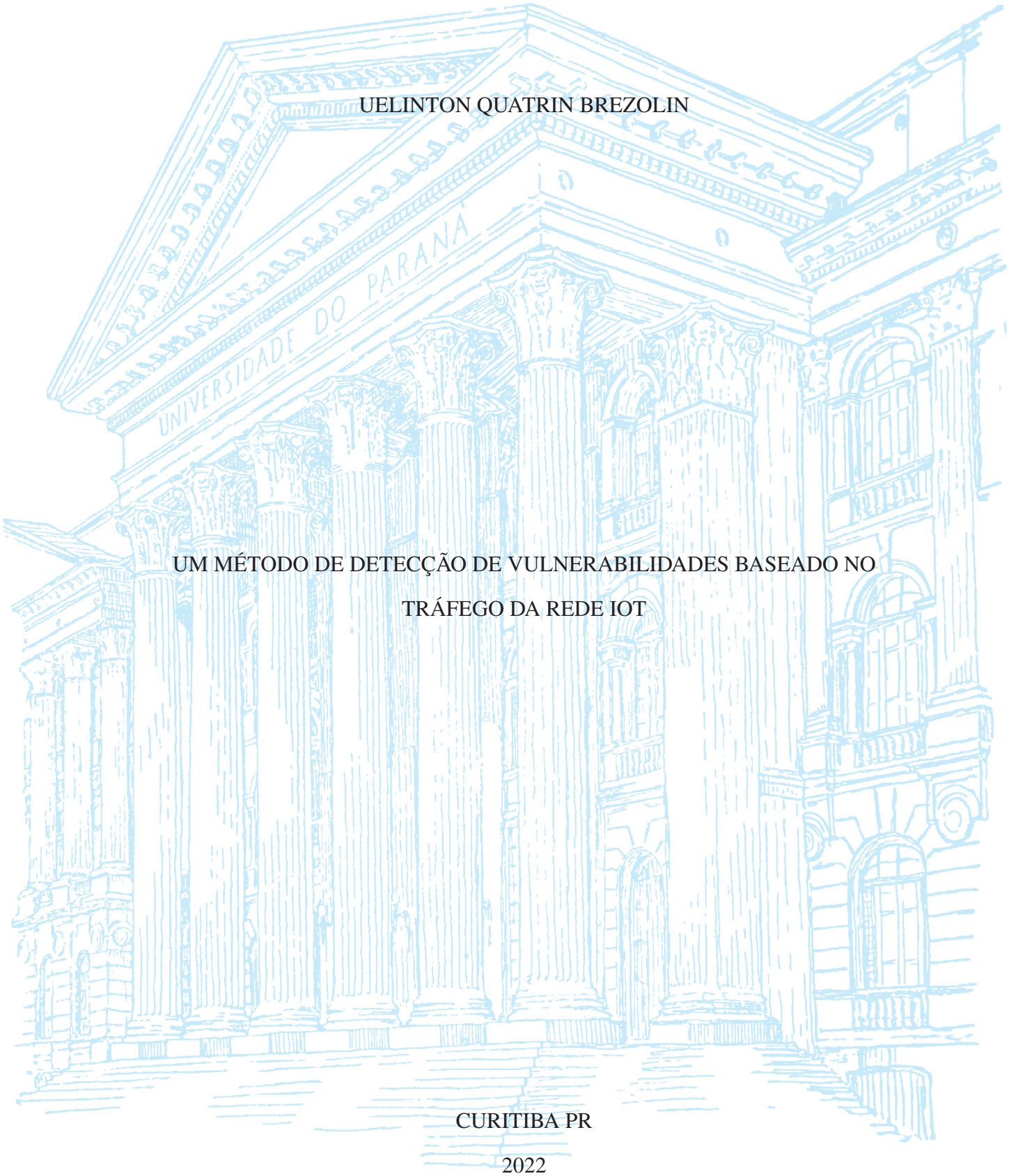
UNIVERSIDADE FEDERAL DO PARANÁ

UELINTON QUATRIN BREZOLIN



UM MÉTODO DE DETECÇÃO DE VULNERABILIDADES BASEADO NO TRÁFEGO DA REDE IOT

CURITIBA PR

2022

UELINTON QUATRIN BREZOLIN

UM MÉTODO DE DETECÇÃO DE VULNERABILIDADES BASEADO NO TRÁFEGO DA REDE IOT

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Orientador: Profa. Dra. Michele Nogueira Lima.

CURITIBA PR

2022

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)
UNIVERSIDADE FEDERAL DO PARANÁ
SISTEMA DE BIBLIOTECAS – BIBLIOTECA CIÊNCIA E TECNOLOGIA

Brezolin, Uelinton Quatrin.
Um método de detecção de vulnerabilidades baseado no tráfego da Rede IoT. / Uelinton Quatrin Brezolin. – Curitiba, 2023.
1 recurso on-line : PDF.

Dissertação (Mestrado) – Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática.
Orientadora: Profa. Dra. Michele Nogueira Lima.

1. Internet das coisas. 2. IoT (*Computer networks*). 3. Segurança, Sistemas eletrônicos de. 4. Privacidade, Direito à. I. Lima, Michele Nogueira. II. Universidade Federal do Paraná. Programa de Pós-Graduação em Informática. III. Título.

Bibliotecário: Nilson Carlos Vieira Júnior CRB-9/1797

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **UELINTON QUATRIN BREZOLIN** intitulada: **Um Método de Detecção de Vulnerabilidades baseado no Tráfego da Rede IoT**, sob orientação do Prof. Dr. MICHELE NOGUEIRA LIMA, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 30 de Setembro de 2022.

Assinatura Eletrônica
10/10/2022 14:47:53.0
MICHELE NOGUEIRA LIMA
Presidente da Banca Examinadora

Assinatura Eletrônica
14/10/2022 09:47:35.0
ALEX BORGES VIEIRA
Avaliador Externo (UNIVERSIDADE FEDERAL DE JUIZ DE FORA)

Assinatura Eletrônica
11/10/2022 09:42:46.0
ALDRI LUIZ DOS SANTOS
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

“Dificuldades preparam pessoas comuns para destinos extraordinários” — C.S Lewis

AGRADECIMENTOS

A minha família Edivan José Brezolin, Solange Beatriz Quatrin Brezolin e Andressa Quatrin Brezolin pelo suporte, apoio e incentivo incondicional. A minha orientadora Dra. Michele Nogueira Lima pelo auxílio, correções, apontamentos e incentivos para a produção e formação acadêmica. A minha banca avaliadora pelas correções e dicas para melhorar o desenvolvimento desta dissertação. Aos meus amigos pela compreensão e paciência em todas as nossas conversas, principalmente Nelson Gonçalves Prates Junior, Andressa Vergütz e Bruno Henrique Schwengber. A universidade e seu corpo docente, que oportunizaram meu aprendizado. E a todos que direta ou indiretamente fizeram parte da minha formação, meu muito obrigado.

RESUMO

A Internet das Coisas (do inglês, *Internet of Things* - IoT) é composta por dispositivos computacionais embarcados em objetos comuns do cotidiano, conectados á Internet. A IoT é um paradigma de rede com dispositivos sem fio, recursos computacionais e energéticos limitados. Os inúmeros benefícios da IoT passam pela coleta, transmissão e processamento de dados sensíveis sobre os usuários, por exemplo, a sua localização geográfica ou as condições fisiológicas. A IoT vem sendo alvo de atacantes que comprometem a segurança e a privacidade dos usuários. Os atacantes exploram vulnerabilidades para efetivar o ataque em si. Estas vulnerabilidades consistem em falhas de segurança, que comprometem a integridade dos dispositivos, como "senhas padrões" e transferência de dados sem criptografia. Na literatura, os trabalhos apresentam técnicas de detecção de vulnerabilidades baseadas em análises de código do *software*, na coleta de dados a partir dos componentes de *hardware* ou do tráfego de rede. Os trabalhos que detectam vulnerabilidades a partir do *hardware* e *software* necessitam de acesso físico aos dispositivos e solucionar os problemas encontrados neste contexto exige alterações de códigos proprietários ou são custosas para os usuários finais, que precisarão trocar os equipamentos vulneráveis por equipamentos modificados mais seguros. Os trabalhos que analisam o tráfego de rede atuam remotamente e as vulnerabilidades encontradas podem ser tratadas sem modificações nos dispositivos. No entanto, as soluções de rede detectam principalmente ataques em execução e não as vulnerabilidades que os levam a serem efetivos. Deste modo, este trabalho apresenta um estudo sobre a detecção de vulnerabilidades de segurança em redes IoT e propõe um método de detecção de vulnerabilidade a fim de auxiliar na prevenção dos ataques. O método proposto detecta vulnerabilidades de segurança por meio do cálculo da entropia e permite a análise da incerteza das informações contidas no tráfego de rede e a identificação dos fluxos provenientes de dispositivos inseguros. Além disso, o método tem como base modelos de aprendizado de máquina rotulados pelo resultado da entropia e usa classificadores para identificar automaticamente os dispositivos vulneráveis recebendo como entrada as características estatísticas extraídas do tráfego. A avaliação de desempenho foi conduzida experimentalmente por meio da análise do tráfego de rede de IoT, considerando um cenário experimental e um cenário orientado a traços. Ambos os cenários foram avaliados *offline* e *stream*. Com base nos conjuntos de dados de cada cenário, é extraído as características de rede, criado as tuplas de fluxo, calculado a entropia para rotulagem dos dados e classificam-se os fluxos. Os resultados mostram a eficiência do método com até 99% de precisão na identificação de dispositivos vulneráveis. Além disso, os resultados apontam ser possível rotular e treinar uma base de dados de acordo com a entropia, e assim criar modelos capazes de identificar dispositivos vulneráveis que não possuem técnicas de criptografia na sua comunicação.

Palavras-chave: Internet das Coisas. Detecção de Vulnerabilidades. Entropia. Análise de Tráfego. Privacidade.

ABSTRACT

The Internet of Things (IoT) is composed of computing devices embedded in common everyday objects, connected to the Internet. IoT is a network paradigm with wireless devices, limited computing, and energy resources. The numerous benefits of IoT include the collection, transmission, and processing of sensitive data about users, for example, their geographic location or physiological conditions. The IoT has been targeted by attackers who compromise the security and privacy of users. Attackers exploit vulnerabilities to carry out the attack itself. These vulnerabilities consist of security flaws that compromise the integrity of devices, such as "default passwords" and unencrypted data transfer. In the literature, works present vulnerability detection techniques based on software code analysis, and data collection from hardware components or network traffic. Work that detects vulnerabilities from hardware and software requires physical access to the devices and solving the problems found in this context requires changes to proprietary codes or is costly for end users, who will need to exchange vulnerable equipment for more secure modified equipment. Jobs that analyze network traffic work remotely and vulnerabilities found can be addressed without modifying the devices. However, network solutions primarily detect running attacks and not the vulnerabilities that make them effective. Thus, this work presents a study on the detection of security vulnerabilities in IoT networks and proposes a vulnerability detection method to help prevent attacks. The proposed method detects security vulnerabilities through entropy calculation and allows the analysis of the uncertainty of the information contained in the network traffic and the identification of flows coming from insecure devices. In addition, the method is based on machine learning models labeled by the entropy result. It uses classifiers to automatically identify vulnerable devices receiving as input the statistical characteristics extracted from the traffic. The performance evaluation was experimentally conducted through the analysis of IoT network traffic, considering an experimental scenario and a trace-oriented scenario. Both scenarios were evaluated offline and streamed. Based on the data sets of each scenario, the network characteristics are extracted, flow tuples are created, entropy is calculated for data labeling and the flows are classified. The results show the efficiency of the method with up to 99% accuracy in identifying vulnerable devices. In addition, the results indicate that it is possible to label and train a database according to entropy, thus creating models capable of identifying vulnerable devices that do not have encryption techniques in their communication.

Keywords: Internet of Things. Vulnerability Detection. Entropy. Traffic Analysis. Privacy.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

6LoWPAN	*IPv6 over Low power Wireless Personal Area Networks* IPv6 sobre Redes de baixa Capacidade Energética e de Área Pessoal
ARPANET	*Advanced Research Projects Agency Network* Rede da Agência para Projetos de Pesquisa Avançada
CoAP	*Constrained Application Protocol* Protocolo de Aplicação Restrita
DDoS	*Distributed Denial of Service* Negação de Serviço Distribuída
DNS	*Domain Name Service* Serviço de Nomes e Domínios
DoS	*Denial of Service* Negação de Serviço
GPS	*Global Positioning System* Sistema de Posicionamento Global
IEEE	*Institute of Electrical and Electronics Engineers* Instituto de Engenheiros Eletricistas e Eletrônicos
IETF	*Internet Engineering Task Force* Força-tarefa de Engenharia para Internet
IoT	*Internet of Things* Internet das Coisas
IPv6	*Internet Protocol version* 6 Protocolo da Internet versão 6
MAC	Media Access Control Controle de Acesso ao Meio
MQTT	*Message Queuing Telemetry Transport* Protocolo de Enfileiramento de Mensagens de Telemetria de Transporte
PHY	*Physical Layer* Camada Física
RFC	*Request for Comments* Pedido de Comentários
RPL	*IPv6 Routing Protocol for Low-Power and Lossy Networks* Protocolo de Roteamento para Redes IPv6 com Baixa Capacidade Energética e com Perdas
TCP	*Transmission Control Protocol*

	Protocolo de Controle de Transmissão
UDP	*User Datagram Protocol*
	protocolo de datagrama de uso
WLAN	*Wireless Local Area Network*
	Rede Local sem Fio
APKs	*Android App Bundles*
	Pacotes de Aplicativos Android
IP	*Internet Protocol Address*
	Endereço de Protocolo da Internet
TLS	*Transport Layer Security*
	Segurança da Camada de Transporte
ID	Identidade
KNN	*K-Nearest Neighbor*
	K Vizinhos mais Próximos
EPROM	*Erasable Programmable Read-only Memory*
	Memória Programável Apagável somente de Leitura
CVEs	*Common Vulnerabilities and Exposures*
	Vulnerabilidades e Exposições Comuns

LISTA DE SÍMBOLOS

$\sum$	Somatório
p	Precisão
r	*Recall*
VP	Verdadeiro Positivo
FP	Falso Positivo
VN	Verdadeiro Negativo
FP	Falso Positivo
sum	Soma
μ	Média
DP	Desvio Padrão

SUMÁRIO

1 INTRODUÇÃO

O avanço tecnológico das redes sem fio e da Internet das Coisas (do inglês, *Internet of Things* - IoT) impulsionam o crescimento no numero de dispositivos conectados à Internet (Xie et al., 2017). Os dispositivos IoT, também conhecidos como "coisas" (*smartwatchs*, termostatos e lâmpadas inteligentes), são interconectados em diversos ambientes, como ambientes hospitalares (Fernandez e Pallis, 2014), militares (Iyer e Patil, 2018) e, principalmente, cidades e casas inteligentes (Verma e Sood, 2018). Por exemplo, a IoT oferece serviços de atendimento de urgência que monitoram a saúde das pessoas diariamente e notificam os transeuntes próximos sobre emergências. Assim, estes dispositivos compreendem sensores vestíveis, sensores de movimento, *smartphones*, equipamentos domésticos, câmeras de monitoramento, entre outros. Estes dispositivos são conectados via Internet a servidores responsáveis pelo processamento dos dados coletados para fornecer informações visando a melhor qualidade de vida do usuário e resultando em uma grande diversidade e quantidade de dispositivos.

A Internet das Coisas (IoT) surgiu como um conceito no início dos anos 2000, e as tendências mostram que essa tecnologia está se consolidando ao passar do tempo. Segundo relatórios, 35, 82 bilhões de dispositivos IoT estão sendo instalados em todo o mundo até final de 2021 e 75, 44 bilhões de dispositivos IoT serão instalados até 2025 (Steward, 2022). Os dispositivos IoT estão em toda parte, de *smartwatches* a assistentes de voz, moldando nosso trabalho, conversas e interações. Devido a esta crescente quantidade de dados gerados pelos dispositivos e novas aplicações inteligentes, muitas vulnerabilidades emergem expondo falhas críticas de segurança como a transmissão do tráfego da rede sem criptografia, o uso de senhas fracas, a falta de mecanismo de atualização seguros, entre outras (Jia et al., 2018). Assim, pessoas maliciosas exploram estas vulnerabilidades no nível de *firmware*, *software*, *hardware* e tráfego de rede tendo em vista o acesso ao dispositivo, conteúdo transmitido, ou ainda no comprometimento da operação da rede.

No Brasil, decorrente da pandemia do novo coronavírus (Covid-19), foi observado um aumento significativo e sem precedentes no tráfego de dados na Internet ocasionado pela quarentena e necessidade de manter o distanciamento social (da Silva et al., 2021). Isso levou a um aumento entre 40% a 50% no uso da Internet de acordo com dados da Agência Nacional de Telecomunicações (ANATEL) (ITC, 2020). Além disso, o número de ataques cibernéticos em 2020 também aumentou. As notificações referentes a ataques cibernéticos contra empresas brasileiras cresceram 220% no primeiro semestre deste ano de 2021 em comparação com o mesmo período de 2020 de acordo com os dados da MZGroup (2021). O último relatório da Nokia (2020) mostra que os ataques cibernéticos a dispositivos IoT aumentaram aceleradamente. No ano de 2020, 33% dos dispositivos infectados eram dispositivos IoT, um aumento de 17% em relação a 2019. Tendo em vista o aumento no número de ataques cibernéticos, na maioria dos casos eles exploram vulnerabilidades em níveis de *firmware*, *software* e tráfego de rede para assim disseminar ataques afetando a vítima.

1.1 MOTIVAÇÃO

No contexto de redes de computadores, observa-se um aumento na taxa de dados trafegados (Souza, 2020). As redes IoT cooperam nesta alta, tendo um aumento gradativo de 15% desde o ano de 2018, podendo chegar ao equivalente de 27% no ano de 2023 de acordo com o relatório anual da Cisco (Cisco, 2020). Entretanto, a segurança computacional desses dispositivos

muitas vezes é deixada de lado devido a limitações de projeto, heterogeneidade, processamento limitado e baixa capacidade energética (Bedhief et al., 2016). Dessa forma, os atacantes exploram as vulnerabilidades de segurança existentes nestes dispositivos e comprometem os principais pilares da segurança de redes e da informação, que garantem a confidencialidade, a integridade e a disponibilidade dos dados trafegados.

