DETECÇÃO DE ATAQUES DE NEGAÇÃO DE SERVIÇO EM DISPOSITIVOS IOT

Thiago Viana Dantas

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Rosa Maria Meri Leão

Rio de Janeiro
Julho de 2021

DETECÇÃO DE ATAQUES DE NEGAÇÃO DE SERVIÇO EM DISPOSITIVOS IOT

Thiago Viana Dantas

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientador: Rosa Maria Meri Leão

Aprovada por: Prof. Rosa Maria Meri Leão
Prof. Edmundo Albuquerque de Souza e Silva
Prof. Ana Paula Couto da Silva

RIO DE JANEIRO, RJ – BRASIL
JULHO DE 2021

Dantas, Thiago Viana

Detecção de ataques de negação de serviço em dispositivos IoT/Thiago Viana Dantas. – Rio de Janeiro: UFRJ/COPPE, 2021.

XIX, 68 p.: il.; 29, 7cm.

Orientador: Rosa Maria Meri Leão

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2021.

Referências Bibliográficas: p. 55 – 60.

1. Redes de acesso residencial. 2. Detecção de ataques de negação de serviço em IoT. 3. Aprendizado de máquina. I. Leão, Rosa Maria Meri. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Agradecimentos

À Marinha do Brasil pela oportunidade para a realização deste curso.

Aos meus pais, Marleide e Kennedy que sempre estiveram ao meu lado e que não mediram esforços para me dar todo o suporte para estudar; a minha esposa Fabrícia pelo amor, compreensão, paciência e apoio nos momentos difíceis.

A minha orientadora, prof. Rosa, pela disponibilidade e por todos os ensinamentos e orientações que permitiram o meu crescimento profissional.

Ao prof. Edmundo e a todos os demais professores e alunos do laboratório LAND pelas valiosas contribuições durante todo o desenvolvimento desta dissertação.

Por fim, a todos aqueles que contribuíram de alguma forma para a minha formação acadêmica e para a realização deste trabalho.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

DETECÇÃO DE ATAQUES DE NEGAÇÃO DE SERVIÇO EM DISPOSITIVOS IOT

Thiago Viana Dantas

Julho/2021

Orientador: Rosa Maria Meri Leão

Programa: Engenharia de Sistemas e Computação

A crescente popularização da Internet das Coisas (IoT) e a ampla gama de dispositivos conectados à Internet aumenta os riscos de infecção por *malware*, devido a muitas vulnerabilidades existentes nos IoT. Portanto, é fundamental o desenvolvimento de mecanismos capazes de mitigar os impactos desses ataques. A fim de identificar fluxos de dados maliciosos, diversos trabalhos na literatura utilizaram dados sintéticos ou *datasets* reais muito pequenos para desenvolver modelos baseados principalmente em inspeção profunda de pacotes. O principal objetivo deste trabalho é detectar de maneira eficiente ataques de negação de serviço em dispositivos IoT, usando o mínimo de informação possível. O *dataset* usado nos experimentos possui dados de tráfego real coletados de mais de 800 clientes de um ISP e tráfego de ataques gerado em laboratório utilizando o código fonte das *botnets* BASHLITE e Mirai. Inicialmente foi desenvolvida uma metodologia para identificação dos equipamentos existentes no *dataset* e em seguida foi proposto um *framework* para detecção de ataques. O *framework* tem dois elementos principais: algoritmo para seleção de *features* e modelo de aprendizado de máquina supervisionado. Os resultados obtidos mostraram que *Random Forest* foi o algoritmo de melhor desempenho para identificar a presença de ataques em dispositivos IoT, apresentando valores de *precision* e *recall* superiores à 0,9776 e 0,9930, respectivamente. Mostra-se também que a performance da *Random Forest* para detecção de tráfego malicioso foi melhor do que políticas baseadas em *threshold*, o que confirma que a detecção de ataques em IoT é uma tarefa não trivial. Assim, a abordagem simples proposta neste trabalho demonstrou ser capaz de detectar diferentes vetores de ataques de maneira bastante eficiente.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

DETECTION OF DENIAL OF SERVICE ATTACKS ON IOT DEVICES

Thiago Viana Dantas

July/2021

Advisor: Rosa Maria Meri Leão

Department: Systems Engineering and Computer Science

The increasing popularization of the Internet of Things (IoT) and the wide range of devices connected to the Internet increases the risk of malware infection, due to the many vulnerabilities of the IoT. Therefore, it is essential to develop mechanisms capable of mitigating the impacts of these attacks. In order to identify malicious data flows, several studies in the literature have used synthetic data or very small real datasets to develop models based primarily on Deep Packet Inspection (DPI). The main objective of this work is to efficiently detect Denial of Service attacks (DoS) on IoT devices, using as little information as possible. The dataset used in the experiments has real traffic data collected from more than 800 customers from an ISP and attack traffic generated in the laboratory using the source code of the BASHLITE and Mirai botnets. Initially, a methodology for identifying equipment in the dataset was developed and then a framework for attack detection was proposed. The framework has two key elements: algorithm for feature selection and supervised machine learning models. The results obtained showed that Random Forest was the algorithm with the best performance to identify the presence of attacks on IoT devices, with precision and recall values higher than 0.9776 and 0.9930, respectively. It is also shown that Random Forest's performance for malicious traffic detection was better than threshold-based policies, which confirms that IoT attack detection is a non-trivial task. Thus, the simple approach proposed in this work demonstrated to be able to detect different attack vectors in a very efficient way.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

$\emptyset$	Conjunto vazio, p. 29
$\gg$	Muito maior que, p. 29
$\mathcal{N}$	Distribuição Gaussiana, p. 16

Lista de Abreviaturas

ACK	*Acknowledge*, p. 12
APP	Aplicativo, p. 18
AWS	*Amazon Web Services*, p. 2
C2	Servidor de Comando e Controle, p. 2
CDF	*Cumulative Distribution Function*, p. 34
CNN	*Convolutional Neural Network*, p. 6
COPPE	Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa em Engenharia, p. 9
DDoS	*Distributed Denial of Service*, p. 2
DNN	*Deep Neural Network*, p. 6
DNS	*Domain Name Service*, p. 13
DNS	*Domain Name System*, p. 8
DPI	*Deep Packet Inspection*, p. vi
DT	*Decision Tree*, p. 30
DoS	*Denial of Service*, p. vi
ETH	*Ethernet*, p. 13
FN	*False Negatives*, p. 29
FPR	*False Positive Rate*, p. 46
FP	*False Positives*, p. 29
GNB	*Gaussian Naive Bayes*, p. 31
GRE	*Generic Routing Encapsulation*, p. 13

HTTP	*Hypertext Transfer Protocol*, p. 12
IDC	*International Data Corporation*, p. 1
IEEE	*Institute of Electrical and Electronics Engineers*, p. 18
IP	*Internet Protocol*, p. 11
ISP	*Internet Service Provider*, p. vi
IoT	*Internet of Things*, p. vi
KNN	*K-Nearest Neighbor*, p. 30
LOF	*Local Outlier Factor*, p. 6
LR	*Logistic Regression*, p. 31
LSTM	*Long Short Term Memory*, p. 7
MAC	*Media Access Control*, p. 9
MLP	*Multi-layer Perceptron*, p. 7
Mbps	Megabits por segundo, p. 16
NB	*Naive Bayes*, p. 6
NTP	*Network Time Protocol*, p. 8
OUI	*Organizationally Unique Identifier*, p. 18
PCA	*Principle Component Analysis*, p. 8
RF	*Random Forest*, p. 30
SDN	*Software–Defined Networking*, p. 4
SSH	*Secure Socket Shell*, p. 12
STOMP	*Simple Text Oriented Messaging Protocol*, p. 13
SVM	*Support Vector Machine*, p. 6
SYN	*Synchronize*, p. 12
TCP	*Transmission Control Protocol*, p. 12
TN	*True Negatives*, p. 29

TP	*True Positives*, p. 29
Tbps	Terabits por segundo, p. 2
UDP	*User Datagram Protocol*, p. 12
VSE	*Valve Source Engine*, p. 13
WLAN	*Wireless Local Area Network*, p. 8
Web	*World Wide Web*, p. 7
bps	bits por segundo, p. 28
pps	pacotes por segundo, p. 16

Capítulo 1

Introdução

O termo Internet das Coisas surgiu em 1999 durante uma apresentação realizada por Kevin Ashton na Procter & Gamble (P&G) [3], porém só começou a ganhar popularidade a partir de 2010. Desde então, soluções IoT vem ganhando cada vez mais mercado e apesar da pandemia da COVID-19, em 2020 foram investidos $742 bilhões em IoT, o que representa um crescimento de 8,2% em relação ao ano anterior [4]. Com todo esse investimento, os dispositivos IoT passaram a representar pela primeira vez na história a maior parte dos equipamentos ativos na rede (54% do total ou 11,7 bilhões de equipamentos)[5]. A *International Data Corporation* (IDC) estima que em 2025 três em cada quatro equipamentos conectados à Internet serão IoT, representando um total de 41,8 bilhões de dispositivos, os quais serão responsáveis por produzir 73,1 ZB (Zettabytes) de dados [6].

