve-2023-48121”, Relatório Técnico, NIST, 2023.

[24] Avast. In: https://press.avast.com/avast-exposes-internet-of-things-attack-risk-in-barcelona-home-of-mobile-world-congress-2017,
2017.

[25] Avelino Zorzo, D. D. “Overview and open issues on penetration test”, *Journal of the Brazillian Computer Society*, 2017.

[26] Bitdefender. “Vulnerabilities identified in ezviz smart cams”, Relatório Técnico, 2022.

[27] Burhan, M.; Rehman, R. A.; Khan, B.; Kim, B.-S. “Iot elements, layered architectures and security issues: A comprehensive survey”, *Sensors*, vol. 18–9, 2018.

[28] CIS. In: The Mirai Botnet - Threats and Mitigations, 2021.

[29] Cristoffer Leite, João Gondim, P. B. “Pentest on internet of things devices”, *2019 XLV Latin American Computing Conference (CLEI)*, 2019.

[30] De Keersmaeker, F.; Cao, Y.; Ndonda, G. K.; Sadre, R. “A survey of public iot datasets for network security research”, *IEEE Communications Surveys Tutorials*, vol. 25–3, 2023, pp. 1808–1840.

[31] Gang Gan, Zeyong Lu, J. J. “Internet of things security analysis”. In: 2011 International Conference on Internet Technology and Applications, 2011, pp. 1–4.

[32] Ge Chu, A. L. “Penetration testing for internet of things and its automation”. In: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th Intl. Conference on Data Science and Systems, 2018, pp. 1479–1484.

[33] Geeta Yadav, Kolin Paul, A. A. K. O. “Iot-pen: An e2e penetration testing framework for iot”, *Journal of Information Processing*, 2020, pp. 633–642.

[34] Gittins, C. “5ghz vs 2.4ghz: weighing up wi-fi frequency bands”, Relatório Técnico, 2024.

[35] Hameed, A.; Alomary, A. “Security issues in iot: A survey”. In: 2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), 2019, pp. 1–5.

[36] Honkanen, M.; Lappetelainen, A.; Kivekas, K. “Low end extension for bluetooth”. In: Proceedings. 2004 IEEE Radio and Wireless Conference (IEEE Cat. No.04TH8746), 2004, pp. 199–202.

[37] @joerngermany. “Ezviz cs-c6n camera vulnerability (cve-2023-48121)”, Relatório Técnico, 2023.

[38] Karen Rose, Scott Eldridge, L. C. “The internet of things: An overview”, *The Internet Society*, 2015.

[39] Kozlov, D.; Veijalainen, J.; Ali, Y. “Security and privacy threats in iot architectures”. In: Proceedings of the 7th International Conference on Body Area Networks, 2012, pp. 256–262.

[40] Kurdistan Ali, S. A. “Security issues and vulnerability of iot devices”, *International Journal of Science and Business*, 2021, pp. 101–115.

[41] Lunardi, R. C.; Zorzo, A. F.; Michelin, R. A.; Neu, C. V. “Distributed access control on iot ledger-based architecture”. In: NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, 2018, pp. 1–7.

[42] Mario Ballano Barcena, C. W. “Insecurity in the internet of things”, *Security Response*, 2015.

[43] Matthew Denis, Carlos Zena, T. H. “Penetration testing: Concepts, attack methods, and defense strategies”. In: 2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT), 2016, pp. 1–6.

[44] Murzaeva, A.; Kepçeoğlu, B.; Demirci, S. “Survey of network security issues and solutions for the iot”. In: 2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), 2019, pp. 1–6.

[45] Vaidya, V. D.; Vishwakarma, P. “A comparative analysis on smart home system to control, monitor and secure home, based on technologies like gsm, iot, bluetooth and pic microcontroller with zigbee modulation”. In: 2018 International Conference on Smart City and Emerging Technology (ICSCET), 2018, pp. 1–4.

[46] Zafft, A.; Agu, E. “Malicious wifi networks: A first look”. In: 37th Annual IEEE Conference on Local Computer Networks - Workshops, 2012, pp. 1038–1043.