Novas estratégias de detecção e identificação de vulnerabilidades se tornam indispensáveis. Para a detecção de vulnerabilidades, existem diversas técnicas, algumas utilizam grafos, outras aprendizagem de máquina e técnicas *fuzzy*, entre outros. Estas técnicas visam detectar vulnerabilidades em níveis de *firmware*, *software* e tráfego de rede. Porém, para se ter acesso ao nível de *firmware* e *software* é necessário acesso ao código fonte. Em contrapartida, para análise no nível de rede basta escutar a rede em modo promíscuo via Wi-Fi. Além disso, existe a detecção de vulnerabilidade por meio de grafos, porém tal técnica consome muita memória e processamento visto que realiza um mapeamento de todo o caminho dos pacotes de rede trafegados entre os dispositivos na rede. Contudo, ressalta-se a importância da detecção baseada na análise de tráfego de rede de forma passiva sem a necessidade de acesso ao dispositivo físico, pois essa análise é interessante por se sobressair em relação as outras visto que não necessita de acesso ao dispositivo físico, não requer acesso ao sistema operacional, muito menos a detalhes de baixo nível do dispositivo. Pelo contrário, basta acesso ao meio sem fio em que os dispositivos estão conectados para capturar todo o tráfego em modo promíscuo.

Devido à exploração destas vulnerabilidades por atacantes, na maioria das vezes estas vulnerabilidades são usadas para disseminar ataques ou até mesmo para espionar o comportamento do usuário baseado na sua atividade do cotidiano. De acordo com (Mukherjee, 2020), uma das principais vulnerabilidades exploradas corresponde ao uso de "senhas padrões", onde os usuários não se preocupam em alterar a senha padrão do dispositivo após ter adquirido o mesmo. Outra vulnerabilidade comumente abordada consiste na transferência de dados pela rede sem o uso de criptografia. No geral, os dispositivos IoT não aplicam criptografia visto que coletam dados rotineiros como monitoramento do ambiente, movimento de pessoas, iluminação do ambiente, entre outras. Em contrapartida, muitos dispositivos capturam dados altamente sensíveis como sinais vitais e rotina do usuário. Todavia, tais dispositivos se abstém de criptografia em virtude da limitação de recursos computacionais como baixa capacidade energética e baixo poder computacional. Portanto, é imprescindível um método de detecção de vulnerabilidades em dispositivos IoT para aperfeiçoar a segurança e a privacidade do usuário final.

1.2 PROBLEMA

Uma vulnerabilidade consiste uma falha de segurança que deixa equipamentos e sistemas expostos a ações maliciosas por parte de *hackers*, ou seja, permite que um invasor reduza a segurança de um sistema (Gu et al., 2020). Por exemplo, a transmissão de pacotes de rede sem o uso de técnicas de criptografia resulta em acesso indevido por terceiros. Dessa forma, as vulnerabilidades comprometem a integridade e a confidencialidade dos sistemas e equipamentos, deixando dados e informações confidenciais de usuários expostos a ações de *hackers*. Para explorar uma vulnerabilidade, um agente malicioso se utiliza de uma ferramenta ou aplica técnicas capazes de se beneficiar da falha no sistema.

Os dispositivos IoT têm chamado atenção de atacantes (Luo et al., 2018). Sua capacidade computacional limitada deixa pouco espaço para segurança integrada, aumentando as ameaças contra a IoT. As restrições computacionais, a heterogeneidade e a baixa capacidade energética dos dispositivos IoT resultam em uma série de vulnerabilidades de segurança de complexa identificação (Bedhief et al., 2016). A Forbes aponta as vulnerabilidades da IoT como uma das

cinco principais ameaças de segurança atual (Forbes, 2021). Tais vulnerabilidades envolvem ausência de criptografia na troca de mensagens, uso de portas inseguras, ausência de protocolos de comunicação de segurança, entre outras. Apesar da criptografia evitar o acesso indevido a dados sensíveis, hoje sua implementação na IoT acaba sendo uma dificuldade devido às restrições computacionais. Assim, os atacantes exploram facilmente essas vulnerabilidades, comprometendo a confidencialidade e a integridade dos dados. A exploração de tais vulnerabilidades resulta na violação da privacidade dos usuários em seus ambientes privados. Por exemplo, a falta de criptografia na troca de mensagens na IoT facilita o acesso dos atacantes a informações sensíveis da rotina dos usuários. Os dispositivos IoT inseguros deixam as informações dos usuários, da infraestrutura e dos serviços da rede expostas a ações maliciosas, sendo assim de extrema importância a detecção das vulnerabilidades.

Na literatura, a detecção convencional de vulnerabilidades ocorre a partir de bases de dados que listam as vulnerabilidades mais comuns (do inglês, *Common Vulnerabilities and Exposures*– CVEs). Porém, essas bases são limitadas a vulnerabilidades conhecidas, o que na maioria das vezes não é o caso na IoT. Existem também mecanismos que tomam como base a análise do tráfego da rede para detectar vulnerabilidades, o que reduz os custos operacionais e simplifica o processo de detecção, pois não requer acesso ao dispositivo físico (Huang et al., 2020). No geral, tais mecanismos empregam técnicas que reconhecem padrões dos dispositivos e da rede por meio de algoritmos de aprendizado de máquina (Sonnekalb, 2019) e técnicas que mapeiam a rede de comunicação por meio de grafos (Fang et al., 2019). Entretanto, os trabalhos que analisam o comportamento da rede, ignoram a detecção de vulnerabilidades críticas como a detecção da ausência de criptografia na comunicação (Wang et al., 2011; Rezaei e Liu, 2019). Além disso, as técnicas baseadas em grafos possuem um custo de memória elevado por mapear toda a rede (Jia et al., 2018). Diante da importância da criptografia na IoT, existem trabalhos que focam em detectar a vulnerabilidade do tráfego não criptografado (Puhan et al., 2014). Porém, tais trabalhos são limitados a esta vulnerabilidade. Portanto, há uma necessidade de novos métodos de detecção de vulnerabilidades na IoT, simples e eficazes, a fim de melhorar a segurança dos usuários.

1.3 OBJETIVO

Esta dissertação tem como objetivo contribuir com os estudos relacionados à segurança de redes IoT por meio do desenvolvimento de um método de detecção de vulnerabilidades baseado na análise de tráfego de rede. Assim o trabalho apresenta o método MANDRAKE (do inglês, *a **M**ethod for vulner**A**bilities detectio**N** base**D** on the IoT netwo**R**k p**A**c**KE**t traffic*). O método visa detectar vulnerabilidades na IoT por meio de uma abordagem simples, eficiente, com baixo processamento e independente do conhecimento prévio da rede. O método consiste em três fases: *(i)* a captura e extração de características do tráfego da rede; *(ii)* a rotulação do tráfego baseado no cálculo da entropia; e *(iii)* a classificação do tráfego através de técnicas de aprendizado de máquina. Seguindo estas fases, o método coleta, filtra e separa o tráfego da rede em fluxos conforme a tupla (IP de origem, IP de destino, porta de origem, porta de destino e protocolo da camada de transporte). A partir dos fluxos, o método extrai as características de rede e calcula a entropia de cada pacote dos fluxos. O valor da entropia serve de entrada para a definição do modelo de classificação e seu treino. Por fim, a saída da classificação aponta quais fluxos de rede possuem criptografia, identificando assim os fluxos vulneráveis com ausência de criptografia na IoT.

1.4 ESTRUTURA DO TEXTO

O restante deste manuscrito está organizado como segue. O Capítulo 2 apresenta os fundamentos para compreensão do funcionamento da IoT, sua relação com a Internet e os serviços fornecidos através dela, além disso o capítulo descreve como as vulnerabilidades são prejudiciais para a segurança. O Capítulo 3 mostra uma revisão bibliográfica dos trabalhos que abordam métodos de detecção de vulnerabilidades. O Capítulo 4 descreve o método de detecção de vulnerabilidades de redes na IoT. O Capítulo 5 detalha as avaliações preliminares sobre o método e discute os resultados. O Capítulo 6 conclui esta dissertação e indica direções futuras.

2 FUNDAMENTOS

Este capítulo apresenta os principais fundamentos sobre a detecção de vulnerabilidades nas redes IoT, necessário para o entendimento desta dissertação. A Seção 2.1 introduz os principais conceitos e terminologias empregados na IoT. A Seção 2.2 apresenta os fundamentos das vulnerabilidades nas redes IoT. A Seção 2.3 descreve como a análise de tráfego é usada e seu risco para a segurança. A Seção 2.4 demonstra os fundamentos da técnica de entropia utilizada nesta dissertação. A Seção 2.6 resume o capítulo.

2.1 INTERNET DAS COISAS

O novo paradigma denominado *Internet of Things* (IoT), introduzido por Kevin Ashton em 1998, atraiu cada vez mais a atenção da academia e da indústria (Bandyopadhyay e Sen, 2011). Esse paradigma inclui objetos comuns, denominados neste texto de dispositivos, equipados com sensores/atuadores e capazes de se interconectar através tecnologias de comunicação entre si e com a Internet. Questões como interoperabilidade entre dispositivos, segurança na transmissão e no processamento dos dados motivam o meio acadêmico a desenvolver soluções para a Internet das Coisas. Um dos desafios para o desenvolvimento de soluções para redes IoT é a limitação dos recursos computacionais dos dispositivos. Assim, as principais soluções para os desafios da IoT são baseadas em tecnologias como a computação em nuvem, pois trata desafios relacionados ao *Big Data* e fornece recursos computacionais pela Internet.

A IoT suporta uma variedade de aplicações sendo apoiada pelo desenvolvimento de tecnologias de rede, protocolos, equipamentos de rede e serviços. Ao contrário de outras redes e sistemas, a IoT não possui um consenso sobre qual arquitetura deve ser seguida. Dessa forma, a IoT segue dois modelos de camadas, sendo o modelo da internet TCP/IP e o modelo de três camadas que compreendem as camadas de percepção, rede e aplicaçãoZhao e Ge (2013). No modelo de três camadas, a camada de percepção compreende dispositivos que interagem com o ambiente, sendo sensores e/ou atuadores que coletam dados e enviam para um determinado dispositivos para ser interpretados, que consiste no rol de dispositivos disponíveis na solução, como sensores, atuadores, câmeras, GPS, celulares, *tablets* entre outros (Said e Masud, 2013). A camada de rede efetua as comunicações de dispositivo-a-dispositivo, para encaminhar os dados coletadas pela camada anterior (camada de percepção). A camada de aplicação utiliza as informações tratadas e encaminhadas pelas camadas de recepção e de rede. Esta organização em particular apoia o desempenho de atividades voltadas para as demandas de diferentes contextos, como casas de inteligentes, cidades inteligentes, entre outros.

Devido às características dos dispositivos IoT que consistem em portabilidade, escassez de recursos e redes suscetíveis a perda de pacotes, a implementação de protocolos padronizados da Internet torna-se inviável, pois são esses protocolos são projetados para dispositivos mais com maior capacidade computacional e conexões confiáveis. Diante disso, a *Internet Engineering Task Force* (IETF) padronizou uma série de protocolos com o objetivo de interconectar as redes por meio da Internet e fornecer serviços para tais dispositivos com recursos limitados. Com base nisso, é demonstrado os protocolos da pilha de TCP/IP de acordo com a organização do modelo de cinco camadas, como segue aplicação, transporte, rede, enlace e física que a IoT implementa.

Os principais protocolos da camada de aplicação da pilha de TCP/IP que a IoT implementa, consistem no Protocolo de Enfileiramento de Mensagens de Telemetria de Transporte (do inglês, *Message Queuing Telemetry Transport* - MQTT) (Banks e Gupta, 2014) e o Protocolo

de Aplicações Restritas (do inglês, *Constrained Application Protocol* - CoAP) (Shelby et al., 2014). O MQTT é dito como um protocolo de mensagens leves para sensores e pequenos dispositivos móveis. O CoAP é um protocolo que se destina à conexão de dispositivos com recursos limitados na Internet, como os nós das redes de sensores sem fio, definido pela RFC 7252 (Shelby et al., 2014). A camada de transporte define como os dispositivos vão se comunicar, ou seja, estabelecendo uma forma padrão para a troca de mensagens numa comunicação fim-a-fim. Dentre isso, existem dois modelos para esta comunicação assim como a Internet, o modelo orientado à conexão e o modelo não orientado à conexão. Os protocolos equivalentes a esses dois modelos são o Protocolo de Controle de Transmissão (do inglês, *Transmition Control Protocol* - (TCP)) e o Protocolo de Datagrama do Usuário (do inglês, *User Datagram Protocol* - (UDP)). O TCP consiste em aplicações que necessitam de maior confiabilidade na comunicação, onde o TCP garante a entrega dos dados transferidos entre origem e destino. O UDP não oferece a garantia de entrega, utilizado para aplicações que não possuem restrições em relação à confiabilidade.

Na camada de rede, devido ao problema a respeito do limite de endereços do protocolo da Internet naa versão 4 do protocolo IP, adota-se uma solução que implementa o IPv6 para redes sem fio e com baixo consumo de energia, o protocolo 6LoWPAN. O protocolo 6LoWPAN padronizado pela IETF (RFC 4919) (Montenegro et al., 2007), implementa o IPv6 para redes sem fio com baixo consumo energético, assim suprindo o problema clássico sobre o limite de endereços da versão 4 do protocolo da Internet (IP). O 6LoWPAN permite que estruturas de rede de curta distância e largura de banda baixa se comuniquem com dispositivos na Internet por meio do protocolo IPv6. O IPV6 comprime os cabeçalhos IPv6. Para as camadas de enlace e física, os protocolos IEEE 802.15.4 MAC e PHY (802.15.4, 2016) são implementados, respectivamente. O IEEE 802.15.4 MAC efetua o controle de acesso para redes sem fio de baixas taxas de transmissão. Já o IEEE 802.15.4 PHY controla a transmissão e recepção de bits no meio físico de acordo com a disponibilidade indicada pela camada MAC. Contudo, o protocolo IEEE 802.11 WLAN frequentemente chamado de *Wi-Fi*, certificado pela *Wi-Fi Alliance*[1] é usado na maioria das redes domésticas e de escritório para permitir *laptops*, impressoras, *smartphones* entre outros dispositivos, para se comunicarem entre si e acessarem a Internet sem a necessidade de cabeamento. Porém, possui alta taxas de transmissão de dados em dispositivos IoT que possuem baixo poder de processamento.

2.2 VULNERABILIDADES

Uma vulnerabilidade de rede consiste numa falha de segurança que expõe sistemas e equipamentos a ações maliciosas por *hackers*. Uma falha de segurança é qualquer incidente que resulte em acesso não autorizado a dados, aplicativos, redes ou dispositivos, o resultado é que as informações são acessadas sem autorização. Assim, esses tipos de vulnerabilidades são ocasionadas por erros de projeto ou na implementação e configuração de redes, *softwares* e *hardware*. Os sistemas de detecção de vulnerabilidades analisam os componentes (tanto de *hardware* quanto de *software*) da infraestrutura de rede para evitar cenários que coloquem a segurança dos dados e dos usuários em risco. No entanto, muitos dispositivos e softwares são colocados no mercado sem o devido cuidado em relação à segurança diante da escassez de legislação relacionada, contribuindo para que os fabricantes ignorem as considerações de segurança e projetem dispositivos, aplicações e serviços IoT potencialmente vulneráveis. Além disso, os sistemas de detecção de vulnerabilidades na IoT possuem várias dificuldades técnicas, incluindo armazenamento limitado, energia e recursos computacionais (Neshenko et al., 2019). Os sistemas de detecção de vulnerabilidades realizam atividades de monitoramento e precisam

[1] *Wi-Fi Alliance*:`https://www.wi-fi.org/`

estar constantemente atualizados, pois apenas um dispositivo vulnerável pode revelar informações de segurança sobre uma infraestrutura inteira.

Uma vulnerabilidade é uma falha intrínseca no *software* ou *hardware* de um dispositivo que pode ser explorada para quebrar as barreiras impostas pelos mecanismos de segurança dos alvos e violar a confidencialidade, a integridade e/ou a disponibilidade das informações trocadas em rede. De acordo com *Common Vulnerabilities and Exposures* define vulnerabilidade como uma fraqueza na lógica computacional (por exemplo, código) encontrada em componentes de *software* e *hardware* que, quando explorados, resulta em um impacto negativo na confidencialidade, integridade ou disponibilidade (Vulnerabilities e Exposures, 2022). A mitigação das vulnerabilidades neste contexto normalmente envolve alterações de codificação, também inclui alterações de especificação ou até mesmo descontinuidades de especificação (por exemplo, remoção de protocolos afetados ou funcionalidade em sua totalidade). Dessa forma, as vulnerabilidades que permitem o acesso não autorizado aos recursos e dados são problemas de confidencialidade. Os problemas de integridade consistem em ocultar vulnerabilidades que possibilitam modificar de forma não autorizada os dados protegidos e as configurações da infraestrutura de rede. As vulnerabilidades que impedem ou dificultam o acesso de usuários autorizados resultam em problemas de disponibilidade.

As vulnerabilidades estão contidas em todas as camadas da IoT. A camada de percepção possui vulnerabilidades associadas aos elementos de *hardware* da IoT, como os dados relacionados ao consumo de energia, que podem ser usados para violar a confidencialidade da chave secreta empregada pelos mecanismos de criptografia Gendreau e Moorman (2016). As vulnerabilidades da camada de aplicação se baseiam no *firmware* local e/ou *software* de controle remoto dos dispositivos IoT, que por mau funcionamento dos mecanismos de autenticação permitem violar a integridade da infraestrutura. Em contraste, as vulnerabilidades contidas na camada rede operam principalmente nos problemas originados pelos protocolos de comunicação que definem o tráfego de rede destes dispositivos. Uma visualização de vulnerabilidades por camadas é encontra na Tabela 2.1.

Camadas TCP/IP	**Vulnerabilidades IoT**
Aplicação	Senhas Fracas Serviços inseguros
Transporte	*Tráfego decriptografado* Transferência e armazenamento de dados inseguros
Rede	Configurações padrão inseguras
Física	Falta de mecanismos seguros de atualização Uso de componentes inseguros ou desatualizados

Tabela 2.1: Vulnerabilidades por Camadas

Os atacantes exploram estas vulnerabilidades contidas nos dispositivos conectados à Internet com pouco ou nenhum esforço para roubar informações, causar prejuízos através de ataques de negação de serviço, entre outros. Dada a implantação de dispositivos IoT, esses efeitos maliciosos têm um impacto profundo na segurança e na resiliência de toda a Internet. Entre os muitos casos que recentemente chamaram a atenção do público, o ataque cibernético lançado pelo *malware* Mirai específico de IoT fornece um exemplo claro da gravidade da ameaça

causada à IoT, pois ele é capaz de analisar vulnerabilidades e assumir o controle de dispositivos conectados à Internet automaticamente.