De maneira geral, Internet das Coisas se refere a objetos do mundo real (*things*) que estão conectados à Internet. Essas "coisas" são chamadas de objetos inteligentes (*smart objects*), caso sejam capazes de interagir com o ambiente sem a intervenção humana, através de sensores, atuadores e softwares embarcados [7].

Soluções IoT são cada vez mais populares pois facilitam a vida humana em ambientes residenciais, industriais e empresariais. No entanto, muito desses dispositivos são utilizados com suas senhas padrão de fábrica ou não recebem atualizações de segurança para corrigir suas vulnerabilidades durante o seu ciclo de vida. Dessa forma, esses equipamentos são facilmente infectados por *malwares*, sendo muito empregados como plataformas para ataques cibernéticos [1, 8].

Em particular, ataques de negação de serviço distribuídos (DDoS – *Distributed Denial of Service*) tem sido cada vez mais frequentes e segundo previsão da Cisco, em 2023 ocorrerá mais de 15 milhões desse tipo de ataque, o que representa aproximadamente o dobro do registrado em 2018 [9]. Em 2017, a Google sofreu o maior ataque DDoS conhecido até o momento, alcançando picos de 2,5 Tbps (Terabits por segundo), porém esses dados só foram divulgados publicamente no segundo semestre de 2020 [10]. Outros ataques de grandes volumes foram registrados na AWS (*Ama-*

zon Web Services) em 2020 e no serviço do Github em 2018, os quais atingiram picos de 2,3 Tbps [11] e 1,35 Tbps [12], respectivamente.

Segundo a Akamai, "em 2020, houve 4.542.524 solicitações bloqueadas para a infraestrutura de C2 (Servidor de Comando e Controle) de botnets, indicando um possível comprometimento" [13]. Esse número é alarmante e comprova que *hackers* poderão utilizar dispositivos IoT para criar *botnets* de larga escala capazes de gerar ataques DDoS de grande porte [8].

A fim de minimizar os impactos desses ataques é fundamental que os mecanismos de detecção atuem próximo à sua origem, ou seja, na borda da rede [14]. Assim sendo, Provedores de Serviço de Internet precisam utilizar ferramentas para identificar quais dispositivos dos seus clientes estão infectados e bloquear o tráfego gerado por eles. A solução adotada pelos ISPs não pode ser específica a determinadas assinaturas de ataques, pois novas *botnets* são criadas ou evoluções são desenvolvidas a partir de *botnets* conhecidas [1, 10, 14].

O objetivo deste trabalho é detectar ataques de negação de serviço em dispositivos IoT de maneira simples, a fim de permitir que a solução possa ser implementada pelos provedores de serviço de Internet em um dos seus servidores de processamento de dados e caso seja detectado algum equipamento malicioso, enviar o comando de bloqueio para os roteadores.

Este trabalho é baseado em MENDONÇA *et al.* [15], no qual os autores propõem um método para detecção de ataques DDoS em roteadores domésticos usando um algoritmo de aprendizado de máquina supervisionado treinado com doze *features*, as quais foram obtidas a partir do contador de bytes e pacotes do tráfego total das residências a cada minuto.

Assim sendo, neste trabalho é proposta uma metodologia que utiliza o contador de bytes e pacotes gerado por cada equipamento individualmente para treinar um modelo de aprendizado de máquina supervisionado, ou seja, a solução proposta cria um modelo de detecção específico para cada categoria de equipamento IoT. Dessa forma, o ISP consegue identificar o dispositivo infectado e bloquear apenas o seu tráfego, ao invés de impedir o acesso total do seu cliente à Internet, como ocorreria na abordagem de MENDONÇA *et al.* [15].

As contribuições deste trabalho são resumidas a seguir:

- *Utilização de dados de tráfego de dispositivos IoT reais.* Neste trabalho foram utilizados dados de mais de 800 clientes de um ISP, durante um período de cinco meses. Dessa forma, o *dataset* utilizado representa o tráfego real dos equipamentos.

- *Metodologia para identificação das categorias e fabricantes dos dispositivos IoT.* Foi proposta uma abordagem para classificação dos equipamentos IoT

usando apenas informações do endereço MAC e do *hostname*. Os resultados indicaram que esse método possibilitou a identificação da maior parte dos equipamentos coletados.

- *Framework para detecção de ataques de negação de serviço em IoT*. Os resultados mostraram que o *framework* baseado em estatísticas relacionadas ao contador de bytes e pacotes a cada minuto dos dispositivos IoT individualmente foi capaz de detectar diferentes vetores de ataques com eficiência, apresentando *recall* superior a 0,99 para todas as categorias estudadas e taxa de alarme falso de 0,0026 para as câmeras e 10^{-5} para as demais categorias avaliadas. Adicionalmente, também foi proposto um mecanismo para seleção das melhores *features* para cada categoria de equipamento IoT. Esse procedimento adicional permitiu reduzir a complexidade do modelo e não prejudicou o seu desempenho. Experimentos adicionais mostraram a eficiência do modelo na detecção de ataques desconhecidos e desempenho superior ao de políticas baseadas em *threshold*.

- *Tolerância à heterogeneidade*. Dependendo de suas funcionalidades, aplicações ou padrões de utilização, dispositivos IoT podem apresentar perfis de tráfegos muito distintos. Para lidar com a heterogeneidade desses equipamentos, foi proposta uma metodologia que cria um modelo de detecção de ataques DoS específico para cada categoria de dispositivo IoT. Os resultados dos classificadores mostraram que a categorização utilizada conseguiu mapear os padrões de tráfego natural gerado por cada tipo de equipamento, apresentando altas taxas de detecção em todas as categorias estudadas. Além disso, essa abordagem permite identificar exatamente qual dispositivo está gerando o ataque DoS.

No que se segue, o trabalho está organizado da seguinte forma. No Capítulo 2 é realizada uma revisão da literatura relacionada a detecção de ataques DDoS em dispositivos IoT. O Capítulo 3 descreve como foi feita a coleta dos dados e a geração do tráfego de diferentes vetores de ataques de negação de serviço. O Capítulo 4 apresenta a metodologia proposta para classificação dos tipos de equipamentos coletados e a criação dos modelos de detecção de ataques em cada categoria de dispositivo. No capítulo 5 é realizada a avaliação de desempenho dos modelos e a comparação da sua performance com políticas baseadas em *threshold*. Por fim, no Capítulo 6 são apresentadas as conclusões e os trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

Em 2016, a *botnet* de IoT Mirai utilizou 24.000 dispositivos para causar um ataque DDoS que atingiu pico de 623 Gbps. Nesse mesmo ano foi registrado outro ataque a partir de 210.000 dispositivos (em sua maioria *Digital Video Recorders* – DVRs, câmeras de vigilância e roteadores domésticos) [16]. Embora seja uma ameaça antiga, ataques DDoS continuam ocorrendo. De julho de 2019 à junho de 2020, a Akamai registrou 5624 ataques dessa natureza, os quais atingiram diversos setores (jogos, serviços financeiros, comércio, serviços corporativos, setor público, entre outros) [17].

Botnets de IoT estão constantemente evoluindo e aliado a isso, seus atacantes vem sempre inovando nas estratégias adotadas. Por esse motivo, desenvolver mecanismos de defesa contra esses ataques continua sendo um grande desafio. Com o intuito de evitar a detecção dos *bots*, os *hackers* têm realizado ataques intermitentes [18] e inundado suas vítimas com volumes de tráfego que se assemelham ao gerado por usuários legítimos [19].

O uso cada vez mais frequente de soluções baseadas em IoT associado a potencialidade desses equipamentos serem empregados em grandes *botnets*, devido aos riscos e vulnerabilidades ainda existentes nesse tipo de dispositivo, tem motivado diversos estudos nessa área.

Nesse contexto, muitas soluções tem sido propostas (ver Tabela 2.1). Algumas delas objetivam detectar ataques antes que eles ocorram [20–22], enquanto outras buscam identificar o tráfego malicioso durante a execução do ataque [23–31].

ÖZÇELIK *et al.* [20] propuseram um mecanismo para identificar um *host* infectado por um *malware* durante a fase de varredura utilizando uma arquitetura baseada em SDN (*Software–Defined Networking*). No trabalho foram emulados sete nós IoT via *software*, os quais foram infectados com o *malware* Mirai. Os resultados mostraram que o modelo demorou em média 6,02 segundos para detectar um nó malicioso. Adicionalmente, também foi observado uma degradação no *throughput* dos *nós* benignos durante a execução do ataque de um nó malicioso conectado na mesma rede.

Tabela 2.1: Trabalhos anteriores para detecção de ataques em IoT.

Ref.	***Dataset***	**número *features***	**Modelo proposto**	**Resultado**
[20]	Simulado via *Software* (7 nós IoT)	–	ECESID[1](TRW-CB[2]+RL[3])	– [4]
[21]	CTU-13 CICDDoS2019 ISOT HTTP *Botnet* CAIDA DDoS *Attack* 2007 CSE-CIC-IDS2018	20	KMeans *Random Forest* Regressão Logística *Gradient Boosting*	*recall* [5]
[22]	CCC *dataset* 08, 09, 10 CCC *pratice dataset 13* dados benignos coletados em laboratório	55	*Random Forest*	*recall* = 0,99
[23]	N-BaIoT – Simulado em laboratório (9 IoT)	115	*Deep autoencoder*	TPR = 1 FPR = 0,007
[24]	N-BaIoT	115	*Deep Neural Network*	acurácia 0,99
[25]	N-BaIoT	115	Árvore de Decisão *Random Forest* CNN[6]	*f1-score* = 1 *f1-score* = 1 *f1-score* = 0,99
[26]	NSL-KDD *dataset*	9 41	*Deep Neural Network*	acurácia = 0,93 acurácia = 0,97
[27]	Mirai *dataset* da universidade Robert Gordon USTC-TFC 2016	–	CNN + *deep autoencoder*	acurácia = 0,99 acurácia = 0,99
[28]	CICIDS2017	–	CNN + LSTM[7]	acurácia = 0,97 *precision* = 0,97 *recall* = 0,99
[29]	Simulado em laboratório (3 IoT)	11	*Random Forest* *K-Nearest Neighbors* *Support Vector Machine* *Árvore de decisão* Rede neural	acurácia = 0,99 (todos os modelos)
[30]	Simulado em laboratório (3 IoT)	–	*Support Vector Machine* *Random Forest* Regressão Logística *Árvore de decisão*	acurácia = 0,98 acurácia = 0,99 acurácia = 0,97 acurácia = 0,98
[31]	Simulado em laboratório (10 IoT)	–	SDN[8]	acurácia = 0,94

A partir da premissa que os *bots* trocam informações com o Servidor de Comando e Controle antes da execução do ataque, NEIRA *et al.* [21] propõem o sistema ANTE, o qual utiliza um algoritmo de aprendizado de máquina treinado a partir de 20 *features* extraídas do fluxo de pacotes (p. ex. tamanho dos pacotes e frequência de envio e recebimento de pacotes). Para treinamento do modelo foram utilizados os dados coletados 2 minutos antes do início do ataque. Por sua vez, para avaliação do sistema ANTE foram usados dados gerados no minuto anterior a execução do ataque. Adicionalmente, os autores também testaram seu modelo para detectar ataques em execução. Foram avaliados os algoritmos *KMeans*, Regressão Logística, *Gradient Boosting* e *Random Forest* em diferentes bases de dados. Os resultados demonstraram que não houve um modelo vencedor em todos os cenários avaliados.

Por sua vez, LU *et al.* [22] sugeriram uma metodologia para detectar sessões do Servidor de Comando e Controle utilizando a *Random Forest* como classificador. A partir dos pacotes trocados entre dois endereços IP foram calculados 55 *features*

[1] *Edge-Centric Software-Defined IoT Defense*
[2] *Threshold Random Walk with Credit Based Rate Limiting*
[3] *Rate Limiting*
[4] Foi apresentado o tempo médio de detecção e a degradação no *throughput* dos nós benignos.
[5] Não houve um modelo vencedor em todos os *datasets*. O *recall* variou entre 0,83 e 1.
[6] *Convolutional Neural Network*
[7] *Long Short Term Memory*
[8] *Software-Defined Networking*

para treinamento do modelo. Para avaliação do método foram utilizadas as bases de dados CCC *dataset* (08-10) e CCC *practice dataset* 13 (tráfego malicioso) e dados obtidos em laboratório (tráfego benigno). Os resultados experimentais mostraram que o classificador proposto apresentou melhor performance do que o SVM (*Support Vector Machine)* e NB (*Naive Bayes*).

Abordagens que identificam *hosts* infectados durante a fase de escaneamento ou troca de informações com o Servidor de Comando e Controle são vantajosas pois impedem que o ataque ocorra. No entanto, essas metodologias são totalmente dependentes dos protocolos de varredura e comunicação com o Servidor de Comando e Controle adotados por uma determinada *botnet*. A fim de evitar essas especificidades, este trabalho foca na detecção de tráfego malicioso durante o lançamento do ataque. A identificação de um IoT infectado durante um ataque pode não ser capaz de interromper de imediato o ataque em curso, mas é importante pois pode evitar ataques futuros a partir deste dispositivo.

MEIDAN *et al.* [23] sugeriram utilizar *deep autoencoder* para detectar anomalias em dispositivos IoT. Para alcançar esse objetivo foi proposto o treinamento de um modelo para cada equipamento, usando apenas o tráfego benigno de cada um deles. O *dataset* utilizado nesse trabalho (N-BaIoT) possui dados de tráfego sem ataque e com ataque coletados em laboratório de nove equipamentos IoT, são eles: campainha eletrônica (2), termostato (1), monitor de bebê (1), câmera de segurança (4) e *webcam* (1). Após a coleta do tráfego benigno, os dispositivos foram infectados a partir do código fonte das *botnets* Mirai e BASHLITE. Para cada equipamento foram calculadas 23 *features* estatísticas para cinco janelas de tempo diferentes, totalizando 115 *features*. Os resultados experimentais mostraram maior acurácia do *autoencoder* em relação aos algoritmos *Local Outlier Factor* (LOF), One-Class SVM e *Isolation Forest*.

O *dataset* N-BaIoT [23] também foi utilizado por HOLBROOK e ALAMANIOTIS [24] e KIM *et al.* [25]. Ambos os trabalhos propuseram um modelo específico para cada dispositivo. HOLBROOK e ALAMANIOTIS [24] propuseram identificar anomalias no tráfego através de *Deep Neural Network* (DNN). Os resultados indicaram uma melhor performance da DNN quando comparado aos algoritmos SVM e *Random Forest*. Por sua vez, KIM *et al.* [25] propuseram detectar ataques DDoS utilizando um classificador binário e um classificador multiclasse. O primeiro deles é responsável por detectar a presença do tráfego malicioso, enquanto o segundo identifica o vetor de ataque empregado. Foi avaliado o desempenho de cinco algoritmos de *Machine Learning* e três de *Deep Learning*. Os resultados experimentais mostraram melhor performance dos algoritmos Árvore de Decisão, *Random Forest* e *Convolutional Neural Network* (CNN).

Destaca-se que o classificador multiclasse proposto por KIM *et al.* [25] não tem

muita utilidade no mundo real, uma vez que o modelo não será capaz de identificar vetores de ataques desconhecidos durante o treinamento do classificador, ou seja, ineficaz para detectar novas *botnets*.