2.3 ANÁLISE DE TRÁFEGO

A análise de tráfego se trata de uma técnica de engenharia de rede que consiste em examinar as características estatísticas de pacotes de fluxo (por exemplo, tamanhos de pacotes, tempos entre chegadas e direções de pacotes). Os administradores de rede aproveitam as técnicas de análise de tráfego para detectar intrusões, anomalias e para diferenciar fluxos de tráfego malicioso. Além disso, uma implantação adequada de análise de tráfego fornece informações valiosas para controle de tráfego, verificação de diagnóstico e gerenciamento de recursos. Os engenheiros podem usar essas informações para construir redes robustas e evitar possíveis atrasos. Para esse fim, a análise de tráfego é usada para dar suporte a serviços baseados na Internet, incluindo bancos, saúde, governo, sistemas elétricos e transporte (Chaddad et al., 2021).

A análise de tráfego é o processo de captura, farejamento e análise de mensagens em uma rede (Staudemeyer et al., 2005). Porém, a análise de tráfego pode ser mal utilizada para lançar ataques para inferir conhecimento sobre o tráfego da rede por meio da exploração do vazamento de informações do canal lateral. Mesmo com o uso crescente de criptografia para proteger o conteúdo do tráfego, a análise de tráfego está se tornando um ataque comum, ameaçando a privacidade, o anonimato e a confidencialidade. Esses tipos de ataques podem causar grandes violações de privacidade em sistemas militares, bancos, saúde, etc (Soltani et al., 2017).

Neste contexto, os pacotes de rede são capturados por ferramentas de coleta, no qual cada pacote possui informações. A Figura 2.1 mostra os campos de cabeçalho contidos neles considerando as camadas de rede e transporte da pilha TCP/IP. No cabeçalho IP, na maioria dos casos, o pacote é encapsulado pelo IP de origem/destino e protocolo. A partir do cabeçalho IP, são usados o IP de origem/destino e o campo do protocolo, junto com a porta de origem e destino, assim cria-se a clássica 5-tupla para geração de fluxos, levando em consideração tanto, pacotes que utilizam o protocolo de transporte TCP *(Transmission Control Protocol)* ou UPD *(User Datagram Protocol)*.

2.4 ENTROPIA

Na teoria da informação, a entropia se trata do nível médio de "incerteza" inerente aos resultados possíveis de uma variável de entrada, exemplo: variável de entrada pode ser dita como o *payload* de um pacote, assim a entropia calcula nível médio de "incerteza" desta variável. O conceito de entropia da informação ou entropia de Shannon foi criada por Shannon (1948) em seu artigo "*A Mathematical Theory of Communication*". Conceitualmente, as informações de uma determinada aplicação podem ser consideradas armazenadas ou transmitidas como variáveis. Neste caso, uma variável pode ser entendida como uma unidade de armazenamento que pode assumir, em momentos diferentes, um entre vários valores diferentes, seguindo algum processo para assumir esses valores (Vajapeyam, 2014). Dessa forma, a entropia quantifica a incerteza envolvida no valor de uma variável aleatória.

De acordo com McClean (2003), o conceito mais importante na teoria da informação é a Entropia de Shannon, que mede a quantidade de informação contida nos dados. A entropia quantifica até que ponto os dados estão espalhados por seus valores possíveis. Assim, alta entropia significa que os dados estão espalhados o máximo possível, enquanto baixa entropia significa que os dados estão quase todos concentrados em um valor. A Teoria da Informação também tem sido usada como uma medida de interesse que nos permite levar em conta a frequência com que uma

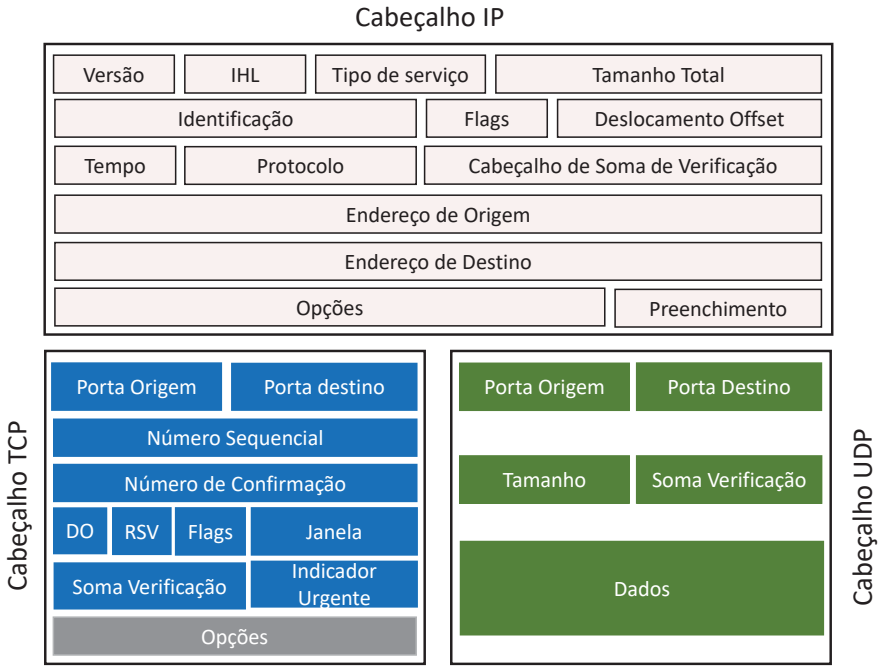


Figura 2.1: Cabeçalhos Pacote

regra ocorre e o quão bem-sucedida ela é, esta regra representa um exemplo de probabilidade de ocorrer uma ordem e/ou determinação em algum evento e/ou jogo. A entropia possui inúmeras interpretações, pois é usada em diferentes campos de pesquisa. Porém, neste caso a entropia é usada para estimar o índice de aleatoriedade contida em cada pacote. Dentre isso, a variável aleatória dada como entrada se trata do pacote em si. Os pacotes são compostos por *bytes*. Então, cada *byte* que compõem um pacote representa um símbolo, originando um conjunto de símbolos a cada pacote analisado. Dessa forma, para cada símbolo I de um conjunto de símbolos W com comprimento N é possível contar as ocorrências através de N_I e suas frequências $P_I = \frac{N_I}{N}$. Ou seja, estimando a incerteza para W, sendo o conjunto total de *bytes* do pacote.

Um exemplo pode ser visto na Figura 2.2. Assume-se aqui que a definição de entropia é uma definição introduzida na teoria da informação, que descreve a entropia como uma medida do grau de indecisão para uma determinada variável aleatória. Nesse caso, seu valor é considerado como uma possível expressão do nível de heterogeneidade da variável analisada. É importante lembrar que o valor da entropia aumentará conforme o domínio de aparência da variável (aplicando as estatísticas acima) se torna mais disperso. A entropia é usada para medir seu nível e incorporá-la como um valor que pode ser usado para caracterizar e comparar diferentes amostras.

A entropia foi calculada baseado na teoria de Shannon (1948) por sua vez descreve a entropia como uma medida do grau de incerteza de uma determinada variável aleatória. Assim, seu valor é visto como uma possível formulação do nível de heterogeneidade da variável em análise. No caso desta dissertação, a variável aleatória se trata do *payload* de cada pacote extraído. Em redes de computadores, o *payload* caracteriza os dados a serem transmitidos, ou seja, o conteúdo do pacote (Braden, 1989). Portanto, o método funciona aplicando o cálculo da entropia em cada conteúdo dos pacotes extraído de cada fluxo gerado de acordo com a Subseção 4.2.1, resultando numa nota entre 0 e 1.

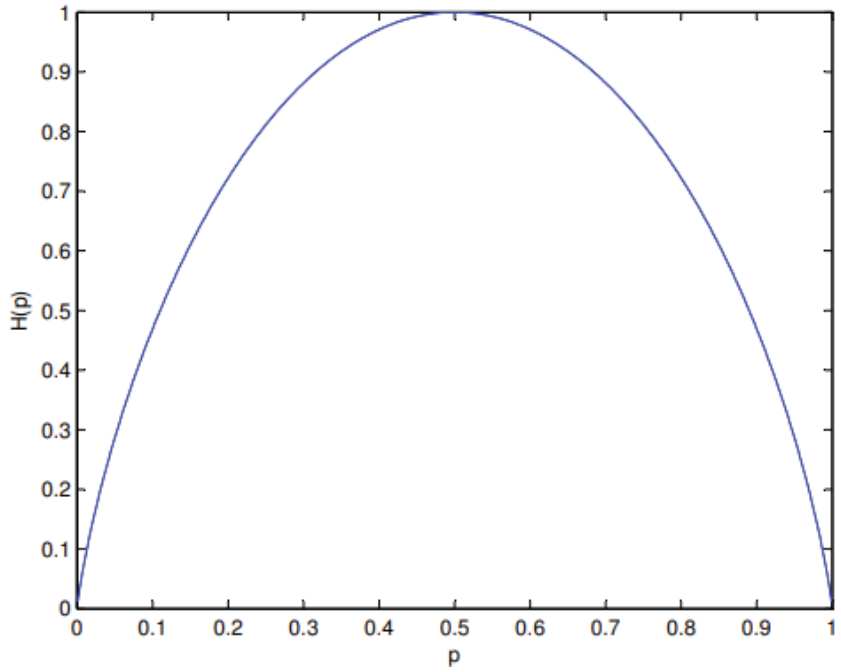


Figura 2.2: Representação Entropia

2.5 MODELO DO ADVERSÁRIO

Há diversos tipos de adversário e ameças estudados nas tecnologias IoT atuais. De acordo com Shirey (2007) uma ameaça é caracterizada como um potencial de violação de segurança, que existe quando há uma entidade, circunstância, capacidade, ação ou evento que pode causar danos. Ross et al. (2018) conceituam ameaça como um evento ou condição que tem o potencial de causar perda de ativos e as consequências indesejáveis ou impacto de tal perda. Por sua vez, adversário também chamado de oponente ou inimigo, um adversário se trata de uma entidade maliciosa cujo objetivo é impedir que os usuários atinjam seu objetivo. De acordo com Edgar e Manz (2017), um adversário cibernético se trata de alguém ou um grupo que pretende realizar ações maliciosas contra outros recursos cibernéticos.

O modelo do adversário tratado neste trabalho é constituído pela perspectiva do atacante explorar uma vulnerabilidade da rede para analisá-la e se beneficiar da mesma, executando um ataque para prejudicar um serviço da rede. Do ponto de vista do atacante uma vulnerabilidade pode ser uma senha padrão de um roteador convencional, no qual é possível acessá-lo de forma simples e explorar o uso da Internet dos usuários que utilizam a rede doméstica. Além disso, outra vulnerabilidade bastante explorada pelos ataques se refere ao tráfego sem criptografia devido ao alto índice prepucial para a segurança na rede, pois pode se adicionar, remover e/ou adulterar informações contidas nos pacotes que são trafegados pela Internet, tendo o poder de afetar até o provedor de serviço.

Uma vulnerabilidade é uma falha intrínseca no *software* ou *hardware* de um dispositivo que pode ser explorada para quebrar as barreiras impostas pelos mecanismos de segurança dos alvos e violar a confidencialidade, a integridade e/ou a disponibilidade das informações trocadas em rede. As vulnerabilidades que permitem o acesso não autorizado aos recursos e dados são problemas de confidencialidade. Os problemas de integridade consistem em ocultar vulnerabilidades que possibilitam modificar de forma não autorizada os dados protegidos e as configurações da infraestrutura de rede. As vulnerabilidades que impedem ou dificultam o acesso de usuários autorizados são problemas de disponibilidade.

Uma vez que nas redes sem fio os dispositivos IoT transmitem dados pessoais em forma de pacotes padronizados por protocolos populares, não é possível saber quem está recebendo ou "farejando" (*sniffing*/ouvindo sem autorização) o tráfego da rede. Neste contexto, um dispositivo vulnerável que não emprega corretamente as técnicas de criptografia serve como entrada para usuários mal-intencionados realizar vetores de ataques, controlar os dispositivos remotamente, roubar os dados pessoais e/ou causar prejuízos financeiros.

2.6 RESUMO

Este capítulo apresentou os conceitos necessários para o entendimento do restante deste manuscrito. Dessa forma, o capítulo apresentou os conceitos relacionados ao funcionamento da Internet das coisas, assim como também sua arquitetura, principais protocolos padronizados e a comunicação empregada pelos dispositivos. O capítulo também apresentou como as vulnerabilidades em redes IoT são prejudiciais para a segurança do usuário. Mostrou a análise de tráfego como um método eficiente para examinar a rede em questão. Além disso, conceituou-se a entropia usada para medir a aleatoriedade de uma informação e por fim mostrou o modelo de adversário detectado nessa dissertação.

3 TRABALHOS RELACIONADOS

Na literatura, diferentes trabalhos detectam vulnerabilidades de segurança em dispositivos conectados à Internet. Essas vulnerabilidades estão relacionadas a vários aspectos, desde o código-fonte do dispositivo até o canal de comunicação entre o dispositivo e seu provedor de serviço. Uma vulnerabilidade encontrada por um agente malicioso pode ser prejudicial para uma rede de comunicação, pois se torna possível explorar e usar essa vulnerabilidade para disparar ataques. Além disso, diferentes trabalhos utilizam técnicas de classificação e rotulagem de dados de tráfego para análise e detecção de vulnerabilidades contidas na rede. Este capítulo tem o objetivo de apresentar os principais tipos de detecção de vulnerabilidade abordados na literatura. Dessa forma, este capítulo está organizado como segue. A Seção 3.1 apresenta as diferentes abordagens de detecção de vulnerabilidades, em conjunto a Subseção 3.1 descreve os níveis de detecção abordados pela literatura. A Seção 3.2 demonstra os tipos de classificação de tráfego e mostra formas de rotulagem de tráfego. A seção 3.3 discute as lições aprendidas dos trabalhos da literatura. E, por fim, a Seção 3.4 resume o capítulo.

3.1 ABORDAGENS DE DETECÇÃO DE VULNERABILIDADES

Existem diversos estudos na literatura que abordam a detecção de vulnerabilidades de segurança (Yi et al., 2019; Jia et al., 2017; McDaid et al., 2021). No geral, esses trabalhos seguem técnicas e abordagens específicas. A Tabela 3.1 apresenta uma classificação das abordagens de detecção de vulnerabilidades que seguem dois principais tipos de análises: passiva e ativa (Standaert, 2010). Uma análise de vulnerabilidade ativa envia transmissões de "tráfego de teste" para os nós ou terminais na rede corporativa, em seguida, examina as respostas recebidas desses nós para avaliar qual nó representa um ponto fraco. Já uma análise passiva de vulnerabilidade observa o fluxo de tráfego da rede para coletar informações sobre seus sistemas e terminais. Ao contrário das análises ativas, análise passiva não interage diretamente com esses sistemas enviando uma solicitação de sondagem ou solicitando uma resposta de sondagem. Essas informações podem ser capturadas usando ferramentas de análise de tráfego. Ambas as análises, ativa e passiva, seguem diferentes abordagens e níveis. As abordagens aplicadas na detecção de vulnerabilidade, geralmente, utilizam técnicas baseadas em grafos mapeando a rede de comunicação para auxiliar na detecção (Jia et al., 2018; Fang et al., 2019), ou técnicas de aprendizagem de máquina para o reconhecimento de padrões para a detecção (Sonnekalb, 2019; Grieco e Dinaburg, 2018; Chernis e Verma, 2018).

Jia et al. (2018) apresentaram um mecanismo baseado em grafos para identificar vulnerabilidades na comunicação de dispositivos IoT. Este mecanismo constrói um grafo a partir dos dados de tráfego IoT, sendo dividido em subgrafos isolados, correlacionado com as ações dos dispositivos (ligar/desligar uma lâmpada). Dessa forma, foi quantificado o nível de sensibilidade de cada subgrafo correlacionado com base nos pares de valor-atributo transportados pelos tráfegos correspondentes, identificando qual mais vulnerável. Todavia, o mecanismo obtém informações da carga útil do pacote. Fang et al. (2019) demonstraram uma estrutura formal de camada cruzada para revelar de forma abrangente as vulnerabilidades em dispositivos IoT, chamado ForeSee, essa estrutura gera um grafo que representa todos os componentes essenciais da IoT, desde ambientes físicos de baixo nível, até processos de tomada de decisão de alto nível. Esta estrutura por sua vez permite a captura com precisão dos caminhos correspondentes de ataques em potencial. Todavia, a estrutura possui um custo de memória excessivo por mapear o caminho do ataque,

Trabalhos	Abordagem		Níveis			Análise	
	ML	*Grafos*	*Software*	*Firmware*	*Rede*	*Ativa*	*Passiva*
Sachidananda et al. (2020)				✓			✓
He et al. (2020)				✓		✓	
Oser et al. (2022)				✓		✓	
Medeiros et al. (2016)	✓			✓		✓	
Sonnekalb (2019)	✓			✓			✓
Harer et al. (2018)	✓		✓			✓	
Zhang (2020)			✓			✓	
Chernis e Verma (2018)	✓		✓			✓	
Bhatia et al. (2019)	✓			✓		✓	
Jia et al. (2017)		✓			✓	✓	
Yi et al. (2019)		✓			✓		✓
Fang et al. (2019)		✓			✓	✓	
Jia et al. (2017)		✓			✓	✓	
Koike et al. (2021)	✓				✓		✓
Rose et al. (2021)					✓	✓	
Husnain et al. (2022)					✓		✓

Tabela 3.1: Classificação trabalhos Literatura

pois o grafo é criado para todas as comunicações da rede sendo assim os componentes do grafo são armazenados em memória.

Sonnekalb (2019) propuseram um processo de aprendizado de máquina supervisionado que extrai padrões de trechos de código vulneráveis e os identifica em código-fonte invisível. Como se trata de um método supervisionado, foi usada uma lista de fraquezas comum de *software* e *hardware* (CWE) para auxiliar no aprendizado de máquina. Grieco e Dinaburg (2018) apresentaram o *Central Exploit Organizer* (CEO), uma ferramenta de prova de conceito para otimizar a seleção de ferramentas de detecção de vulnerabilidade (VDT). O CEO usa o aprendizado de máquina para otimizar a seleção e configuração da ferramenta de detecção de vulnerabilidade mais adequada. Chernis e Verma (2018) demonstrou como é possível detectar uma grande porcentagem de bugs extraindo recursos de texto de funções no código-fonte C e analisando-os com um classificador de aprendizado de máquina. Entretanto, ambos os autores apresentam abordagem baseada em aprendizado de máquina, mas deixam de lado a análise de tráfego. Bhatia et al. (2019) apresentou uma abordagem baseada em aprendizado de máquina não supervisionado para detecção de anomalias centradas em redes IoT. Mostrou que classificadores não supervisionados baseados em *Autoencoder*, quando treinados em dados de tráfego benigno, são eficazes na modelagem do comportamento da rede e na detecção de anomalias e ataques em redes industriais. Porém, a abordagem foca somente em tráfego TCP, além disso, a detecção ocorre após o ataque estar em execução.