Outras abordagens baseadas em *deep learning* também foram propostas por AMARASINGHE *et al.* [26], HWANG *et al.* [27] e ROOPAK *et al.* [28]. AMARASINGHE *et al.* [26] sugeriram o algoritmo DNN treinado a partir de 41 *features* incluindo informações do cabeçalho de pacotes para detecção de anomalias em IoT. O modelo foi treinado e avaliado utilizando o *dataset* KDD-NSL. Por sua vez, HWANG *et al.* [27] propuseram o modelo baseado em CNN + *autoencoder* para identificar fluxos de dados maliciosos. Para avaliação da metodologia foram utilizados o Mirai *dataset* da universidade Robert Gordon, USTC-TFC 2016 e dados de ataques Mirai coletados em laboratório. Por fim, ROOPAK *et al.* [28] avaliou o emprego dos modelos MLP (*Multi-layer Perceptron*), 1d-CNN, LSTM (*Long Short Term Memory*), CNN+LSTM para identificar ataques DDoS. No trabalho foi utilizado o *dataset* CICIDS2017 e o algoritmo CNN+LSTM foi o que apresentou melhor desempenho.

Focado em especificidades de IoT como por exemplo, interação com um número limitado de servidores e padrões periódicos para envio de pacotes, DOSHI *et al.* [29] definiram um conjunto de *features* (11 no total) capazes de mapear esse comportamento para treinar modelos de aprendizado de máquina com o intuito de identificar pacotes maliciosos (gerados por ataques DoS). Os dados sem ataque empregado no trabalho foram gerados em laboratório utilizando três dispositivos IoT (monitor de pressão sanguínea, câmera e interruptor inteligente). Adicionalmente, uma máquina virtual Linux (executando o código da *botnet* Mirai) realizou ataques DoS em um *Raspberri Pi* que estava configurado como um servidor Web Apache. Por fim, esse tráfego foi agregado ao tráfego benigno para gerar o *dataset* com ataque. Foram treinados cinco modelos de aprendizado de máquina e os experimentos mostraram uma melhor performance da *Random Forest* em relação aos outros quatro classificadores analisados.

CHAUDHARY e GUPTA [30] também utilizaram características específicas de IoT, como por exemplo, comunicação com um número limitado de servidores, para desenvolvimento dos seus modelos de detecção de pacotes maliciosos. Para treinamento dos classificadores foram coletados em laboratório dados de tráfego benigno de três equipamentos IoT (sensor, câmera e interruptor inteligente) e tráfego de ataque gerado através das ferramentas hping3 e Gondeneye. Algumas das *features* utilizadas nesse trabalho foram endereço IP e número da porta da origem, endereço IP e número da porta de destino, tempo de comunicação, tamanho do pacote, intervalo de tempo entre pacotes e número de solicitações feitas em diferentes servidores. Assim como em DOSHI *et al.* [29], a *Random Forest* foi o algoritmo de melhor desempenho. No entanto, essa simulação não considerou que o tráfego gerado por um

dispositivo IoT infectado possui tanto o seu tráfego natural quanto o de ataque.

Divergindo um pouco das abordagens anteriores, HAMZA *et al.* [31] mapearam os padrões de tráfego esperado de cada equipamento (comunicação com o servidor do fabricante e com serviços NTP – *Network Time Protocol*, consultas DNS – *Domain Name System*, entre outras) através dos traces dos pacotes e sugeriram monitorar o comportamento de cada dispositivo IoT na rede utilizando SDN e um algoritmo de aprendizado de máquina. A partir desses perfis foi treinado um modelo de aprendizado de máquina para cada dispositivo. Dessa forma, o algoritmo deve identificar desvios do fluxo analisado com o padrão esperado dos equipamentos. Adicionalmente os autores utilizaram o PCA (*Principle Component Analysis*) para reduzir o custo computacional do modelo. Os resultados experimentais indicaram que os algoritmos não supervisionados conseguiram melhores resultados para detectar ataques volumétricos nos dispositivos.

Trabalhos anteriores utilizaram em sua grande maioria informações do cabeçalho dos pacotes ou *payloads* obtidos de um conjunto de dados simulados em laboratório ou *datasets* pequenos para desenvolvimento dos modelos de detecção de ataques. Neste trabalho propomos uma abordagem simples que utiliza uma quantidade pequena de *features*, calculadas a partir do contator de bytes e pacotes trafegados na WLAN (*Wireless Local Area Network*) dos clientes de um ISP. Como grande diferencial empregamos no treinamento e avaliação dos modelos de detecção uma grande quantidade de dados reais coletados dos clientes de um provedor de serviço de Internet de médio porte. Adicionalmente, nossa metodologia utiliza um classificador para cada categoria de dispositivo, o que permite uma maior tolerância à heterogeneidade dos equipamentos IoT.

Capítulo 3

Ambiente de coleta de dados

Para a realização deste trabalho foram utilizados dados de tráfego real de centenas de dispositivos de moradores de Nova Friburgo-RJ e dados de ataques de negação de serviço de duas *botnets* de IoT obtidos através de experimentos realizados em laboratório.

3.1 Coleta de dados de tráfego real

Por meio de uma parceria da COPPE/UFRJ com a *startup* Anlix [32] e um Provedor de Serviço de Internet de médio porte [33] foram obtidas séries temporais do tráfego real de mais de vinte mil dispositivos pertencentes à vários clientes do ISP, no período de fevereiro à junho de 2020, totalizando mais de 290 milhões de séries temporais.

Os roteadores dos usuários do provedor rodam o *firmware OpenWrt* [34] e um *software* de código aberto desenvolvido pela Anlix [32], o qual é responsável por coletar a cada minuto, dados da WLAN (p. ex. contador de bytes e pacotes) de todos os equipamentos conectados à sua rede Wi-Fi. Posteriormente os dados são enviados para um servidor, conforme ilustrado na Figura 3.1. Destaca-se que não é possível inferir o conteúdo acessado pelos clientes do ISP, pois não são coletadas informações dos *payloads* ou cabeçalho dos pacotes.

3.2 Geração de ataques de negação de serviço

O *dataset* de tráfego de ataques DoS foram obtidos de MENDONÇA *et al.* [15], o qual coletou dados de tráfego de ataques a partir de simulações realizadas em laboratório utilizando o código-fonte das *botnets* de IoT BASHLITE e Mirai.

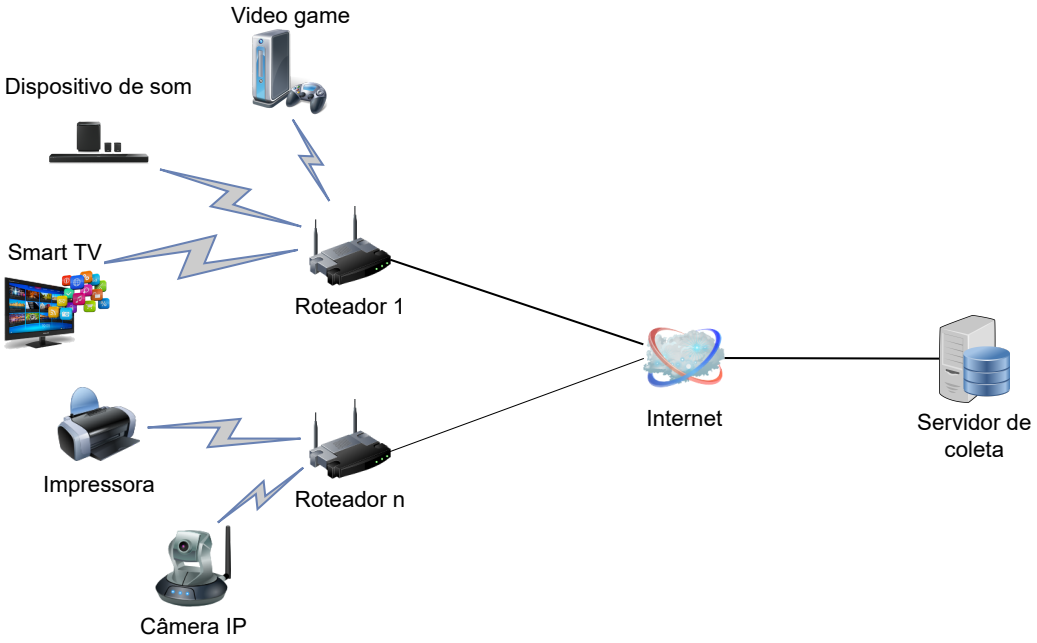

Figura 3.1: Infraestrutura de coleta dos dados.

3.2.1 Ataques de negação de serviço

Ataque de negação de serviço (DoS) é um ataque cibernético no qual o atacante sobrecarrega os recursos do seu alvo, tornando-o indisponível para usuários legítimos. Nesse tipo de ataque, um computador ou dispositivo IoT é infectado com um *malware* e passa a ser controlado remotamente pelo *hacker*. O equipamento contaminado também é chamado de *bot* ou zumbi.

No momento desejado o atacante envia o comando para o *bot* efetuar o ataque, o qual consiste no envio de uma grande quantidade de pacotes que superam a capacidade de processamento da máquina de destino. Dessa forma, as solicitações dos usuários reais não são mais respondidas pelo servidor [35].

Por sua vez, ataques de negação de serviço distribuído (DDoS) são ataques DoS sincronizados, ou seja, proveniente de múltiplas fontes para um mesmo alvo [20]. Essa rede formada por diversos *bots* conectados pela Internet e controladas pelo mesmo atacante é chamada de *botnet* [36].

3.2.2 Visão geral das *botnets* de IoT BASHLITE e Mirai

Botnets de IoT são geralmente estruturadas em seis componentes (*Bots*, Varredores, Invasores, *Servidores de Comando e Controle* (C2s), Servidor de distribuição do *Malware* e Servidor de Relatórios) [37]. Para fins de redundância ou escalabilidade é possível que vários *bots* executem o mesmo papel. Além disso, também pode ocorrer de um equipamento da rede acumular diversas funções [37, 38].

Os *bots* são os dispositivos infectados pelo *malware* e o agente responsável por efetuar o ataque DoS quando determinado pelos C2s. Já os varredores são os componentes encarregados de procurar dispositivos IoT vulneráveis na Internet. Por fim, os invasores são os elementos incubidos de logar nos equipamentos expostos e comandar o *download* do *malware* pelo dispositivo IoT [38].

O elemento responsável por todo o gerenciamento da *botnet* são os Servidores de Comando e Controle. É através dele que o *hacker* determina quando os *bots* irão atacar e qual o endereço IP e porta da vítima serão utilizadas. Por sua vez, o Servidor de distribuição de *Malware* hospeda o código fonte do *malware*. Por último, o Servidor de Relatórios possui a atribuição de armazenar várias informações da *botnet*, tais como: endereço IP (*Internet Protocol*), usuário e senha dos dispositivos vulneráveis e *bots* ativos na *botnet* [37].

A Figura 3.2 apresenta a estrutura operacional de uma *botnet* de IoT BASHLITE e Mirai. Encontra-se destacado em azul (seta pontilhada) a funcionalidade de varredura de potenciais *bots*, em vermelho (seta tracejada) o processo de infecção de novos dispositivos IoT com o *malware* e em verde (seta sólida) a parte relativa ao comando e controle da *botnet*.

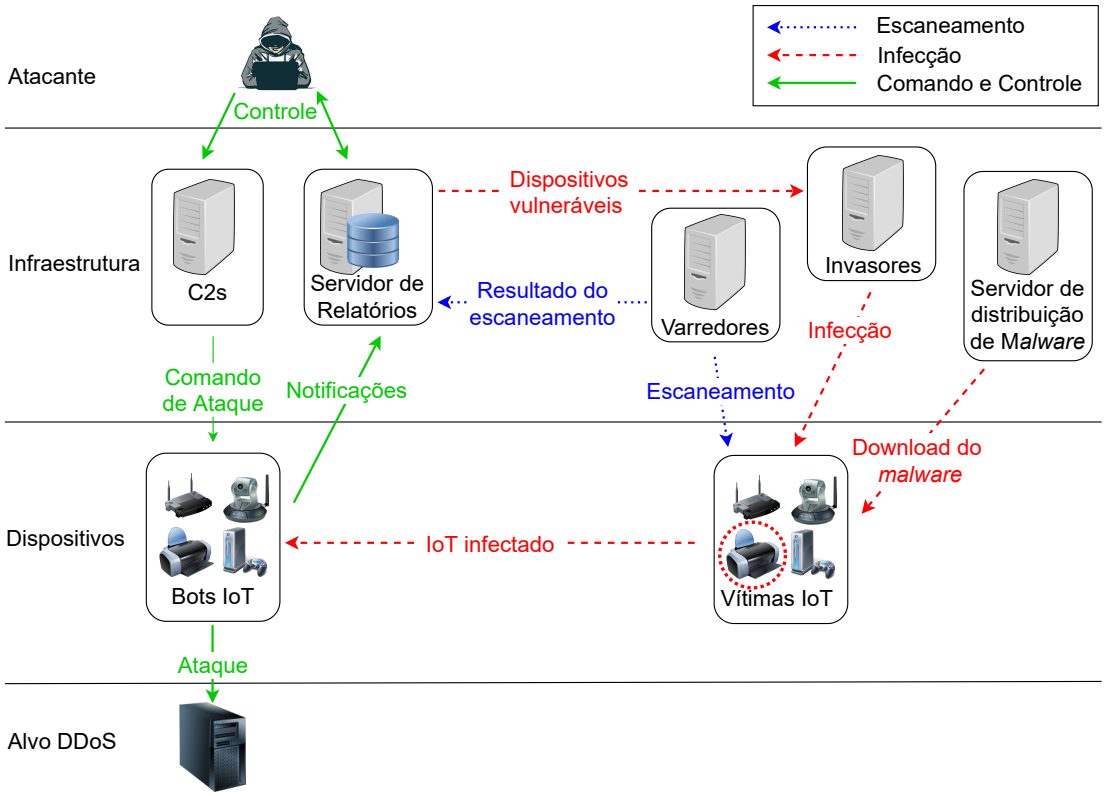

Figura 3.2: Estrutura operacional de uma *botnet* de IoT BASHLITE e Mirai. Adaptado de [1, 2].

Inicialmente os varredores realizam ataques de força bruta em endereços IP e por-

tas aleatórias em busca de dispositivos vulneráveis na rede. Caso seja encontrado um equipamento atacável, o varredor envia para o Servidor de Relatórios várias informações da vítima, tais como, endereço IP, arquitetura de *hardware*, porta aberta, usuário e senha. Esse procedimento de varredura é continuamente realizado, pois possibilita que a *botnet* continue crescendo, podendo realizar um ataque sincronizado de maior porte. Destaca-se que o *malware* Mirai não possui um componente dedicado a essa função, no entanto, os *bots* suprem essa ausência e são os responsáveis por realizar essa operação [37].

A partir das informações dos dispositivos vulneráveis armazenadas no Servidor de Relatórios os invasores iniciam o processo de infecção. Durante esse procedimento é realizado o login e a instrução para a vítima realizar o download e execução do *malware*. Uma vez concluída essa etapa, o *bot* notifica o Servidor de Relatórios sobre a sua situação ativa na *botnet* e fica aguardando instruções dos C2s. Como forma de proteção contra outros *malwares*, após a infecção ser realizada o arquivo binário é excluído e as portas dos serviços SSH (*Secure Socket Shell*) e telnet são fechadas [1].

Por fim, no momento desejado o *hacker* envia o comando de ataque para o Servidor de Comando e Controle que por sua vez repassa essa instrução para os *bots*, os quais efetivamente realizam o ataque DoS no alvo desejado. Juntamente com a ordem de ataque o *hacker* deve informar diversos parâmetros, tais como: endereço IP e porta da vítima, quais *bots* serão utilizados no ataque, tipo e duração do ataque.

BASHLITE

A *botnet* BASHLITE é um *malware* que realiza ataques de negação de serviço a partir de dispositivos IoT de diferentes arquiteturas Linux [37, 39].

O processo de infecção desse *malware* consiste na tentativa de conexão em endereços IPs aleatórios, através do serviço telnet na porta 23/TCP (*Transmission Control Protocol*), utilizando uma combinação de 6 usuários e 14 senhas genéricas pré-definidas no código-fonte do cliente [39, 40]. A Figura 3.1 mostra o dicionário de usuários e senhas usados pelo *malware* BASHLITE.