3.1.0 Níveis de Detecção de Vulnerabilidades

Existem estudos que detectam vulnerabilidade seguindo três níveis de detecção, sendo eles *firmware*, *software* e análise de rede. A Tabela 3.1 apresenta os diferentes níveis em que a vulnerabilidade está sendo detectada baseado na literatura. Como segue, nível *firmware* sendo um conjunto de instruções operacionais que são programadas diretamente no *hardware* dos

dispositivos (EPRON) (Sachidananda et al., 2020; He et al., 2020), nível de rede se tratando do processo de interceptar e examinar dados trafegados pela rede de comunicação (Yi et al., 2019; Jia et al., 2017) e por fim o nível de *software* refere-se a uma sequência de instruções a serem seguidas e/ou executadas, na manipulação, redirecionamento ou modificação do *hardware* do dispositivo (Lin et al., 2020; Zhang, 2020).

Sachidananda et al. (2020) propuseram uma estrutura de análise estática para detectar vulnerabilidade de segurança no nível de *firmware* do dispositivo IoT. A estrutura visa identificar vulnerabilidades em *firmwares*, APKs (Pacotes de Aplicativos *Android*) e software livre de dispositivos. He et al. (2020) projetaram uma plataforma híbrida para detectar vulnerabilidades no *firmware* IoT, que integra detecção estática offline e detecção dinâmica online. A plataforma identifica com eficácia várias vulnerabilidades e riscos de segurança no *firmware*, como processos perigosos, vulnerabilidades exploráveis e outras superfícies de ataque. Medeiros et al. (2016) apresentam a DEKANT, uma ferramenta de análise estática que aprende a detectar vulnerabilidades em aplicações web. A ferramenta usa um modelo de sequência para aprender a caracterizar vulnerabilidades com base num conjunto de "fatias" de código-fonte. Este modelo leva em consideração a ordem em que os elementos do código aparecem e são executados nas fatias. Oser et al. (2022) mostraram o SAFER, uma estrutura de avaliação de segurança para riscos de dispositivos incorporados, que permite uma avaliação de risco semi automatizada de dispositivos IoT em qualquer rede. SAFER combina informações de identificação de dispositivos de rede e análise automatizada de *firmware* para estimar o risco atual associado ao dispositivo. Entretanto, não foi considerado o tráfego de rede para nenhuma das análises apresentadas.

Zhang (2020) propõe um modelo de rede neural para classificação de tráfego baseado em dois níveis, pacotes de dados e fluxos de rede, para detecção de vulnerabilidades de atualização de *software*. Para avaliar o modelo foi construído um conjunto de dados de atualização de *software* vulnerável e conduzido experimentos no conjunto de dados CICIDS 2017. Harer et al. (2018) apresentaram uma técnica de aprendizado de máquina para detecção automatizada de vulnerabilidades em softwares C e C++. Para isso, foi compilado um conjunto de dados contendo funções de código aberto rotulado com as saídas de um analisador estático. Em seguida, comparam-se os métodos aplicados diretamente ao código-fonte com os métodos aplicados aos itens extraídos do processo de construção. Entretanto, as abordagens apresentam detecção de vulnerabilidades no nível de software, mas se comportam de forma invasiva para o usuário final.

Yi et al. (2019) apresentaram um método, onde é usada a tecnologia de grafo de ataque para estabelecer um modelo de avaliação de segurança de rede baseado na análise de associação de vulnerabilidade no ambiente IoT, e um plano de avaliação quantitativa é fornecido para o status de segurança da IoT. Os autores analisam o princípio de formação da detecção de vulnerabilidade de alerta precoce de comunicação inteligente sob a IoT e propõem um modelo de propagação de poluição dinâmica baseada na exploração de vulnerabilidade de alerta precoce de comunicação inteligente sob a IoT. Jia et al. (2017) apresentaram um sistema de detecção semiautomática de vulnerabilidades no cenário de casas inteligentes antes dos dispositivos IoT usados serem vendidos. Para isso, foi criado um modelo de ameaça em camadas pré-estabelecidas para orientar onde o invasor fará os ataques, dessa forma, foram analisados os riscos em cada camada e projetado o modelo de ameaça para guiar na construção de um sistema de detecção. Portanto, o sistema apresentado se comporta de forma invasiva, usando arquivos de configurações dos aplicativos dos dispositivos para detecção.

Rose et al. (2021) introduziu uma estrutura de detecção de intrusão baseada em perfis de tráfego de rede e aprendizado de máquina. O sistema proposto envolve dois componentes: o serviço de perfil utilizado para realizar o perfil da rede e a avaliação/identificação de vulnerabilidades de cenários localizados na rede local. O segundo componente é o sistema IDS,

uma integração de técnicas de visualização binária usadas para detecção de malware e detecção baseada em assinatura de rede da Suricata. No entanto, o artigo apresentou uma forma de detectar anomalias com base no perfil de comportamento do dispositivo. Além disso, qualquer mudança de comportamento é considerada atividade suspeita; portanto, é necessário realizar outra análise em que o ataque pode já ter ocorrido e ter prejudicado a rede. Koike et al. (2021) demonstrou um método para identificar as funções chamadas de dispositivos IoT usando análise de tráfego para visualizar a atividade do dispositivo, implementado para identificar chamadas de dispositivos mal-intencionados, ou seja, chamadas de dispositivos de não fornecedores. No entanto, não foi apresentado como essa identificação foi feita. Husnain et al. (2022) projetou um mecanismo de análise de transporte de telemetria de enfileiramento de mensagens (MQTT) que pode ser integrado com IDS baseado em rede como uma camada inicial para varredura extensiva em relação ao protocolo IoT vulnerabilidades e uso indevido por meio de validação rigorosa de campos de pacotes durante o estágio de análise de pacotes. No entanto, o mecanismo é focado apenas no protocolo de transporte MQTT deixando de lado os outros protocolos.

Portanto, existem trabalhos que detectam vulnerabilidades, no qual levam em consideração, categorias de aplicações diferentes. Dessa forma, alguns trabalhos detectam vulnerabilidades no nível de *firmware*, no nível de *software* e no nível de rede. Outros trabalhos exploram a detecção de vulnerabilidades, utilizando abordagens em grafos e aprendizado de máquina. Porém, os trabalhos na maioria dos casos necessitam de acesso aos componentes dos dispositivos, em contrapartida, os trabalhos que não necessitam de acesso ao aparelho, suas abordagens possuem um alto custo de memória para a execução. Dessa forma, um método de detecção de vulnerabilidade eficaz, sem a necessidade de acesso aos componentes dos dispositivos e com pouco custo computacional se torna indispensável para a segurança de redes IoT.

3.2 CLASSIFICAÇÃO E ROTULAGEM DO TRÁFEGO

No contexto de detecção de vulnerabilidades, existem esforços na literatura focados unicamente na classificação de tráfego, devido suas aplicações para identificação de ataques e até mesmo para identificação de vulnerabilidades em redes. Um exemplo na identificação de vulnerabilidades se trata da classificação de tráfego sem criptografia, tráfego malicioso e tráfego normal. Além disso, existem esforços focados na rotulagem de tráfego, muitas vezes utilizada para técnicas de impressão digital de dispositivos, mas também utilizada para identificação de vulnerabilidades, no qual amostras de tráfego são rotuladas frequentemente para estratégias de aprendizado de máquina. A rotulagem de amostras de tráfego possui grande importância devido o uso para classificadores aprenderem padrões de tráfego, sendo assim suscetível a identificação de alterações no padrão levando a alertas para o gerente de rede, podendo ser alguma atividade maliciosa.

Tendo em vista os trabalhos que utilizam a técnica de classificação do tráfego, Wang et al. (2011) projetaram um classificador para distinguir o tráfego da Internet em diferentes tipos de conteúdo usando técnicas de aprendizado de máquina, sendo oito categorias de conteúdo analisada, incluindo texto, imagem, áudio, vídeo, compactado, imagem codificada em Base64, texto codificado em Base64 e criptografado. Rezaei e Liu (2019) apresentaram uma estrutura geral para classificação de tráfego baseada em aprendizado profundo (*deep learning*). Porém, estas técnicas não distinguem tráfego criptografado e não criptografado. Puhan et al. (2014) demonstrou um método de detecção de memória descriptografada usando análise dinâmica de fluxo de dados, com base na intuição de que os dados descriptografados são gerados na função criptográfica e na característica única dos dados descriptografados, foram analisados os conjuntos de parâmetros da função criptográfica e assim apresentou um modelo baseado na

entrada e saída da função criptográfica. Entretanto, estas abordagens não se mostram eficazes na detecção utilizando o *Transport Layer Security* (TLS), devido o protocolo empregar técnicas de criptografia no transporte do pacote.

Ma et al. (2020) treinaram um modelo baseado no algoritmo *K-nearest neighbor* (KNN) que necessita apenas de uma pequena quantidade de dados para classificação refinada de fluxo de rede criptografados. Contudo, não foi demonstrado como é detectado o tráfego criptografado. Por Franco et al. (2021), foi introduzida a *SecGrid*, uma plataforma habilitada para aprendizado de máquina para análise, classificação e visualização de ataques cibernéticos, esta plataforma implementa um conjunto extensível de mineradores para analisar informações de rastreamentos de rede para fornecer visualizações perspicazes do tráfego malicioso fornecido e classificar automaticamente diferentes tipos de ataques cibernéticos usando ML supervisionado. Lyu e Lu (2019) aplicaram um método baseado em aprendizado profundo para a classificação de quatro tipos diferentes de tráfego de mídia, ou seja, áudio, imagem, texto e vídeo. Niu et al. (2019) denotaram uma abordagem de teste estatístico heurístico (HST) que combina estatística e aprendizado de máquina e provou aliviar suas respectivas deficiências, selecionando manualmente quatro testes de aleatoriedade para extrair pequenos recursos de carga útil para aprendizado de máquina para melhorar o desempenho em tempo real. Entretanto, os trabalhos acima foram em classificação de tráfego, porem deixando de lado a identificação de tráfego vulnerável.

No contexto de rotulação de tráfego para detecção de vulnerabilidade, Robyns et al. (2017) demonstraram duas novas técnicas não cooperativas de impressão digital e rastreamento da camada MAC para dispositivos móveis habilitados para Wi-FI. A primeira técnica constitui como uma análise de entropia por bit de um único quadro capturado permite que um adversário construa uma impressão digital do transmissor. A segunda técnica aproveita solicitações de serviço de anúncio genérico 802.11u ponto a ponto (GAS) e solicitações de reconhecimento de bloco (BA) 802.11e para instigar transmissões sob demanda de dispositivos que suportam esses protocolos. Gu et al. (2020) apresentam um método de identificação de dispositivos baseado em aprendizado profundo para selecionar recursos automaticamente é estudado e projetado, além disso, demonstram um mecanismo eficiente contra o ataque com base na identificação do dispositivo. Zhang et al. (2018) demonstrou uma nova tecnologia de impressão digital de dispositivos baseada na análise do ambiente de execução, que emprega regressão logística para implementar a identificação online com um mecanismo de janela deslizante, atingindo uma precisão de reconhecimento de 77,03% em um período de 60 minutos. Maurice et al. (2013) apresentaram o Diversity, uma abordagem cooperativa de impressão digital que melhora a precisão dos métodos de impressão digitais existentes, contando apenas com hardware pronto para uso. A diversidade melhora a impressão digital até a identificação individual confiável de dispositivos 802.11 idênticos. Essa abordagem modifica a assinatura dos dispositivos modificando ligeiramente seus atributos de tráfego. Portanto, as abordagens têm em vista a impressão digital do dispositivo, apresentando formas para rotulação de tráfego, porém deixando de lado o uso da rotulação de tráfego para identificação de vulnerabilidades.

Nesse sentido, existem trabalhos que detectam vulnerabilidades no nível de *firmware*, *software* e rede. Outros trabalhos exploram a detecção de vulnerabilidades, utilizando abordagens em grafos e aprendizado de máquina. Porém, na maioria dos casos, os trabalhos necessitam de acesso aos componentes de *software* e *firmware* dos dispositivos. Em contrapartida, os trabalhos que não necessitam de acesso ao dispositivo IoT possuem abordagens com alto custo de memória para a execução e detectam a vulnerabilidade após o ataque ser efetivado. Já os trabalhos que fazem a classificação de tráfego vulnerável se mostram funcionais. Porém, eles possuem restrições em protocolos ou não diferenciam tráfego vulnerável. Já os trabalhos que implementam técnicas de rotulagem de amostras de tráfego, focando na impressão digital do

dispositivo, deixando de lado o uso da rotulação de tráfego para identificação de vulnerabilidades. Dessa forma, um método de detecção de vulnerabilidade eficaz, sem a necessidade de acesso aos componentes dos dispositivos, que identifica dispositivos que geram tráfego vulnerável e com o custo de processamento inferior se torna indispensável para a segurança de redes IoT, pois incrementa as técnicas de detecção da literatura.

3.3 LIÇÕES APRENDIDAS

Com base na análise da literatura, nota-se que a análise do tráfego da rede, em especial no contexto da IoT, permite a detecção de ataques, intrusos, ameaças, entre outros. Entretanto, a análise do tráfego da rede facilita também a detecção de vulnerabilidades, abordagem pouco utilizada nos trabalhos. No geral, os trabalhos necessitam de acesso aos dispositivos e, além disso, necessitam que o ataque já tenha ocorrido para realizar a detecção. Dessa forma, percebe-se que a análise do tráfego da rede para auxiliar métodos e soluções de segurança da IoT na detecção de vulnerabilidades é pouco explorado, havendo oportunidades para novas soluções. A análise do tráfego da rede permite diversas aplicações que auxiliam a segurança da IoT, como detecção da ausência de criptografia, dados críticos da IoT transmitidos sem protocolos de segurança (exemplo, tráfego de câmeras de monitoramento).

Dentre os estudos da literatura relacionada a detecção de vulnerabilidades, percebe-se a relevância no contexto da segurança do usuário, devido os agentes maliciosos se beneficiarem de vulnerabilidade para disseminar ataques em redes com o intuito de roubar dados sensíveis do usuário e até mesmo interromper o serviço. As técnicas atuais muitas vezes se limitam a protocolos, tipos de arquitetura de rede, gargalo de processamento, entre outros. Dessa forma, novas técnicas de detecção que suprem essas limitações são de inteiro proveito para melhoria da segurança do usuário. Em vista as análises da literatura em relação à classificação de tráfego é notável a utilidade desta técnica para categorização de tráfego, na maioria dos casos é usado para diferenciar entre tráfego de diferentes dispostos, também usada para diferenciar entre tráfego malicio e não malicioso, porém possui ligeiras limitações, visto a implementação em protocolos com estratégias de segurança.

3.4 RESUMO

Este capítulo apresentou uma revisão da literatura referente à detecção de vulnerabilidades em diferentes tipos de análise, abordagens e níveis. Na Seção 3.1 foram apresentados os trabalhos que abordam a detecção de vulnerabilidade em diferentes abordagens e níveis, onde demonstra o estado da arte diante das diferentes formas de detecção. A Seção 3.2 apresentou o estado da arte em relação à classificação e rotulagem de tráfego para detecção de vulnerabilidades. Dessa forma, o objetivo desta seção foi identificar as técnicas empregadas para a detecção, para servir como base de conhecimento para o método proposto. A partir deste capítulo é possível observar a carência de trabalhos considerando a detecção de vulnerabilidade simples e eficaz em redes IoT, sem a necessidade de acesso ao dispositivo.

4 MÉTODO MANDRAKE

Este capítulo apresenta o método de detecção de vulnerabilidades de segurança em redes IoT denominado de MANDRAKE, do inglês *a **M**ethod for vulner**A**bilities detectio**N** base**D** on IoT netwo**R**k p**A**c**KE**t traffic*. Seu objetivo consiste em detectar vulnerabilidades a partir da análise do tráfego de rede IoT por meio de técnicas de aprendizado de máquina. Particularmente, o método busca classificar tráfego de rede não criptografado e complementa as estratégias e soluções existentes na literatura. O método rotula automaticamente parte da base de dados e utiliza o mínimo possível de instâncias rotuladas para construir um classificador capaz de identificar se o tráfego possui fluxos vulneráveis e sua origem. O capítulo está organizado como segue. A Seção 4.1 apresenta uma visão geral do escopo da proposta e o cenário de rede. A Seção 4.2 detalha o método proposto, suas características, a coleta do tráfego e a maneira de detecção de vulnerabilidades. Por fim, a Seção 4.3 resume o capítulo.

4.1 VISÃO GERAL

O método MANDRAKE analisa o tráfego de rede sem fio de dispositivos IoT para identificar vulnerabilidades de segurança. Existem diferentes formas de capturar o tráfego da rede para posterior análise, por exemplo, espelhamento de portas de tráfego ou captura em modo promíscuo por meio de uma interface sem fio. Neste trabalho, ressalta-se o cenário e configuração de se capturar o tráfego por meio de uma interface de rede sem fio (por exemplo, *Wi-Fi*), em modo promíscuo e programada para coletar pacotes que trafegam na rede do usuário. Porém, o método foi concebido para trabalhar com uma visão geral e não se limita a essa abordagem de coleta de tráfego.

As características extraídas do tráfego são classificadas em amostras de fluxo correspondentes aos dados analisados para detecção. Essas características são extraídas sem a necessidade do conhecimento prévio da rede, pois são rotuladas conforme as informações dos cabeçalhos dos pacotes como o tamanho médio dos pacotes, a duração, entropia, entre outros. Dessa forma, o método analisa os valores das características seguindo técnicas de classificação a fim de detectar padrões que revelem fluxos de rede provenientes de dispositivos que não empregam técnicas de criptografia em seus pacotes para notificar os usuários, serviços ou administradores e prevenir ou conter eventuais problemas de segurança na infraestrutura.

Esta proposta toma como referência o cenário de uma rede local sem fio composta por dispositivos como câmeras de segurança, televisores, sensores de presença, fechaduras eletrônicas, entre outros, ilustrando uma casa inteligente (*smart home*), cenário escolhido devido sua complexidade e a alta gama de dispositivos, porem a proposta não se limita somente neste cenário, podendo ser utilizado em outros cenários, ex: cenário industrial. A Figura 4.1 mostra um exemplo desta rede local sem fio, com dispositivos heterogêneos que coletam e trafegam dados como as imagens do interior do ambiente, quais aplicações e serviços o usuário está utilizando através da Internet. Nas redes sem fio, os dispositivos da IoT transmitem muitas vezes dados pessoais sensíveis pela Internet, em forma de pacotes padronizados por protocolos. Devido à comunicação sem fio, não é possível saber quem está recebendo ou "farejando" (*sniffing*/ouvindo sem autorização) o tráfego, utilizando por exemplo uma interface de rede Wi-Fi. Neste contexto, um dispositivo vulnerável que não emprega corretamente as técnicas de segurança serve de entrada para usuários mal-intencionados realizar vetores de ataques, controlar os dispositivos remotamente, roubar os dados pessoais e/ou causar prejuízos financeiros.

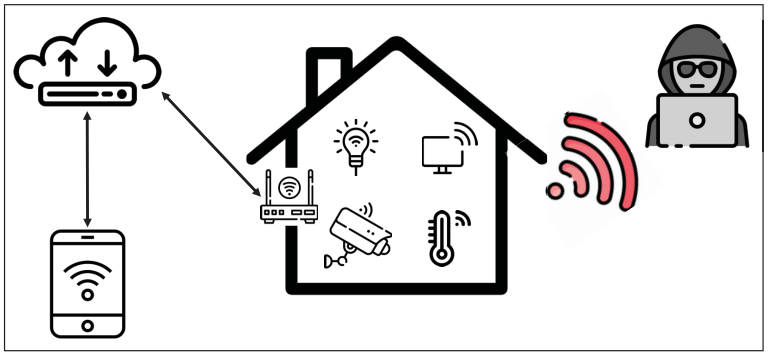


Figura 4.1: Exemplo *Smart Home*

Muito utilizada na literatura, a técnica de entropia mede a desordem contida em um determinado conjunto de e serve como métrica capaz de caracterizar se tal conjunto está criptografado ou não (Dorfinger et al., 2011) Neste trabalho, consideram-se vulneráveis aqueles dispositivos que não empregam técnicas de criptografia. Assim, o método proposto calcula a entropia de *Shannon*, sobre os caracteres contidos na mensagem da camada de aplicação, particularmente, no (*payload*) dos pacotes, e determina quais fluxos de rede são provenientes de dispositivos vulneráveis, ou seja, que os *payloads* não estão criptografados. O método também compreende extrair características estatísticas dos fluxos de rede, como a duração, tamanho médio dos pacotes, instante de início e de fim. A proposta rotula as amostras com base no endereço MAC de cada dispositivo e na definição de fluxos vulneráveis determinados pelos cálculos de entropia. Por fim, a proposta treina os algoritmos de aprendizado de máquina supervisionados e gera modelos capazes de detectar fluxos provenientes de dispositivos vulneráveis sem qualquer conhecimento prévio sobre a infraestrutura ou sobre o *payload* dos pacotes.

Assumimos como vulneráveis aqueles dispositivos que não empregam técnicas de criptografia na transmissão de pacotes, assim esta proposta utiliza a entropia de *Shannon* para auxiliar na criação de modelos de aprendizado de máquina para detecção de vulnerabilidades. Dentre isso, a entropia mede a desordem contida no *payload* dos pacotes e avalia métricas capazes de caracterizar os fluxos como vulneráveis ou não. Assim, os fluxos que possuem uma entropia média menor que 0,7 são determinados como vulneráveis, portanto empregam algoritmos de criptografia ineficientes ou não possuem o *payload* encriptado. O aprendizado de máquina é usado para detectar dispostivos que transmitem fluxos sem o uso de criptografia, sem a necessidade do uso do *payload*, somente com métricas estatísticas extraídas do fluxo. Portanto, o método usa a entropia para rotular as amostras de fluxo vulneráveis e não vulneráveis, assim gerando o modelo de aprendizado de máquina.

4.2 DETALHAMENTO DO MÉTODO

o método MANDRAKE compreende três fases de execução mostradas na Figura 4.2: *(i)* pré-processamento do tráfego, *(ii)* rotulagem do tráfego e *(iii)* classificação do tráfego com base em algoritmos de aprendizagem de máquina. Esta seção detalha essas fases do método proposto. Assim, a Subseção 4.2.1 descreve como funciona o pré-processamento e a extração dos dados a partir do tráfego da rede. A Subseção 4.2.2 apresenta a fase de detecção de vulnerabilidades e explica como o cálculo da entropia foi empregado neste contexto, além de demonstrar a rotulagem do tráfego. A Subseção 4.2.3 detalha a fase de classificação, que emprega aprendizagem de máquina para treinar modelos eficientes de detecção de fluxos vulneráveis.

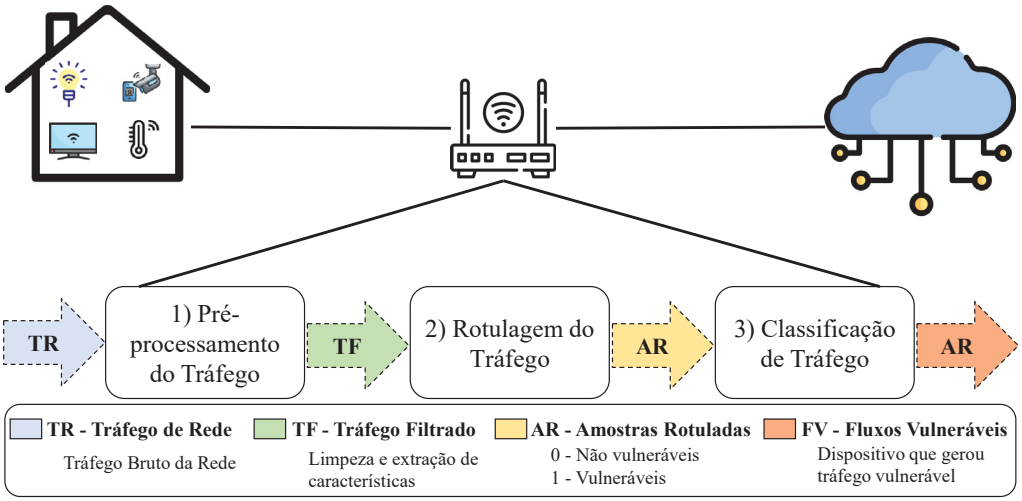


Figura 4.2: Fases de Execução do Método MANDRAKE.

4.2.1 Pré-processamento

A fase de pré-processamento dos dados (i) coleta e filtra o tráfego da rede, (ii) organiza as amostras de tráfego em fluxos e (iii) extrai as características dos pacotes de cada fluxo. Na etapa (i), o método coleta e filtra o tráfego da rede sem fio por meio de um analisador de pacotes (*sniffer*, farejador) em uma placa de rede sem fio no modo promíscuo ou do roteador de borda (*sink node*, *gateway* ou *access point* - AP) da própria infraestrutura almejada. Durante a captura, o *sniffer* filtra os pacotes provenientes das comunicações com a Internet juntamente com as características necessárias para a análise exigida pelo método. O filtro descarta pacotes de *broadcast* na rede local e pacotes com *payloads* vazios, para melhor eficiência do método.

A etapa *(ii)* toma como base o tráfego da rede capturado e separa as amostras do tráfego bruto em fluxos, tanto TCP quanto UDP, conforme a tupla: <IP de origem, IP de destino, porta de origem, porta de destino e protocolo de comunicação>. Cada amostra de fluxo representa um conjunto de pacotes provenientes de uma comunicação de um dispositivo IoT com um serviço ou aplicação na Internet. A partir dos fluxos criados, a etapa *(iii)* calcula uma série de características estatísticas sobre os conjuntos de amostras de pacotes pertencentes ao mesmo fluxo e transforma em uma única amostra de estatísticas do fluxo. As características estatísticas seguem a média (μ), a soma (*sum*) e o desvio padrão (DP) das características de rede como duração do fluxo, instante de tempo *timestamp* de início e fim, e tamanho dos pacotes. No final, cada dispositivo tem suas próprias amostras de fluxos com suas propriedades estatísticas. Portanto, essas etapas são necessárias para obter as características do tráfego e organizar os dados para detectar vulnerabilidades na rede sem fio.

4.2.2 Rotulagem do Tráfego

A fase 2 do método é responsável por rotular os fluxos de tráfego em vulnerável (1) e não vulnerável (0). Para isso, esta fase toma como entrada os fluxos do tráfego filtrados na fase anterior, porém desconsidera as medidas estatísticas na rotulagem. A rotulagem do tráfego vulnerável utiliza apenas os dados oriundos dos fluxos (seguindo a mesma tupla ⟨IP de origem, IP de destino, porta de origem, porta de destino e protocolo de comunicação⟩) e a carga útil (*payload)* de cada pacote pertencente aos fluxos. Sendo assim, seleciona-se um conjunto de pacotes pertencentes a cada fluxo filtrado. Para cada pacote do conjunto selecionado, é

calculada uma estimativa de aleatoriedade. Esta estimativa de aleatoriedade é dada pela fórmula da **entropia** que está relacionada ao grau de desordem de uma variável de entrada Shannon (1948). A fórmula da entropia calcula a aleatoriedade do pacote de rede com base na sua carga útil atribuindo uma nota entre 0 e 1. Essa nota representa a probabilidade do pacote estar criptografado ou não. Quanto mais próximo do valor 1, mais aleatoriedade é encontrada, ou seja, aumenta a possibilidade do pacote de rede estar utilizando técnicas de criptografia. Em contrapartida, uma baixa aleatoriedade na carga útil indica uma provável ausência de criptografia.

Dessa forma, o método MANDRAKE calcula a entropia de *Shannon* sobre os caracteres contidos no *payload* dos pacotes e assim, determina quais fluxos de rede são provenientes de dispositivos vulneráveis. Ou seja, que os *payloads* do tráfego dos dispositivos não estão criptografados. Formalmente, a Equação 4.1 detalha o cálculo da entropia que recebe como entrada a carga útil do pacote sendo uma variável aleatória discreta X, com resultados possíveis $(x_1), ..., (x_n)$, que ocorrem com a probabilidade $P(x_1), ..., P(x_n)$, onde $\sum$ denota a soma dos valores possíveis da variável de entrada X. Além disso, b é a base do logaritmo log usado, tendo em vista a escolha do valor da base variando entre diferentes aplicações, porém Shannon (1948) aponta o valor base igual à 2. Como saída, o cálculo da entropia estima um valor entre 0 e 1 para cada pacote de rede.

$$H(X) = \left\{ - \sum_{i=1}^{N} = P(x_i) \log_b P(x_i) \right. \tag{4.1}$$

Quanto mais próximo de 1 a probabilidade do tráfego estar criptografado aumenta. Porém, de acordo com Dorfinger et al. (2011), os valores acima de $0,7$ estimam corretamente a probabilidade do pacote estar criptografado. O método MANDRAKE contabiliza os pacotes que possuem notas superiores e inferiores à $0,7$, onde uma condição no conjunto de valores estimado pela entropia separa os pacotes com notas superiores e inferiores a $0,7$. Neste condicionamento, o método calcula a taxa de pacotes criptografados em cada fluxo (exemplo, analisa-se entre 70% e 90% dos pacotes do mesmo fluxo possuem criptografia) e assim, estima-se que o tráfego do fluxo não está criptografado, isto é, o tráfego é vulnerável. Por fim, esta fase rotula as amostras com base no endereço MAC de cada dispositivo e na definição de fluxos vulneráveis ("***1***" para as amostras consideradas vulneráveis ou "***0***" para as amostras consideradas não-vulneráveis). Dentre isso, essa fase retorna um conjunto de fluxos vulneráveis e não vulneráveis de acordo com a rotulagem pela entropia.

4.2.3 Classificação do Tráfego Vulnerável

A fase 3 do método recebe como entrada as amostras de fluxo rotuladas, bem como todas as características e medidas estatísticas extraídas do tráfego. Nesta fase, o método MANDRAKE treina os modelos de aprendizagem de máquina com base nas amostras rotuladas para classificar tráfego. Além disso, o método avalia os resultados dos classificadores com base em um limiar de desempenho a fim de definir um conjunto mínimo de amostras rotuladas que sejam suficientes para manter a eficiência do algoritmo. Uma vez que o tráfego de rede IoT é desbalanceado, considera-se a métrica do F1-Score no limiar. Para tanto, o algoritmo de classificação aprende de acordo com o conjunto de treinamento rotulado e gera um modelo de predição. Durante a avaliação do modelo com amostras não rotuladas, o método classifica os fluxos vulneráveis apenas com as seguintes características estatísticas: média, soma e desvio padrão da duração do fluxo, do *timestamp* de início e fim, e tamanho dos pacotes. Desse modo, o método treina o classificador e gera modelos capazes de detectar fluxos provenientes de dispositivos vulneráveis sem qualquer conhecimento prévio sobre a infraestrutura da rede.

Para identificação de dispositivos com tráfego vulnerável, os algoritmos de aprendizado de máquina recebem como entrada um conjunto de treino com características estatísticas extraídas dos fluxos rotulados como vulneráveis e não vulneráveis. Assim, os algoritmos aprendem de acordo com o conjunto de treinamento. Como o foco é detectar fluxos vulneráveis e apontar qual dispositivo está gerando o mesmo, o conjunto de teste é composto por fluxos vulneráveis e não vulneráveis, porém sem rotulagem. Portanto, a saída dos algoritmos de aprendizado de máquina, correspondem a qual dispositivo está gerando tráfego vulnerável com uma excelente assertividade, se tornando menos invasivo.

O método é flexível quanto a especificidade dos classificadores supervisionados. Portanto, a metodologia adotada para coletar o tráfego da rede, organizar as amostras de tráfego em fluxos e extrair as características dos pacotes foi baseada nos problemas de classificação com duas classes e multi-classes. No problema de classificação com duas classes, rotulam-se as amostras de tráfego apenas como vulneráveis 1 e não vulneráveis 0. No problema de classificação multi-classes, as amostras de tráfego são classificadas e rotuladas pelos endereços MAC dos dispositivos, posteriormente classifica-se cada dispositivo como gerador de fluxo vulnerável (rotulado com 1) ou não (rotulado com 0).

4.3 RESUMO

Este capítulo apresentou o método de detecção de vulnerabilidades em redes IoT, detalhando suas características e comportamento. Também demonstrou o funcionamento das fases do método: pré-processamento, rotulagem do tráfego e a classificação de tráfego vulnerável baseada em aprendizado de máquina. Assim, a fase de pré-processamento é responsável por coletar, filtrar e separar em fluxos o tráfego da rede, a fase de rotulagem do tráfego visa identificar as vulnerabilidades e a fase de classificação de tráfego vulnerável baseado em Aprendizado de máquina consiste em criar modelo para detecção de vulnerabilidades. Dessa forma, o método tem como objetivo identificar vulnerabilidade antes de serem exploradas por atacantes e sem o conhecimento prévio da rede. Além disso, cria modelos de aprendizado de máquina capazes de identificar dispositivos que geram tráfego vulnerável.

5 AVALIAÇÃO DE DESEMPENHO

Este capítulo apresenta a metodologia de avaliação e os resultados do método MANDRAKE. As avaliações foram conduzidas de forma *offline* e *stream*, com dados gerados em um cenário realístico de uma casa inteligente e com um cenário orientado a traços. As métricas de desempenho utilizadas são específicas para cada técnica empregada. Para fins de avaliação, as fases de rotulagem do tráfego e classificação de tráfego com algoritmos de aprendizagem de máquina compreendem, respectivamente, rotular a um conjunto de amostras de tráfego com base em um limiar de entropia e treinar modelos eficientes que não utilizam a métrica da entropia como entrada. Este capítulo está organizado como segue. A Seção 5.1 descreve a metodologia de avaliação, os cenários utilizados e as métricas empregadas na obtenção dos resultados. A Seção 5.2 apresenta e discute os resultados alcançados.

5.1 METODOLOGIA

A metodologia de avaliação empregada neste trabalho segue um cenário experimental e um cenário orientado a traços. As métricas de avaliação empregadas determinam o desempenho das fases de rotulagem e classificação de tráfego. O cenário experimental foi composto por dispositivos IoT, conectados em um ponto de acesso *WiFi* e gerando tráfego conforme as suas respectivas aplicações. O cenário orientado a traços é constituído pelo conjunto de dados *IoT Traffic Traces* que contém o tráfego de uma casa inteligente (Sivanathan et al., 2019). A fase rotulagem de tráfego foi avaliada empiricamente para determinar um limiar capaz de definir se o tráfego está criptografado ou não, pois ela rotula automaticamente as amostras conforme os níveis de entropia. Além disso, realiza-se uma análise sobre a seleção de características de tráfego. As avaliações referentes à fase de classificação de tráfego vulnerável com aprendizagem de máquina emprega métricas de avaliação como acurácia, precisão, *recall* e *F1-Score*.

5.1.1 Cenários de Avaliação

Com base no conjunto de dados da rede do ambiente experimental e do *dataset*, extraem-se as características de rede, criam-se as tuplas de fluxo, calcula-se a entropia para rotulagem dos dados e classificam-se os fluxos. A motivação para criação de um ambiente experimental se deve à necessidade de avaliar o método MANDRAKE em um ambiente realista, neste caso, uma casa inteligente. A limitação no número de dispositivos IoT no ambiente experimental motivou o uso de um conjunto de dados também de uma casa inteligente, permitindo a análise em uma maior diversidade de dispositivos. Além disso, o uso de um conjunto de dados facilita a reprodutibilidade do trabalho. Em ambas as abordagens de avaliação, considera-se a execução do método MANDRAKE no ponto de acesso da casa inteligente. Uma vez que nas redes sem fio os dispositivos IoT transmitem dados pessoais em forma de pacotes padronizados por protocolos populares, não é possível saber quem está recebendo ou "farejando" (*sniffing*/ouvindo sem autorização) o tráfego da rede Neste contexto, um dispositivo vulnerável que não emprega corretamente as técnicas de criptografia serve como entrada para usuários mal-intencionados realizar vetores de ataques, controlar os dispositivos remotamente, roubar os dados pessoais e/ou causar prejuízos financeiros.

Para a implementação e análises, a linguagem de programação *Python* versão 3.7 é usada seguido das bibliotecas auxiliares *Pandas* e *Numpy*. A biblioteca *Pandas*, versão 1.1.3, é

utilizada para tratamento e análise de dados do tráfego e a biblioteca *Numpy*, versão 1.19.2, é utilizada para o processamento de grandes quantidades de dados juntamente com uma grande coleção de funções matemáticas, sendo a soma, média e desvio padrão.