O BASHLITE pode efetuar ataques do tipo volumétricos, de exaustão de estado e da camada de aplicação. Destacam-se como mais recorrentes o UDP (*User Datagram Protocol*) *flood*, TCP SYN (*Synchronize*) *flood*, TCP ACK (*Acknowledge*) *flood* e HTTP (*Hypertext Transfer Protocol*) *flood* [37].

Mirai

A *botnet* Mirai é considerada um aperfeiçoamento da BASHLITE, tendo sido criada a partir do seu código-fonte [1, 2]. Assim como o BASHLITE, o Mirai também é capaz de infectar múltiplas arquiteturas Linux utilizando um dicionário de credenciais para realizar tentativas de acesso em vítimas aleatórias. No entanto, o Mirai utiliza 61 pares pré-definidos de usuário e senha ao invés de uma combinação deles, conforme apresentando na Tabela 3.2. Outrossim, a tentativa de conexão é realizada tanto na porta 23/TCP quanto na porta 2323/TCP [8, 37, 40].

Tabela 3.1: Dicionário de usuários e senhas do *malware* BASHLITE.

Usuário	**Senha**
root	root
(vazio)	(vazia)
admin	toor
user	admin
login	user
guest	guest
	login
	changeme
	1234
	12345
	123456
	default
	pass
	password

O Mirai é capaz de efetuar os seguintes tipos de ataques: HTTP *flood*, UDP-PLAIN *flood*, UDP *flood*, TCP ACK *flood*, TCP SYN *flood*, GRE-IP (*Generic Routing Encapsulation-IP*) *flood*, ACK-STOMP (*ACK-Simple Text Oriented Messaging Protocol*) *flood*, VSE (*Valve Source Engine*) *flood*, DNS (*Domain Name Service*) *flood*, GRE-ETH (GRE-*Ethernet*) *flood*. Dentre eles, destacam-se os cinco primeiros como os mais frequentes, correspondendo à mais de 77,83% dos ataques observados por ANTONAKAKIS *et al.* [1].

3.2.3 Experimentos com ataques de *botnets*

Para obter os dados de tráfego de ataques de negação de serviço MENDONÇA *et al.* [15] desenvolveu um módulo de *software* capaz de executar ataques das *botnets* Mirai e BASHLITE. Esse programa foi criado a partir do código-fonte dos *malwares*, os quais encontram-se disponíveis publicamente em https://github.com/ifding/iot-malware [40].

O módulo de *software* desenvolvido por MENDONÇA *et al.* [15] abrange apenas a fase de ataque das *botnets*, ou seja, toda a parte do código responsável por realizar

a busca por novas vítimas e a comunicação com o Servidor de Comando e Controle (C2) foi removida.

Para a realização daquele trabalho foram selecionados ataques do tipo volumétricos (UDP *flood* e UDP-PLAIN *flood*) e de exaustão de estado (TCP SYN *flood* e TCP ACK *flood*). Para as botnets Mirai e Bashlite foram considerados todos os vetores de ataque citados anteriormente com exceção do UDP-PLAIN flood que é exclusivo do Mirai, totalizando sete tipos de ataques. Esses vetores de ataques estão entre os mais recorrentes [1, 37].

Tabela 3.2: Dicionário de usuários e senhas do *malware* Mirai.

Usuário	**Senha**	**Usuário**	**Senha**	**Usuário**	**Senha**
root	xc3511	admin	1111	root	7ujMko0vizxv
root	vizxv	root	666666	root	7ujMko0admin
root	admin	root	password	root	system
admin	admin	root	1234	root	ikwb
root	888888	root	klv123	root	dreambox
root	xmhdipc	Administrator	admin	root	user
root	default	service	service	root	realtek
root	juantech	supervisor	supervisor	root	00000000
root	123456	guest	guest	admin	1111111
root	54321	guest	12345	admin	1234
support	support	admin1	password	admin	12345
root	(vazia)	administrator	1234	admin	54321
admin	password	666666	666666	admin	123456
root	root	888888	888888	admin	7ujMko0admin
root	12345	ubnt	ubnt	admin	1234
user	user	root	klv1234	admin	pass
admin	(vazia)	root	Zte521	admin	meinsm
root	pass	root	hi3518	tech	tech
admin	admin1234	root	jvbzd	mother	f***er
root	1111	root	anko		
admin	smcadmin	root	zlxx.		

O UDP *flood* é um ataque que cria e envia uma grande quantidade de pacotes UDP para portas aleatórias no host de destino (alvo). Os pacotes são preenchidos com dados randômicos de tamanho pré-definido pelo atacante. Por sua vez, o UDP-PLAIN *flood* diferencia-se do anterior pois realiza um mapeamento endereço IP/porta de origem para um descritor de *socket* (*bind*), antes de efetuar a conexão com a máquina de destino. Dessa forma, o hacker pode especificar qual porta será utilizada durante o ataque [41].

Ataques do tipo TCP SYN *flood* e TCP ACK *flood* exploram o mecanismo de estabelecimento de comunicação confiável *three-way handshake* do TCP para inundar o servidor alvo com pacotes TCP SYN ou ACK. No primeiro caso apenas a flag SYN

é ativada, enquanto no segundo tipo de ataque apenas a flag ACK é configurada [41, 42]. Ao receber uma grande quantidade de solicitações, o servidor sobrecarregado torna-se incapaz de responder novas requisições de usuários legítimos.

A Figura 3.3 ilustra a configuração dos experimentos realizados no laboratório para gerar o tráfego de ataques DoS. Como fonte IoT do ataque (*bot*) foi utilizado um *Raspberry Pi* conectado pela rede *Wi-Fi* a um roteador sem fio e a vítima do ataque DoS simulada através de um endereço IP inválido (254.0.0.1). Essa configuração foi utilizada pois não é esperado pelo atacante nenhuma resposta da vítima, portanto o endereço IP alvo é irrelevante para o tráfego gerado pelo ataque.

O *Raspberry Pi* possui o módulo de software desenvolvido por MENDONÇA *et al.* [15] instalado e é capaz de realizar todos os vetores de ataques citados anteriormente. Por sua vez, o roteador sem fio roda o *firmware OpenWrt* [34] e possui a incumbência de coletar o número de bytes e pacotes que passam na interface WLAN a cada segundo.

Figura 3.3: Ambiente de simulação para geração de ataques DoS.

A Tabela 3.3 mostra os parâmetros utilizados durante os experimentos. Para cada vetor de ataque foram realizadas 3 simulações, as quais possuíam os mesmos parâmetros, com exceção do limite da taxa do *Wi-Fi* que foi variado entre 24 Mbps, 54 Mbps e a taxa máxima disponível. Cada simulação foi repetida 300 vezes, assim, para cada vetor de ataque considerado foi construído um dataset que possui amostras de 900 ataques (300 para cada taxa limite do *Wi-Fi*).

Tabela 3.3: Parâmetros dos experimentos para geração de ataques DoS.

Número de rodadas	300
Distribuição dos ataques	$\mathcal{N}(\mu = 120s, \sigma = 10s)$
***Payload* dos ataques do tipo UDP**	1400 bytes
***Payload* dos ataques do tipo TCP**	0
Limite da taxa do Wi-Fi (Mbps)	24 (simulação 1) 54 (simulação 2) máxima disponível (simulação 3)

A variação da taxa do *Wi-Fi* foi realizada para simular dispositivos IoT com diferentes capacidades de transmissão de dados e planos de Internet que podem variar entre os clientes de um ISP.

A parametrização dos ataques foi definida a partir do conhecimento que a maioria dos ataques DoS são de curta duração [43]. Os ataques seguem uma distribuição Gaussiana ($\mathcal{N}$) com média 120 segundos e desvio padrão de 10 segundos. Além disso, ataques do tipo TCP possuem *payload* vazio, enquanto os do tipo UDP possuem 1400 bytes nesse campo.

A Figura 3.4(a) mostra o boxplot do tráfego de *upload* gerado por cada vetor de ataque em Megabits por segundo (Mbps), com o eixo y em escala logarítmica e a Figura 3.4(b) apresenta o mesmo tipo de gráfico em pacotes por segundo (pps), com o eixo y em escala linear. Observa-se que ataques do tipo UDP possuem a mediana superior a 20 Mbps enquanto ataques do tipo TCP geram taxas muito inferiores, com medianas entre 1 e 2 Mbps. Com relação a taxa de *upload* em pacotes por segundo, nota-se que ocorre o inverso, ataques do tipo UDP produzem taxas menores quando comparados aos do tipo TCP, aproximadamente 2000 e 4000 pps, respectivamente.