5.1.1.0 Cenário 1: Casa Inteligente - Cenário Experimental

A Tabela 5.1 detalha os dispositivos do cenário experimental. No total são 5 dispositivos, sendo um PS4 (Playstation 4), duas câmeras de monitoramento, uma TV *smart* e um *smartphone* como por ser visto na Figura 5.1. Tais dispositivos são de diferentes fabricantes como *Sony*, *Yoosee*, *Samsung* e Motorola. Todos os dispositivos enviam e recebem pacotes de dados via *Wi-Fi* em carga alta, pois acessam serviços que geram alta taxa de tráfego. No geral, os dispositivos realizam operações de *streaming* de vídeo em diferentes aplicações. As câmeras de monitoramento transmitem as imagens gravadas para o servidor na nuvem de cada fabricante. O *smartphone*, por sua vez, acessa e recebe estas imagens, gerando assim uma alta taxa de pacotes de dados transmitidos. A TV *smart* está conectada com o serviço de *stream* do *Yotube* e o *Playstation* 4 conectado a um jogo *online* de sua plataforma, a fim de gerar tráfego de rede. Particularmente, a fase de coleta do tráfego foi executada em um laptop contendo o sistema operacional *Ubuntu Desktop* na versão 20.04 LTS, por meio da ferramenta *Wireshark* (Orebaugh et al., 2007). O tráfego de rede foi coletado por 2 horas, gerando 2.9 GB de dados.

Dispositivo	**Endereço MAC**	**Marca**	**Tipo de Conexão**
PS4	e8:d8:19:ea:1a:ab	Sony	Sem fio
Camera 1	b4:c9:b9:f9:74:e1	Yoosee	Sem fio
Camera 2	b4:c9:b9:f9:76:ca	Yoosee	Sem fio
TV Smart	64:e7:d8:0d:4d:24	Samsung	Sem fio
Smartphone	80:58:f8:ec:d5:d6	Motorola	Sem fio

Tabela 5.1: Dispositivos Cenário Experimental

5.1.1.0 Cenário 2: Casa Inteligente - Cenário Orientado a Traços

O conjunto de dados *IoT Traffic Traces* (Sivanathan et al., 2019) possui a coleta do tráfego da rede de uma casa inteligente composta por 28 dispositivos IoT e não-IoT representado pela Tabela 5.2. Na tabela, é possível encontrar também os endereços MAC e o tipo de conexão por dispositivo. A coleta do tráfego da rede ocorreu do dia 22 de setembro de 2016 à 12 de outubro de 2016, totalizando 20 dias de captura de tráfego de rede, 29.3 GB de dados e 1629879 pacotes. A captura foi dividida em 20 arquivos *pcaps* referentes a cada dia de coleta. O *dataset* contém o registro do tráfego de rede de diferentes dispositivos IoT e não-IoT, como assistentes virtuais, lâmpada inteligente, sensor de movimento, monitor de pressão arterial, monitor de bebê, balança Wi-Fi, notebooks, sensor de movimento, smartphones, porta-retrato digital, caixa de som portátil, interruptor de luz inteligente, câmeras de monitoramento Wi-Fi, *gateway*, impressoras, entre outros. Por ser um conjunto de dados composto por diferentes dispositivos, as características do tráfego da rede apresentam variabilidade de um dispositivo para o outro, como diferentes tamanhos de pacotes. Desse modo, o *dataset* possui uma ampla diversidade com dados sensíveis coletados pelos dispositivos, viabilizando a avaliação da detecção de vulnerabilidades.

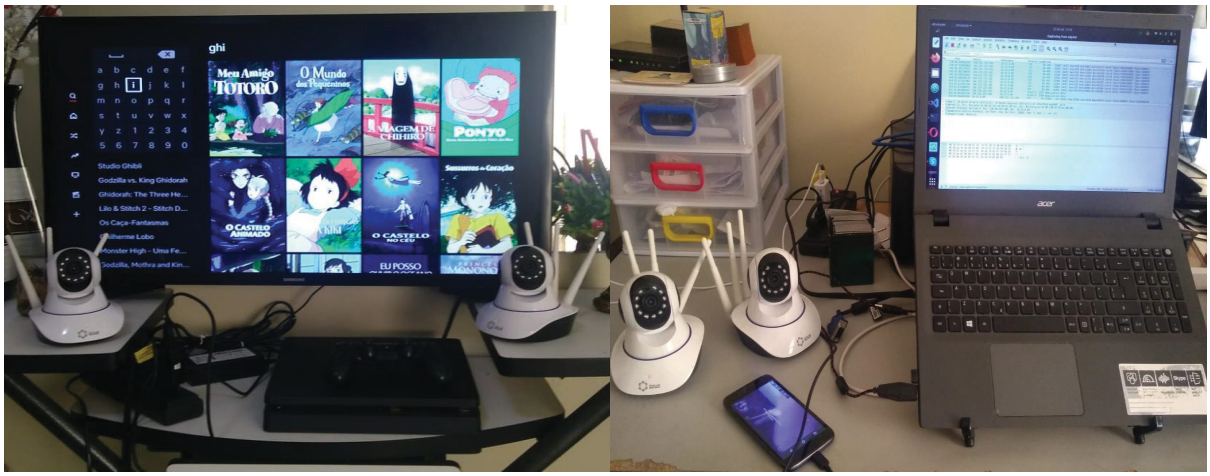


Figura 5.1: Cenário Experimental

Dispositivos	Endereço MAC	Conexão
Amazon Echo	44:65:0d:56:cc:d3	Sem fio
Netatmo Welcome	70:ee:50:18:34:43	Sem fio
TP-Link Day Night Cloud camera	f4:f2:6d:93:51:f1	Sem fio
Samsung SmartCam	00:16:6c:ab:6b:88	Sem fio
Dropcam	30:8c:fb:2f:e4:b2	Sem fio
Insteon Camera	00:62:6e:51:27:2e	Com fio
Withings Smart Baby Monitor	00:24:e4:11:18:a8	Com fio
Belkin Wemo switch	ec:1a:59:79:f4:89	Sem fio
TP-Link Smart plug	50:c7:bf:00:56:39	Sem fio
iHome	74:c6:3b:29:d7:1d	Sem fio
Belkin wemo motion sensor	ec:1a:59:83:28:11	Sem fio
NEST Protect smoke alarm	18:b4:30:25:be:e4	Sem fio
Netatmo weather station	70:ee:50:03:b8:ac	Sem fio
Withings Smart scale	00:24:e4:1b:6f:96	Sem fio
Blipcare Blood Pressure meter	74:6a:89:00:2e:25	Sem fio
Withings Aura smart sleep sensor	00:24:e4:20:28:c6	Sem fio
Light Bulbs LiFX Smart Bulb	d0:73:d5:01:83:08	Sem fio
Triby Speaker	18:b7:9e:02:20:44	Sem fio
PIX-STAR Photo-frame	e0:76:d0:33:bb:85	Sem fio
HP Printer	70:5a:0f:e4:9b:c0	Sem fio
Samsung Galaxy Tab	08:21:ef:3b:fc:e3	Sem fio
Nest Dropcam	30:8c:fb:b6:ea:45	Sem fio
Android Phone	40:f3:08:ff:1e:da	Sem fio
Laptop	74:2f:68:81:69:42	Sem fio
MacBook	ac:bc:32:d4:6f:2f	Sem fio
Android Phone	b4:ce:f6:a7:a3:c2	Sem fio
IPhone	d0:a6:37:df:a1:e1	Sem fio
MacBook/Iphone	f4:5c:89:93:cc:85	Sem fio

Tabela 5.2: Dispositivos do conjunto de dados Sivanathan et al. (2019)

5.1.2 Capturas de Tráfego

As capturas do tráfego da rede, realizada por meio da ferramenta Tshark[1], do ambiente experimental e do *dataset* servem de entrada para a avaliação do método. Assim, por meio das bibliotecas Numpy[2] e Pandas[3] da linguagem Python v3, extraem-se as características da rede, criam-se as tuplas de fluxo e computam-se as medidas estatísticas do tráfego da IoT. As características de rede consideradas envolvem as características da tupla do fluxo ⟨ endereço IP de origem e destino, número de porta de origem e destino, protocolo da camada de transporte ⟩, além do endereço MAC, *payload*, tamanho do pacote, instante de início e fim do fluxo, e duração do fluxo. Para o cálculo das medidas estatísticas e da entropia, criaram-se amostras de fluxo com 10 pacotes visto que alguns dispositivos IoT geraram pouca quantidade de pacotes de rede (exemplo, 10 pacotes por dia). A partir das amostras de fluxo, computam-se as medidas estatísticas da média, soma e desvio padrão. Por fim, geraram-se dois arquivos: um arquivo composto pelas amostras de fluxo e *payload*, e um segundo arquivo contendo as amostras de fluxo com as respectivas medidas estatísticas. O primeiro arquivo serve de entrada para o cálculo da entropia.

5.1.3 Pré-processamento do Tráfego

Na fase de pré-processamento do método, é realizada a coleta e a filtragem do tráfego da rede, organizando as amostras de tráfego em fluxos e extraindo as características dos pacotes de cada fluxo. Dentre isso, a extração de característica é denominada para cada endereço MAC (dispositivo) da rede em questão. Então, para cada dispositivo, são extraídas características como seguem: Endereço IP de origem e Endereço IP de destino, Endereço MAC de origem e Endereço MAC de destino, Porta de origem, *timestamp* e *payload* de cada pacote dos fluxos. Além disso, considerando o fluxo, as novas características são extraídas sendo a duração do intervalo entre início e fim dos pacotes, em conjunto com o total do tamanho dos pacotes de cada fluxo. Para calcular os valores das medidas estatísticas, foram criadas pequenas amostras para os fluxos extraídos. Assim, conforme a Tabela 5.3 foram calculados a soma (sum), a média (μ) e o desvio padrão (DP) das amostras. No final, cada dispositivo tem suas próprias amostras e propriedades estatísticas, além disso, é empregado o rótulo de acordo com seu endereço MAC e definição de fluxo vulnerável denominado na Subseção 2.2.

Medida Estatística	**Equação**
Soma	$sum = \sum_{i=1}^{N} x_i$
Média	$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$
Desvio Padrão	$DP = \sqrt{\frac{\sum_{i=1}^{n} (x_i - M_a)^2}{n}}$

Tabela 5.3: Medidas Estatísticas para Detecção de Tráfego Vulnerável

5.1.4 Técnicas de Rotulagem via Entropia

O cálculo da entropia, por sua vez foi implementado seguindo a linguagem de programação *Python* na versão 3, baseado no exemplo de BlackYe (2016), porém adaptado para o método

[1]Tshark - Wireshark: https://www.wireshark.org/docs/man-pages/tshark.html. Acessado em Fev/2022.
[2]Biblioteca Numpy: https://numpy.org/. Acessado em Fev/2022.
[3]Biblioteca Pandas: https://pandas.pydata.org/. Acessado em Fev/2022.

de detecção apresentado nesta dissertação. No qual o cálculo utiliza o *payload* extraído de cada pacote, medindo seu grau de aleatoriedade. Seguindo a Equação 4.1, o *payload* é representado por um *string* de acordo com a aplicação que constituiu o pacote. Dessa forma, a fórmula recebe como entrada a *string* calculando seus valores possíveis entre $(x_1), ..., (x_n)$ que ocorrem com a probabilidade $P(x_1), ..., P(x_n)$, onde $\sum$ denota a soma dos valores possíveis da variável de entrada, no qual b é a base do logaritmo log usado, tendo em vista a escolha do valor da base variando entre diferentes aplicações, porém Shannon (1948) aponta o valor base igual à 2.

A entropia, por sua vez foi avaliada empiricamente, sendo implementada para medir a aleatoriedade do *payload* dos pacotes da rede. Dessa forma, a cada *payload* analisado antes da execução do cálculo da entropia, uma tentativa de decodificação de caracteres é usada, pois o conteúdo do pacote extraído corresponde num conjunto de caracteres (*string*). Assim essa tentativa de decodificação valida a nota dada pela fórmula da entropia. Quando esta tentativa de decodificação é bem sucedida a nota da entropia automaticamente apresentava um valor inferior a $0,7$, compreendendo a teoria apresentada por Dorfinger et al. (2011). Caso contrário quando está tentativa de decodificação do *payload* era mal sucedida a nota da entropia correspondia a um valor superior a $0,7$. Portanto, esta tentativa de decodificação demonstrou que o cálculo da entropia é eficiente na detecção de alto índice de aleatoriedade dos pacotes. Em conjunto para a validação da entropia, algoritmos de aprendizado de máquina foram utilizados. Seguindo uma classificação binária, onde a entropia é usada para rotular um conjunto de amostras de tráfego entre vulnerável (1) e não vulnerável (0).

5.1.5 Classificadores

A identificação de dispositivos que geram fluxos vulneráveis após estarem rotulados, engloba quatro algoritmos de aprendizagem de máquina amplamente utilizados e bem conhecidos na literatura (Pacheco et al., 2018). Tais algoritmos compreendem um algoritmo baseado em árvores de decisão *Adaptive Random Forrest*, outro compreende uma rede neural *Multilayer Perceptron*, *K-Nearest Neighbors* baseado em vizinhos mais próximos e por fim o *Naive Bayes* que se baseia no teorema de Bayes descreve a probabilidade de um evento. O *Random Forrest* é denominado um algoritmo *ensemble*, pois ele utiliza um conjunto de árvores de decisão para a classificação, assim cada árvore é treinada individualmente e ao final os resultados são combinados. Dessa forma, este algoritmo trata perfeitamente conjunto de dados desbalanceados no caso desta dissertação. O *Multilayer Perceptron* denominado uma rede neural artificial *feedforward* composta por camadas de neurônios ligadas entre si por sinapses com pesos. O algoritmo *K-Nearest Neighbors* armazena todos os dados disponíveis e classifica um novo ponto de dados com base na medida de similaridade por exemplo, funções de distância. Já o *Naive Bayes* inspira um algoritmo probabilístico, o que significa que prevê com base na probabilidade de um objeto. Portanto, para verificar o algoritmo de classificação seguiu-se o processo tradicional, utilizando o conjunto de dados de treinamento (70% dos dados) e teste (30% dos dados) para treinar o modelo de classificação e calcular as métricas de desempenho.

A classificação dos fluxos vulneráveis seguiu dois cenários: duas classes e multi-classes. Nas duas classes, classificam-se os fluxos como vulneráveis (1) e não vulneráveis (0). Enquanto no multi-classes, o método aponta se os diferentes dispositivos IoT possuem fluxo vulnerável. Além disso, com o propósito de tornar mais realista a avaliação, avaliaram-se os classificadores seguindo dois modos de aprendizagem: *offline* (*m-off*) e *stream* (*m-st*). O modo *offline* compreendeu a implementação tradicional dos algoritmos de classificação por meio da biblioteca *Scikit-Learn*[4]. Assim, os algoritmos receberam como entrada as amostras de tráfego divididas em bases de

[4]Biblioteca Scikit-Learn: https://scikit-learn.org/stable/. Acessado em Fev/2022.

treino e teste. Nesta avaliação, a separação das amostras não respeita a linha do tempo do tráfego analisado. Este fato torna possível que os modelos gerados pelos algoritmos sejam treinados de uma vez só com uma quantidade maior de informações. Em contrapartida, o modo de aprendizado em *stream* utilizou a implementação da biblioteca *Scikit Multiflow*[5] que modifica os classificadores para que eles sejam parcialmente treinados e testados durante o decorrer da base, respeitando a ordem que o tráfego de rede foi gerado. Este modo de aprendizado em *stream* (*m-st*) permite que um único modelo seja treinado parcialmente mais de uma vez e segue as amostras em formato de *stream* para simular o fluxo de dados de um cenário realista.

As métricas compreendem avaliar os métodos de classificação avaliado para identificar dispositivos que geram fluxos vulneráveis. As métricas consideradas envolvem a acurácia, precisão, *recall* e *F-Score* (também conhecida como *F-Measure*). Estas métricas consideram a taxa de verdadeiros positivos (VP), verdadeiros negativos (VN), falsos positivos (FP) e falsos negativos (FN). Assim, a acurácia se refere à proporção do tráfego de dados classificados corretamente em relação a todas as amostras de tráfego. A precisão (p) estima a porcentagem de verdadeiros positivos ($VP/(VP + FP)$) em todas as amostras de tráfego classificadas como positivas. O *recall* (r) ou revocação consiste na porcentagem de amostras verdadeiras positivas em todas as amostras cuja categoria esperada é positiva ($VP/(VP + FN)$). Finalmente, *F-Score* estabelece uma relação entre a medida de precisão e *recall* estimando a média harmônica ($2rp/(r + p)$). Portanto, os resultados dos classificadores são baseados nessas métricas.

5.2 RESULTADOS

Esta seção apresenta os resultados das análises realizadas do método MANDRAKE proposto. O método visa detectar vulnerabilidades de rede, como dispositivos que geram tráfego vulnerável na rede. Para isso foi criado um cenário experimental, também usado um cenário orientado a traços para execução dos testes. Os resultados se embasam no tráfego de dispositivos IoT de uma casa inteligente. Tais dispositivos como, câmeras de monitoramento, Tv *Smart*, *Smartphone*, lampadas inteligentes, sensores e atoadores conectados a internet sem fio e com fio. Assim, os resultados são apresentados e discutidos seguindo uma análise para cada cenário e suas respectivas vulnerabilidades no tráfego. Os resultados seguem dois métodos de avaliação seguindo a fase dois constituído pela rotulagem do tráfego e a fase três constituído pela classificação do tráfego.

A Tabela 5.4 apresenta os resultados relacionados à avaliação empírica do cálculo da entropia, seguindo a aplicação local enviando e recebendo pacotes e as máquinas virtuais em rede trocando informações isolados do tráfego normal. Em ambos os cenários, os resultados da detecção da vulnerabilidade de ausência de criptografia foram similares e satisfatórios. Precisamente, na aplicação local, a precisão da entropia manteve-se em torno de 90%, identificando corretamente os pacotes criptografados, e 100% de precisão para pacotes sem criptografia. No ambiente da máquina virtual, a precisão da entropia manteve-se em torno de 91%, identificando corretamente os pacotes criptografados e 100% de precisão para pacotes sem criptografia. Vale ressaltar que cada pacote trafegado foi analisado individualmente. Para conteúdo do pacote sem técnica de criptografia a nota da entropia era inferior a $0,7$, caso contrário, a nota era superior a $0,7$. Com base nos resultados satisfatórios da entropia, é possível seguir com a detecção de fluxos vulneráveis por meio dos classificadores.

A Figura 5.2 apresenta uma visualização da taxa total de pacotes vulneráveis trafegados por dispositivos no cenário experimental criado devido ao maior tempo para análise do cenário. Neste caso, os pacotes vulneráveis se tratam de pacotes sem uso de técnicas de criptografia, no

[5]Biblioteca Scikit Multiflow: https://scikit-multiflow.github.io/. Acessado em Fev/2022.

Métrica	Aplicação Local		Máquina Virtual	
	Criptografado	N/Criptografado	Criptografado	N/Criptografado
Precisão	90%	100%	91%	100%

Tabela 5.4: Avaliação Empírica do Cálculo da Entropia

qual se tornam propensos a brechas de segurança. Como pode ser visto, a Tv *Smart* se destaca com mais de 60% do seu tráfego vulnerável, devido estar conectada a um serviço de *stream* de vídeo do *Yotube* que na maioria das vezes utiliza o protocolo de transporte UDP (*User Datagram Protocol*) sem necessidade de técnicas de criptografia nos pacotes. Mesmo não necessitando de criptografia, este tráfego está propenso a exploração vulnerabilidade, do ponto de vista do atacante uma informação simples do usuário se mostrar interessante para outras aplicações induzirem o mesmo a usar novos serviços da web. O *Smartphone* apresenta 50% do tráfego vulnerável, sendo que sua finalidade era receber as imagens gravadas pelas câmeras, porém outras aplicações rodavam no celular por ser de uso pessoal, exemplo: (*WhatsApp* e *Facebook*). O PS4 estava conectado a um jogo de sua plataforma teve a menor taxa de pacotes vulneráveis do cenário experimental, tendo menos de 12% do seu tráfego gerado sem técnicas de criptografia. Já as câmeras obtiveram resultados semelhantes, pois se tratam de câmeras do mesmo fabricante, no qual sua taxa de pacotes vulneráveis ficou em média a 20% de tráfego vulnerável.

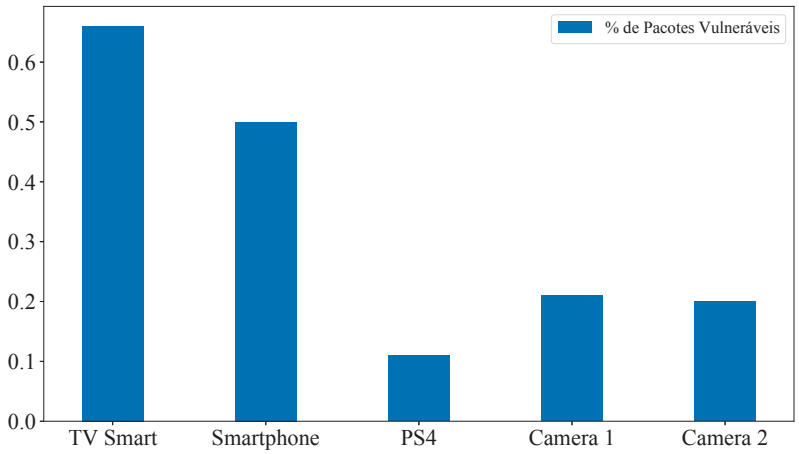


Figura 5.2: Taxa de Pacotes Vulneráveis por Dispositivo

Na Figura 5.2, observa-se que a TV *Smart* possui a maior de taxa de pacotes vulneráveis. Dessa forma, a Figura 5.3 apresenta por quais portas o tráfego vulnerável foi transmitido, além disso, demonstra a porcentagem de pacotes transmitidos por cada porta. Assim, os resultados mostram que o tráfego da Tv *Smart* é transmitido por diferentes portas, e a taxa de pacotes não criptografados variam entre 40% do tráfego vulnerável transmitido por uma determinada porta, até 100% do tráfego transmitido sendo vulnerável por uma determinada porta. Portanto, é possível uma análise precisa da rede, apontando onde o tráfego vulnerável está sendo gerado, ou seja, qual porta os pacotes estão trafegando.

As Figuras 5.4 e 5.5 mostram o desempenho do método MANDRAKE na detecção dos fluxos vulneráveis seguindo os cenários experimental e orientado a traços. Estes resultados seguem dois cenários de avaliação, duas classes e multi-classes no modo de aprendizagem *m-st* discutido na Subseção 5.1.5. Seguindo a forma de treinamento onde os classificadores foram

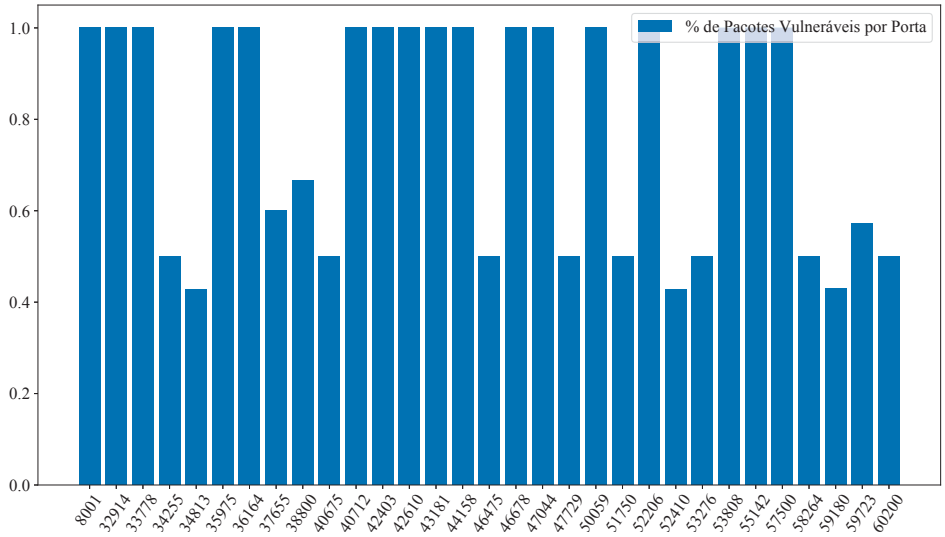
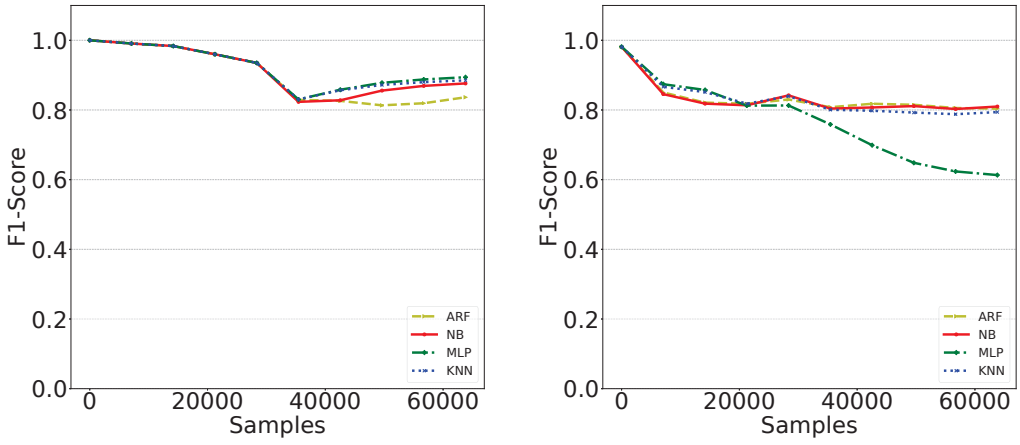


Figura 5.3: Taxa de Pacotes Vulneráveis por Porta da Tv *Smart*

treinados parcialmente com 100 primeiras instâncias de cada classe e no decorrer da base eles foram treinados novamente toda vez que o limiar de 80% de F1-Store era atingido, visando encontrar o menor modelo de treinamento para detecção de fluxos vulneráveis, deixando de lado o uso a rotulação usando a entropia. As Figuras 5.4(a) e 5.4(b) representam a classificação duas classes e multi-classes do cenário experimental. A Figura 5.4(a) aponta que os classificadores mantiveram o desempenho ≈ 80% de F1-Score na classificação duas classes em toda a classificação de amostras sem necessidade de novas amostras de treino para classificação devido ao limiar F1-score não ter sido inferior a 80% no decorrer da base de teste. A Figura 5.4(b) mostra que devido à complexidade de multi-classes os classificadores tiveram uma queda em relação a duas classes, porém mesmo assim tendo resultados satisfatórios dos classificadores terem resultados superior ao limiar de 80%, todavia nota-se que o classificador MLP precisou de mais amostras de tráfego, contudo não conseguiu obter resultados acima 80% de F1-Score.



(a) Duas Classes

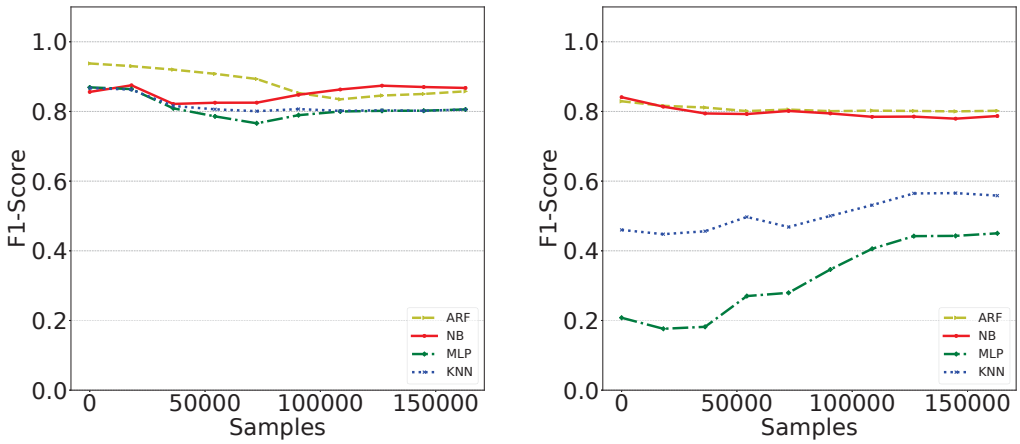
(b) Multi-classes

Figura 5.4: Classificação Cenário Experimental

A Figura 5.5(a) aponta que os classificadores tiveram resultados satisfatórios na classificação no cenário mais complexo com mais quantidade de dados. Mantiveram o desempenho de ≈ 80% de F1-Score na classificação duas classes em toda a classificação de amostras sem necessidade de novas amostras de treino para classificação devido ao limiar F1-score não ter sido

inferior a 80% no decorrer da base de teste. A Figura 5.5(b) mostra que devido à quantidade de dados e a complexidade de multi-classes os classificadores tiveram uma queda nas duas classes, porém mesmo assim os classificadores ARF e NB tiveram resultados satisfatórios com resultados superiores ao limiar de 80%. Nota-se que os classificadores MLP e KNN necessitaram de mais amostras de tráfego, contudo não conseguiram obter resultados acima 80% de F1-Score.

Esses resultados demonstram que a rotulagem da entropia, a qual serviu de entrada para os classificadores detectarem tráfego vulnerável e não vulnerável, obteve sucesso, visto que todos os classificadores obtiveram ≈ 80% de F1-Score. Em contrapartida, o cenário de detecção de dispositivos contendo tráfego vulnerável (multi-classes) apresentou valores próximos ou menores do que 60% de F1-Score, atingindo 20% no algoritmo MLP da Figura 5.5(b). Isso se deve ao fato da base orientada a traços possuir uma maior quantidade de dispositivos do que a base experimental, o que impactou nos algoritmos KNN e MLP que tomam como base a similaridade dos dados. Por isso, mesmo utilizando todas as instâncias no treinamento (150000), os algoritmos não conseguiram atingir o limiar de 80% de F1-Score. Além disso, nos cenários multi-classes, o algoritmo NB, que utiliza distribuições de probabilidade, manteve o F1-Score médio de 81,5% e, para manter esse resultado, precisou de 72% da base no treinamento.



(a) Duas Classes

(b) Multi-classes

Figura 5.5: Classificação Cenário Orientado a Traços

A Tabela 5.5 mostra os resultados considerando todas as métricas, bem como os cenários de duas classes (Duas) e multi-classes (Mult), os modos de aprendizagem *m-off* e *m-st*, e as duas bases experimental e orientada a traços. Os resultados corroboram que o cenário de duas classes (taxa de precisão entre 85% e 99%) apresenta um desempenho melhor do que o multi-classes (taxa de precisão entre 32% e 98%). Em relação aos modos de aprendizagem, observa-se que no modo *m-off* os resultados atingiram valores médios de 91% de F1-Score, relativamente maior que no *m-st* com 79% de F1-Score médio. Estes resultados se justificam, pois o modo *m-off* não respeita a ordem do fluxo ao dividir as amostras e melhora o balanceamento dos dados. No entanto, note que mesmo com essa vantagem o algoritmo NB apresentou acurácia de ≈ 60% no modo *m-off*, sendo menos eficiente que os demais classificadores considerados nesta avaliação. Tanto no cenário de classificação com duas classes quanto no multi-classes, o algoritmo RF apresentou as maiores taxas médias de acerto na detecção de fluxos vulneráveis, com ≈ 91% de F1-Score, enquanto o algoritmo KNN atingiu a segunda melhor taxa com ≈ 85% de F1-Score médio. Portanto, com base nestes resultados, o método MANDRAKE apresentou sucesso na detecção de fluxos vulneráveis no ambiente experimental e orientado a traços.

Os resultados apontam que o método MANDRAKE torna possível a rotulação da base, identificação da presença de tráfego vulnerável e de seus respectivos dispositivos de origem, sem

Classificador	Duas/ Mult	Exp/ Traços	Acurácia		Precisão		Recall		F1-Score	
			Off	St	Off	St	Off	St	Off	St
Random Forest	Duas	**Exp**	**99**	**91,3**	**99**	**91,3**	**99**	**91,2**	**99**	**89,9**
		Traços	98	88,5	98	88,5	98	89,2	96	88,3
	Mult	**Exp**	**98**	**84,4**	**99**	**84,4**	**99**	**84,7**	**98**	**83,5**
		Traços	95	82,1	95	82,1	95	81,7	95	80,7
Multilayer Perceptron	Duas	**Exp**	**97**	**93,2**	**98**	**93,2**	**95**	**92**	**96**	**92,2**
		Traços	88	82	88	82	88	80,8	88	80,9
	Mult	**Exp**	**95**	**79,3**	**95**	**79,3**	**95**	**75,3**	**95**	**76,8**
		Traços	77	32	86,3	32	86,2	32,1	85,5	32
NaiveBayes	Duas	**Exp**	**97**	**92,3**	**96**	**92,3**	**96**	**91,4**	**93**	**91,2**
		Traços	84	84,3	83	84,3	83	88	83	85,3
	Mult	**Exp**	**93**	**84,4**	**94**	**84,4**	**94**	**84,6**	**93**	**83,3**
		Traços	60,3	80	60,6	80	60	81,8	57	79,7
K-Nearest Neighbor	Duas	**Exp**	**98**	**93**	**98**	**93**	**99**	**92,4**	**97**	**91,9**
		Traços	93,2	83	93,2	83	92	82	92,2	81,7
	Mult	**Exp**	**96**	**85,5**	**96**	**85,5**	**96**	**83,8**	**96**	**83,3**
		Traços	85	53,4	85	53,4	85,7	49,5	85	50,5

Tabela 5.5: Resultados de Detecção de Fluxos Vulneráveis do Método MANDRAKE

conhecimento prévio da rede. Cada uma das fases do método foi avaliada experimentalmente com bases realistas. A Fase *(1)* (pré-processamento do tráfego) tornou possível a captura e criação de amostras de fluxos. A Fase *(2)* (rotulagem do tráfego com base na entropia) não apresentou falsos negativos, ou seja, nenhuma amostra vulnerável foi identificada como não-vulnerável. Este fato confirma a eficiência do cálculo da entropia. Por fim, a Fase *(3)* (classificação dos fluxos vulneráveis) apresentou altas taxas de acerto em diferentes cenários de avaliação e problemas de classificação, com acurácia entre 88,5% e 99% nos problema de duas classes e com acurácia entre 82,1% e 98% no problema multi-classes.

5.3 RESUMO

Este capítulo apresentou a metodologia de avaliação de desempenho e os resultados alcançados do método MANDRAKE. O método por sua vez compreendem em detectar vulnerabilidades presentes em redes com base na análise de tráfego. Assim, a avaliação de desempenho seguiu duas metodologias. A primeira metodologia avaliou empiricamente o cálculo da entropia para rotulagem do tráfego, essa metodologia foi conduzida em dois cenários sendo uma Aplicação Local enviando e recebendo pacotes e duas Máquinas Virtuais em rede trocando informações isolados do tráfego normal. Dessa forma, confirmou-se que o tráfego de dispositivos IoT muitas vezes não usa técnicas de criptografia, ou seja, tráfego vulnerável. A segunda metodologia de avaliação foi conduzida de forma *offline* e *stream* com dados gerados em um cenário experimental realístico e com cenário orientado a traços, sendo o conjunto de dados Sivanathan et al. (2019). Visto isso, confirma-se que é possível criar modelos de aprendizado de máquina e classificar corretamente tráfego vulnerável usando caraterísticas estatísticas extraídas do tráfego. Portanto, o método proposto cumpre a finalidade de detectar vulnerabilidade no tráfego, demonstrando que muitas aplicações não utilizam técnicas de criptografia na transmissão de pacotes, além disso, o método tem resultados satisfatórios em diferentes tipos de rotulagem e classificação discutidos neste capítulo.

6 CONCLUSÃO

Este capítulo conclui o presente trabalho, onde foi introduzido o método MANDRAKE. MANDRAKE detecta vulnerabilidades em redes de comunicação IoT, visto que muitas vulnerabilidades são exploradas por agentes mal-intencionados para comprometer a segurança do usuário. Por exemplo, a falta de criptografia na comunicação facilita o acesso indevido de invasores às informações confidenciais das rotinas dos usuários. Na literatura, os trabalhos apresentam técnicas de detecção de vulnerabilidades na IoT de diversas formas. Entretanto, a detecção convencional de vulnerabilidades segue bancos de dados que listam os CVEs, que se limitam a vulnerabilidades conhecidas ja detectadas, e criar CVEs para IoT é um desafio devido às restrições de recursos, diversidade e heterogeneidade dos dispositivos, tendo que criar CVEs especificas para cada dispositivos se tornando inviável. Além disso, algumas técnicas requerem acesso a dispositivos e outras possuem alto custo computacional. Nesse contexto, MANDRAKE detecta vulnerabilidades de IoT a partir da análise de tráfego de rede. O método é simples, eficiente e não requer conhecimento prévio da rede. Assim, o método foi avaliado em dois cenários IoT, sendo um cenário experimental e um cenário orientado a traços, composto por dispositivos IoT enviando e recebendo dados de acordo com sua aplicação. Visto os respectivos cenários o MANDRAKE detectou com até 99% de precisão vulnerabilidades de falta de criptografia em pacotes transmitidos pelos dispositivos. Todavia no cenário experimental os resultados se mostraram melhores com precisão entre 83% a 99% devido se tratar de um conjunto de dados com uma complexidade pequena. Já no cenário orientado a traços a complexidade aumenta, visto a quantidade de dispositivos e trafego gerado, assim tendo entre 53,4% até 98% de precisão. Por fim, o método se mostra eficiente na detecção de dispositivos que transmitem pacotes descriptografados na rede em questão. Dessa forma, a Seção 6.1 apresenta uma discussão geral sobre o método MANDRAKE e seus resultados obtidos. Por fim, a Seção 6.2 discute direções futuras.