Esse resultado é esperado pois no protocolo UDP não existe a necessidade de estabelecimento de comunicação entre a origem e o destino (não orientado a conexão). Dessa forma, o atacante é capaz de enviar uma grande quantidade de dados para o seu alvo em um único pacote (nos experimentos realizados foi definido *payload* de 1400 bytes). Por sua vez, o protocolo TCP exige o estabelecimento de comunicação em três vias (*three-way handshake*), assim, antes de enviar uma quantidade massiva de dados é necessário que o atacante envie pacotes de sincronização. Dessa forma, o *hacker* sobrecarrega o seu alvo com vários pacotes de sincronismo.

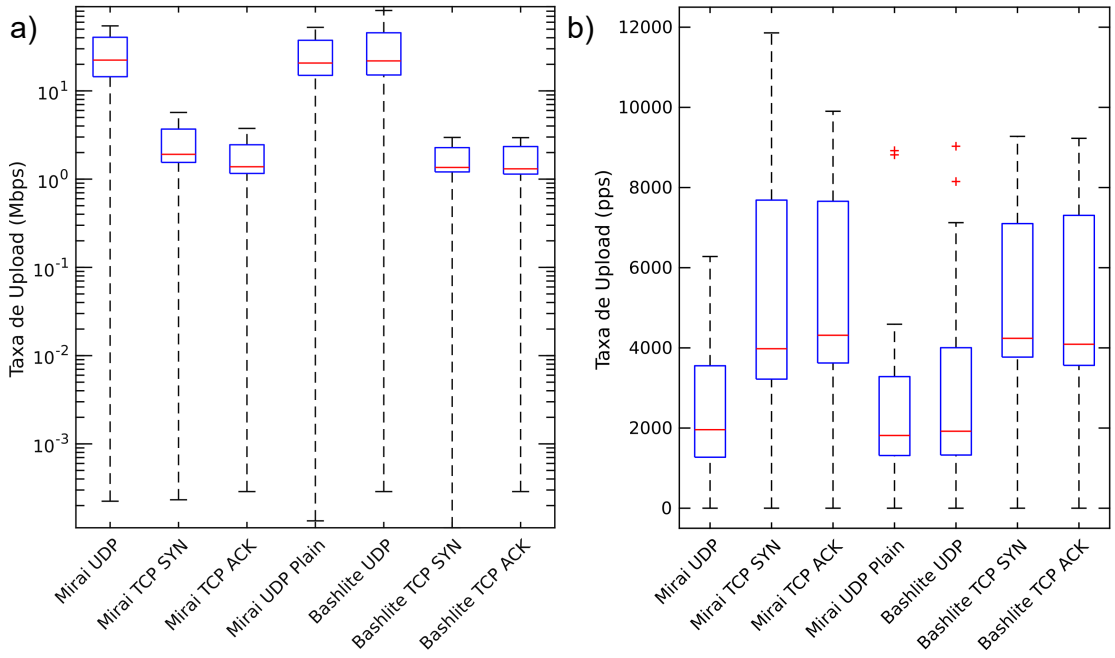

Figura 3.4: a) Boxplot do tráfego de *upload* gerado por cada vetor de ataque em Mbps com o eixo y em escala logarítmica. b) Boxplot do tráfego de *upload* gerado por cada tipo de ataque em pps.

Capítulo 4

Metodologia proposta

Uma das principais contribuições desse trabalho é propor um *framework* que seja simples e eficiente para detecção de ataques de negação de serviço em dispositivos IoT. A metodologia utilizada encontra-se resumida na Figura 4.1.

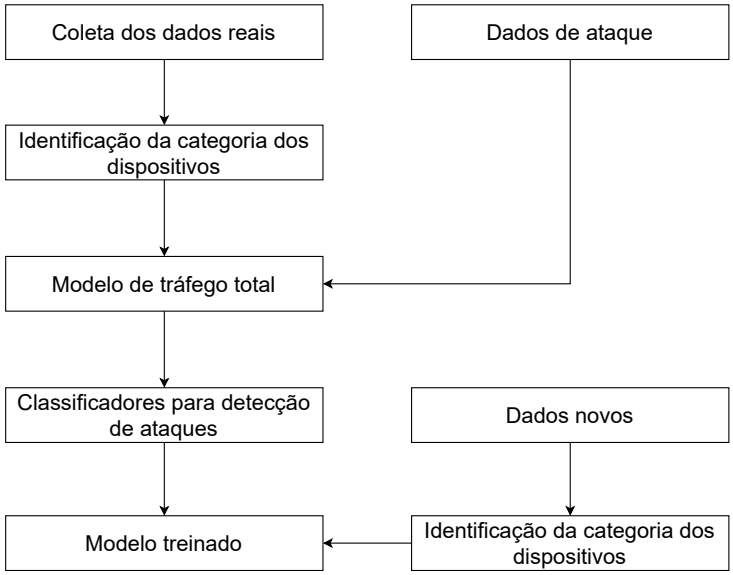

Figura 4.1: *Framework* proposto para detecção de ataques de negação de serviço.

A Seção 4.1 descreve os passos necessários para realizar a identificação dos tipos de equipamentos existentes nas residências a partir dos dados reais coletados (Seção 3.1). Em seguida, na Seção 4.2 é apresentado como é criado o *dataset* contendo tráfego de ataque a partir dos dados obtidos nas Seções 3.1 e 3.2. Por fim, na Seção 4.3 é descrito o procedimento utilizado para seleção de *features* e escolha do melhor modelo para detecção de ataques de negação de serviço para cada categoria de dispositivo.

O *framework* proposto obtém um modelo específico para cada categoria de dispositivo. Assim sendo, antes de realizar a classificação de novos dados é necessário

identificar o tipo de dispositivo para que seja usado o modelo de detecção de ataques correto.

4.1 Identificação da categoria dos dispositivos

No período de 1 de fevereiro a 27 de junho de 2020 foram coletados o contador de bytes e pacotes de 20.036 dispositivos conectados na WLAN de 806 clientes de um provedor de acesso à Internet. Para possibilitar o reconhecimento dos equipamentos responsáveis por gerar cada trace de tráfego também foram obtidas as informações dos *hostnames*, endereços MAC e os nomes dos aparelhos cadastrados pelos usuários do provedor de serviço de Internet no aplicativo disponibilizado pela empresa (*hostname* APP).

O método utilizado para identificação dos dispositivos geradores de cada dado de tráfego está ilustrado na Figura 4.2. Inicialmente foram identificados os endereços MAC únicos existentes nos dados. Em seguida foi aplicado o filtro de OUI (*Organizationally Unique Identifier*), disponível publicamente em http://standards-oui.ieee.org/oui/oui.txt [44]. Esse mecanismo consiste na busca por padrões nos três primeiros octetos do endereço MAC de cada equipamento.

O OUI é um identificador único de 24 bits registrado no IEEE (*Institute of Electrical and Electronics Engineers*), através do qual é possível descobrir o fabricante (*vendor*) de uma placa de rede *Wi-Fi*.

Posteriormente, para cada endereço MAC individual o procedimento abaixo foi repetido.

A categorização de um equipamento foi realizada em até três etapas. Na primeira delas foi efetuada a extração do campo *hostname* do dispositivo analisado, seguido da aplicação do filtro de *hostname*, o qual consiste na busca por palavras-chave nesse campo e atribuição da categoria do equipamento quando o padrão for encontrado.

A Tabela 4.1 apresenta as palavras-chave e as categorias que foram identificadas após análise dos dados disponíveis. Os dispositivos Wi-Fi foram classificados entre os seguintes grupos: *, câmera, *Ipod*, dispositivo *Android*, *chromecast*, *video game*, *Google Home*, dispositivo de rede, não identificado, maquininha de cartão de crédito, computador, impressora, *raspberry*, *smart TV*, *smartphone*, *smartwatch*, dispositivo de som, *tablet* e TV *box*.