6.1 DISCUSSÃO GERAL DA ROTULAGEM

A rotulagem de amostras em classificação supervisionada ainda é um problema bastante discutido na literatura devido a sua complexidade e custo operacional. Nesta dissertação foi utilizada uma forma de rotulação automática seguindo um limiar de entropia. A entropia de *Shannon* muito discutida na literatura se mostrou eficiente na rotulagem de modelos de classificação seguindo vários teste e validações. A entropia por medir a aleatoriedade de um determinado conjunto de caracteres e estimar um valor entre 0 e 10 foi possível encontrar um limiar dinâmico rotulando conjuntos de amostras de tráfego, para identificar dispositivos que geram tráfego vulnerável e criar modelos de aprendizagem de máquina capazes de classificar dispositivos que transmitem tráfego vulnerável.

A rotulagem aconteceu de forma automática onde o tráfego foi extraído e filtrado descartando pacotes com conteúdos vazios, para melhor desempenho do método, sendo assim é atribuído ao cálculo da entropia o *payload* do pacote extraído. Neste caso, o *payload* do pacote caracteriza um conjunto de caracteres ou uma *string*, satisfazendo a entrada para o cálculo da entropia, para medir sua aleatoriedade. Visto isso, devido à entropia estima uma nota entre 0 e 10 e acordo com Dorfinger et al. (2011), notas superiores a 7 indicam corretamente que o *payload* esta criptografo e notas inferiores a 7 indicam que o *payload* esta descriptografado. Portanto, a rotulagem seguiu está métrica, os *payload* que obtinham notas superiores a 7 eram rotulados

como não vulneráveis, ou seja, rótulo 0 e caso contrário *payload* com notas inferiores a 7 eram rotulados como vulneráveis, ou seja, rótulo 1.

6.2 TRABALHOS FUTUROS

Os resultados alcançados e as conclusões acima apresentadas demonstram que este trabalho atingiu os objetivos propostos. Contudo, foi abordada apenas a vulnerabilidade de tráfego sem criptografia, além disso, não foi implementado nenhum método de seleção de características para os classificadores. Diante disso, como trabalhos futuros pretende-se realizar uma análise para detectar novas vulnerabilidades tanto quanto vulnerabilidades conhecidas ou desconhecidas pela literatura, em conjunto realizar técnicas de seleção de características para melhor desempenho dos classificadores utilizados. Ou seja, detectar novas vulnerabilidades com o intuito de criar um modelo capaz de detectar vulnerabilidades de rede em diferentes cenários com desempenho eficaz.

No contexto de análise de tráfego existem muitas possibilidades utilizando essa técnica, desde predição e identificação de ataques a categorização dos dispositivos e classificação de tráfego, entre outros, em redes de comunicação IoT. Visto os estudos da literatura, a categorização de dispositivos pela análise de tráfego é um ponto em aberto devido sua complexidade, levando em conta a heterogeneidade dos dispositivos. Dessa forma, também como trabalhos futuros, pretende-se analisar o contexto de categorização de dispositivo baseado somente na análise de tráfego. Ou seja, identificar os dispositivos (câmera, fechadura eletrônica, sensores), baseado somente na exploração de características extraídas do tráfego em questão.

REFERÊNCIAS

802.15.4, I. S. (2016). Ieee standard for low-rate wireless networks. *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, páginas 1–709.

Bandyopadhyay, D. e Sen, J. (2011). Internet of Things: Applications and challenges in technology and standardization. *Wireless Personal Communications*, 58(1):49–69.

Banks, A. e Gupta, R. (2014). MQTT version 3.1. 1. Relatório técnico, OASIS standard.

Bedhief, I., Kassar, M. e Aguili, T. (2016). Sdn-based architecture challenging the iot heterogeneity. Em *2016 3rd Smart Cloud Networks Systems (SCNS)*, páginas 1–3.

Bhatia, R., Benno, S., Esteban, J., Lakshman, T. V. e Grogan, J. (2019). Unsupervised machine learning for network-centric anomaly detection in iot. Em *Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks*, Big-DAMA '19, página 42–48, New York, NY, USA. Association for Computing Machinery.

BlackYe (2016). Python code examples for calculate entropy. (https://www.programcreek.com/python/?CodeExample=calculate%20entropy). Último acesso Setembro/2021.

Braden, R. (1989). Requirements for internet hosts - communication layers". Em *STD 3, RFC 1122, DOI 10.17487/RFC1122*.

Chaddad, L., Chehab, A. e Kayssi, A. (2021). Opriv: Optimizing privacy protection for network traffic. *Journal of Sensor and Actuator Networks*, 10(3).

Chernis, B. e Verma, R. (2018). Machine learning methods for software vulnerability detection. Em *Proceedings of the Fourth ACM International Workshop on Security and Privacy Analytics*, IWSPA '18, página 31–39, New York, NY, USA. Association for Computing Machinery.

Cisco (2020). Cisco annual internet report (2018–2023) white paper. (https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html). Último Acesso: Agosto de 2021.

da Silva, C. A. G., Ferrari, A. C. K., Osinski, C. e Pelacini, D. A. F. (2021). The behavior of internet traffic for internet services during covid-19 pandemic scenario.

Dorfinger, P., Panholzer, G. e John, W. (2011). Entropy estimation for real-time encrypted traffic identification (short paper). Em Domingo-Pascual, J., Shavitt, Y. e Uhlig, S., editores, *Traffic Monitoring and Analysis*, páginas 164–171, Berlin, Heidelberg. Springer Berlin Heidelberg.

Edgar, T. W. e Manz, D. O. (2017). Chapter 14 - addressing the adversary. Em Edgar, T. W. e Manz, D. O., editores, *Research Methods for Cyber Security*, páginas 345–366. Syngress.

Fang, Z., Fu, H., Gu, T., Qian, Z., Jaeger, T. e Mohapatra, P. (2019). Foresee: A cross-layer vulnerability detection framework for the internet of things. Em *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, páginas 236–244.

Fernandez, F. e Pallis, G. C. (2014). Opportunities and challenges of the internet of things for healthcare: Systems engineering perspective. Em *2014 4th International Conference on Wireless Mobile Communication and Healthcare - Transforming Healthcare Through Innovations in Mobile and Wireless Technologies (MOBIHEALTH)*, páginas 263–266.

Forbes (2021). The Five Biggest Cyber Security Trends In 2022. Disponível em: https://www.forbes.com/sites/bernardmarr/2021/12/17/the-five-biggest-cyber-security-trends-in-2022/?sh=3c70a1594fa3. Acessado em Fevereiro, 2022.

Franco, M., Von der Assen, J., Boillat, L., Killer, C., Rodrigues, B., Scheid, E. J., Granville, L. e Stiller, B. (2021). Secgrid: a visual system for the analysis and ml-based classification of cyberattack traffic. Em *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, páginas 140–147.

Gendreau, A. A. e Moorman, M. (2016). Survey of intrusion detection systems towards an end to end secure internet of things. Em *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, páginas 84–90.

Grieco, G. e Dinaburg, A. (2018). Toward smarter vulnerability discovery using machine learning. Em *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*, AISec '18, página 48–56, New York, NY, USA. Association for Computing Machinery.

Gu, T., Abhishek, A., Fu, H., Zhang, H., Basu, D. e Mohapatra, P. (2020). Towards learning-automation iot attack detection through reinforcement learning. Em *2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, páginas 88–97.

Gu, X., Wu, W., Gu, X., Ling, Z., Yang, M. e Song, A. (2020). Probe request based device identification attack and defense. *Sensors*, 20(16).

Harer, J. A., Kim, L. Y., Russell, R. L., Ozdemir, O., Kosta, L. R., Rangamani, A., Hamilton, L. H., Centeno, G. I., Key, J. R., Ellingwood, P. M., Antelman, E., Mackay, A., McConley, M. W., Opper, J. M., Chin, P. e Lazovich, T. (2018). Automated software vulnerability detection with machine learning.

He, D., Gu, H., Li, T., Du, Y., Wang, X., Zhu, S. e Guizani, N. (2020). Toward hybrid static-dynamic detection of vulnerabilities in iot firmware. *IEEE Network*, páginas 1–6.

Huang, D. Y., Apthorpe, N., Li, F., Acar, G. e Feamster, N. (2020). IoT inspector: Crowdsourcing labeled network traffic from smart home devices at scale. *ACM Interact. Mob. Wearable Ubiquitous Technol.*, 4(2):21.

Husnain, M., Hayat, K., Cambiaso, E., Fayyaz, U. U., Mongelli, M., Akram, H., Ghazanfar Abbas, S. e Shah, G. A. (2022). Preventing mqtt vulnerabilities using iot-enabled intrusion detection system. *Sensors*, 22(2).

ITC (2020). Como a pandemia tem impactado no aumento do tráfego na internet? (`http://itc.com.br/aumento-do-trafego/`). Último acesso Junho/2021.

Iyer, B. e Patil, N. (2018). Iot enabled tracking and monitoring sensor for military applications. *International Journal of System Assurance Engineering and Management*, 9(6):1294–1301.

Jia, X., Li, X. e Gao, Y. (2017). A novel semi-automatic vulnerability detection system for smart home. Em *Proceedings of the International Conference on Big Data and Internet of Thing*, BDIOT2017, página 195–199, New York, NY, USA. Association for Computing Machinery.

Jia, Y., Xiao, Y., Yu, J., Cheng, X., Liang, Z. e Wan, Z. (2018). A novel graph-based mechanism for identifying traffic vulnerabilities in smart home iot. Em *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, páginas 1493–1501.

Koike, D., Ishida, S. e Arakawa, Y. (2021). Called function identification of iot devices by network traffic analysis. Em *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, SAC '21, página 737–743, New York, NY, USA. Association for Computing Machinery.

Lin, G., Wen, S., Han, Q. L., Zhang, J. e Xiang, Y. (2020). Software vulnerability detection using deep neural networks: A survey. *Proceedings of the IEEE*, 108(10):1825–1848.

Luo, J.-Z., Shan, C., Cai, J. e Liu, Y. (2018). Iot application-layer protocol vulnerability detection using reverse engineering. *Symmetry*, 10(11).

Lyu, Q. e Lu, X. (2019). Effective media traffic classification using deep learning. Em *Proceedings of the 2019 3rd International Conference on Compute and Data Analysis*, ICCDA 2019, página 139–146, New York, NY, USA. Association for Computing Machinery.

Ma, C., Du, X. e Cao, L. (2020). Improved knn algorithm for fine-grained classification of encrypted network flow. *Electronics*, 9(2):324.

Maurice, C., Onno, S., Neumann, C., Heen, O. e Francillon, A. (2013). Improving 802.11 fingerprinting of similar devices by cooperative fingerprinting. Em *2013 International Conference on Security and Cryptography (SECRYPT)*, páginas 1–8.

McClean, S. I. (2003). Data mining and knowledge discovery. Em Meyers, R. A., editor, *Encyclopedia of Physical Science and Technology (Third Edition)*, páginas 229–246. Academic Press, New York, third edition edition.

McDaid, A., Furey, E. e Curran, K. (2021). Wireless interference analysis for home iot security vulnerability detection. *International Journal of Wireless Networks and Broadband Technologies (IJWNBT)*, 10(2):55–77.

Medeiros, I., Neves, N. e Correia, M. (2016). Dekant: A static analysis tool that learns to detect web application vulnerabilities. Em *Proceedings of the 25th International Symposium on Software Testing and Analysis*, ISSTA 2016, página 1–11, New York, NY, USA. Association for Computing Machinery.

Montenegro, G., Schumacher, C. e Kushalnagar, N. (2007). IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. Relatório Técnico 4919, IETF.

Mukherjee, L. (2020). The owasp iot top 10 list of vulnerabilities. (`https://sectigostore.com/blog/owasp-iot-top-10-iot-vulnerabilities/`). Último acesso Junho/2021.

MZGroup (2021). Ataques cibernéticos no 1s21. (`https://blog.mzgroup.com/pt-br/ataques-ciberneticos-no-1s21/`). Último acesso Setembro/2021.

Neshenko, N., Bou-Harb, E., Crichigno, J., Kaddoum, G. e Ghani, N. (2019). Demystifying iot security: An exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations. *IEEE Communications Surveys Tutorials*, 21(3):2702–2733.

Niu, W., Zhuo, Z., Zhang, X., Du, X., Yang, G. e Guizani, M. (2019). A heuristic statistical testing based approach for encrypted network traffic identification. *IEEE Transactions on Vehicular Technology*, 68(4):3843–3853.

Nokia (2020). Threat intelligence report 2020. (`https://www.nokia.com/networks/portfolio/cyber-security/threat-intelligence-report-2020/`). Último acesso Agosto/2021.

Orebaugh, A., Ramirez, G., Beale, J. e Wright, J. (2007). *Wireshark & Ethereal Network Protocol Analyzer Toolkit*. Syngress Publishing.

Oser, P., van der Heijden, R. W., Lüders, S. e Kargl, F. (2022). Risk prediction of iot devices based on vulnerability analysis. *ACM Trans. Priv. Secur.*, 25(2).

Pacheco, F., Exposito, E., Gineste, M., Baudoin, C. e Aguilar, J. (2018). Towards the deployment of machine learning solutions in network traffic classification: A systematic survey. *IEEE Commun. Surveys & Tuts,*.

Puhan, Z., Jianxiong, W., Xin, W. e Zehui, W. (2014). Decrypted data detection algorithm based on dynamic dataflow analysis. Em *Proc. of the IEEE CITS*, páginas 1–4, Jeju, Korea (South).

Rezaei, S. e Liu, X. (2019). Deep learning for encrypted traffic classification: An overview. *IEEE Communications Magazine*, 57(5):76–81.

Robyns, P., Bonné, B., Quax, P. e Lamotte, W. (2017). Noncooperative 802.11 mac layer fingerprinting and tracking of mobile devices. *Security and Communication Networks*, 2017:6235484.

Rose, J. R., Swann, M., Bendiab, G., Shiaeles, S. e Kolokotronis, N. (2021). Intrusion detection using network traffic profiling and machine learning for iot. Em *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, páginas 409–415.

Ross, R., McEvilley, M. e Oren, J. (2018). Systems security engineering: Considerations for a multidisciplinary approach in the engineering of trustworthy secure systems.

Sachidananda, V., Bhairav, S. e Elovici, Y. (2020). Over: Overhauling vulnerability detection for iot through an adaptable and automated static analysis framework. SAC '20, página 729–738, New York, NY, USA. Association for Computing Machinery.

Said, O. e Masud, M. (2013). Towards internet of things: Survey and future vision. *International Journal of Computer Networks*, 5:1–17.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(4):623–656.

Shelby, Z., Hartke, K. e Bormann, C. (2014). The Constrained Application Protocol (CoAP). Relatório Técnico 7252, IETF.

Shirey, R. W. (2007). Internet Security Glossary, Version 2. RFC 4949.

Sivanathan, A., Gharakheili, H. H., Loi, F., Radford, A., Wijenayake, C., Vishwanath, A. e Sivaraman, V. (2019). Classifying IoT devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing*, 18(8):1745–1759.

Soltani, R., Goeckel, D., Towsley, D. e Houmansadr, A. (2017). Towards provably invisible network flow fingerprints. Em *2017 51st Asilomar Conference on Signals, Systems, and Computers*, páginas 258–262.

Sonnekalb, T. (2019). Machine-learning supported vulnerability detection in source code. ESEC/FSE 2019, página 1180–1183, New York, NY, USA. Association for Computing Machinery.

Souza, D. (2020). Big data: Grande volume de dados + coleta de dados. (`https://www.linknacional.com.br/blog/big-data-volume-de-dados/`). Último acesso Abril/2020.

Standaert, F.-X. (2010). *Introduction to Side-Channel Attacks*, páginas 27–42. Springer US, Boston, MA.

Staudemeyer, R., Umuhoza, D. e Omlin, C. (2005). Attacker models, traffic analysis and privacy threats in ip networks. página 7.

Steward, J. (2022). A lista definitiva de estatísticas da internet das coisas para 2022. (`https://findstack.com/pt/internet-of-things-statistics/`). Último acesso Março/2022.

Vajapeyam, S. (2014). Understanding shannon's entropy metric for information.

Verma, P. e Sood, S. K. (2018). Fog assisted-iot enabled patient health monitoring in smart homes. *IEEE Internet of Things Journal*, 5(3):1789–1796.

Vulnerabilities, C. e Exposures (2022). National Vulnerability Database. Disponível em: https://cve.mitre.org/. Acessado em Maio, 2022.

Wang, Y., Zhang, Z., Guo, L. e Li, S. (2011). Using entropy to classify traffic more deeply. Em *Proc. of the IEEE VI NAS*, páginas 45–52, Dalian, China.

Xie, W., Jiang, Y., Tang, Y., Ding, N. e Gao, Y. (2017). Vulnerability detection in iot firmware: A survey. Em *2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS)*, páginas 769–772.

Yi, M., Xu, X. e Xu, L. (2019). An intelligent communication warning vulnerability detection algorithm based on iot technology. *IEEE Access*, 7:164803–164814.

Zhang, B. (2020). A software upgrade security analysis method on network traffic classification using deep learning. Em *2020 International Conference on Urban Engineering and Management Science (ICUEMS)*, páginas 568–574.

Zhang, Y., Yang, M., Gu, X., Pan, P. e Ling, Z. (2018). Fingerprinting network device based on traffic analysis in high-speed network environment. Em *2018 Sixth International Conference on Advanced Cloud and Big Data (CBD)*, páginas 249–256.

Zhao, K. e Ge, L. (2013). A survey on the internet of things security. Em *2013 Ninth international conference on computational intelligence and security*, páginas 663–667. IEEE.