A classificação "dispositivo *Android*" engloba qualquer equipamento que rode o Sistema Operacional *Android* (p. ex. *smartphone*, *tablet*, *smartwatch* e etc). Já as categorias "não identificado" e " * " representam dispositivos que não puderam ser classificados, porém por motivos diferentes. No primeiro caso devido a ausência de algum filtro, enquanto no segundo é impossível categorizá-lo com as informações disponíveis.

Se a classificação do equipamento foi diferente de “não identificado”, “ * ” ou “dispositivo *Android*”, a categorização foi finalizada. Em caso contrário, o dispositivo foi submetido a segunda etapa da identificação, a qual consiste na aplicação do filtro de *hostname* APP. Esse filtro é o mesmo utilizado na primeira etapa, porém é aplicado no campo *hostname* APP dos equipamentos.

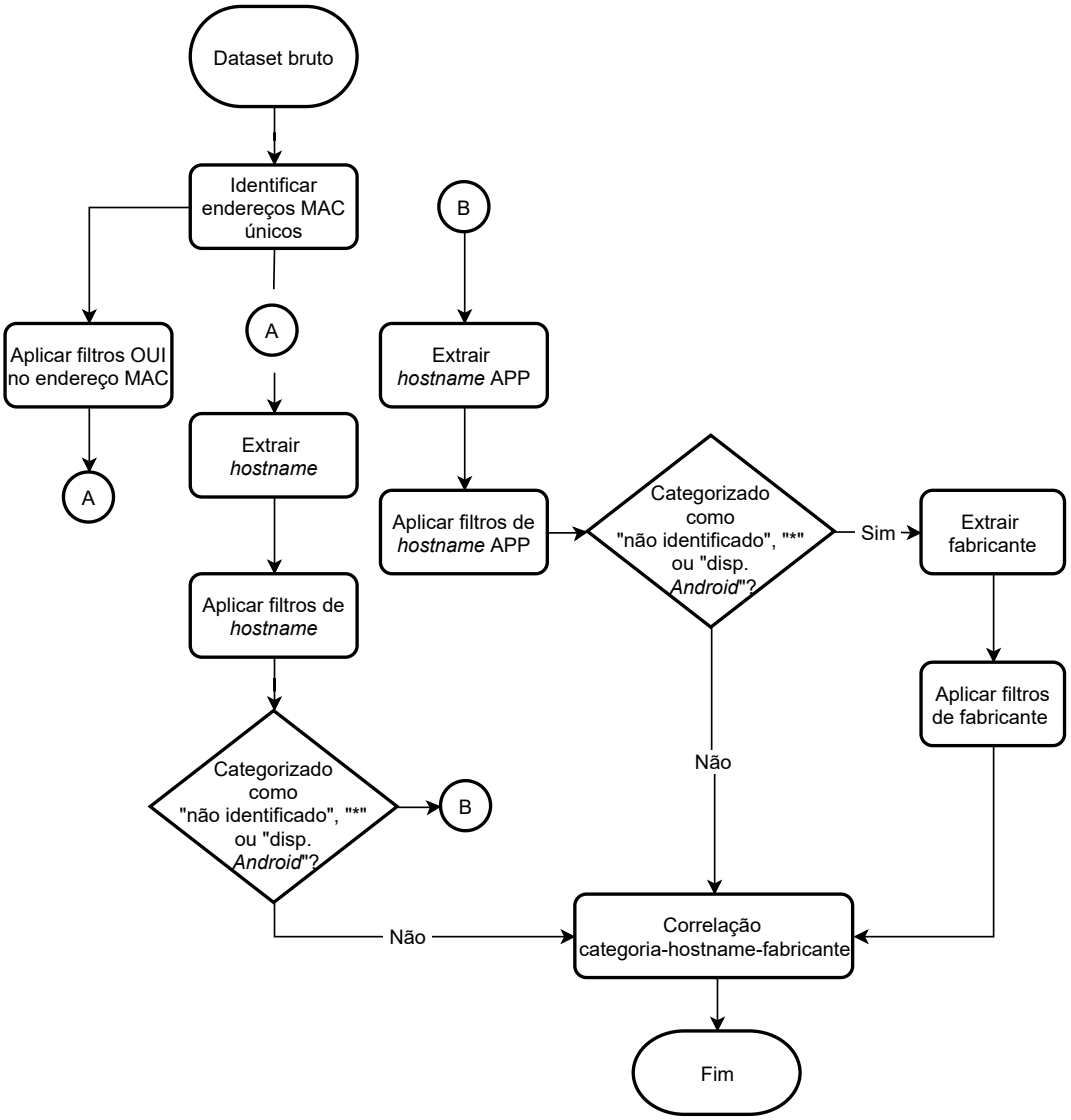

Figura 4.2: Metodologia utilizada para categorização dos dispositivos.

Ao final desse passo foi realizada uma nova verificação da categorização resultante. Caso o resultado obtido seja “não identificado”, “ * ” ou “dispositivo *Android*”, o equipamento foi submetido a terceira etapa de classificação. Por outro lado, a categorização foi encerrada se uma das outras 16 categorias foi atribuída.

Por fim, o terceiro passo consiste na aplicação do filtro de fabricante no campo fabricante dos dispositivos restantes. As palavras-chave utilizadas nesse filtro englobam alguns fabricantes específicos de determinadas categorias de equipamentos, conforme mostrado na Tabela 4.2. A Nintendo Co., Ltd, por exemplo, até onde se

tem conhecimento, fabrica apenas *video games*. Por sua vez, a Sony divide seus produtos em dois ramos, a Sony Interactive Entertainment (exclusiva de *video games*) e a Sony Mobile Communications Inc para os demais produtos da marca.

Tabela 4.1: Palavras-chave utilizadas para categorizar os dispositivos através do *hostname* (filtro de *hostname* e *hostname* APP).

Categoria	**Palavras-chave**
smartphone	phone; galaxy-A3; galaxy-A5; galaxy-A6; galaxy-A7; galaxy-A8; galaxy-A9; galaxy-A10; galaxy-A20; galaxy-A30; galaxy-A50; galaxy-M10; galaxy-M20; galaxy-M30; galaxy-S6; galaxy-S7; galaxy-S8; galaxy-S9; galaxy-S10; galaxy-J2; galaxy-J4; galaxy-J5; galaxy-J6; J7-Prime; J7-Neo; galaxy-J7; galaxy-J8; Redmi7; RedmiNote4; RedmiNote5; RedmiNote6; RedmiNote7; RedmiNote8; Redmi5; redmi8; K40S; Mi8Lite; sm-J410G; SM-J260M; K-12; K12; K50S; K-50S; redmis2; redmi4x; galaxy-S20; celular; OnePlus5T; HUAWEI_P8; HUAWEI_P20; HUAWEI_P30; HUAWEI_Y6;HUAWEI_Y7; Mi9-; Mi8-; Mi6-; Mi9SE; Mi9T; moto g3; moto g4; moto g5; moto g6; motog4; motog5; motog6; motorola one; moto one; motog2; moto g2; motog3; zenfone; lg k10; blackberry; galaxy-note
câmera	ipcam; ipc
computador	laptop; pc; note; desktop; aspire; vostro; inspiron; comp; iMac; macbook; netbook; Air-; -Air; mbp; vaio
TV box	tvbox; tv box; apple-tv; appletv; directv
smart TV	tv; TIZEN; BRAVIA; televisão; televisao
chromecast	chromecast
impressora	EPSON; impressora; printer
tablet	iPad; galaxy-tab; tablet
smartwatch	AppleWa; galaxywatch; gearfit; gearS3; watch; Motoactv; amazfit
dispositivo de rede	repeater; wn3000rpv3; re450; re305; re200; TL-WR; Wireless-N; technicolor; roteador; repetidor; extender; TL-WA; wrn240; mw300re; iwe3001; iwe3000N
video game	Wii; XBOX; PS3; PS4; playstation; play station; vídeo game; vídeogame; video game, videogame
maquininha de cartão de crédito	S920-
dispositivo de som	sound; home theater
Ipod	ipod
Raspberry	raspberry
Google Home	google mini; google home; google-home
dispositivo android	android
*	*
não identificado	Todos os dispositivos não identificados com as palavras-chave anteriores

Independente do resultado obtido ao final da terceira etapa a categorização dos dispositivos está concluída. Adicionalmente foi realizada a correlação categoria-*hostname*-fabricante com o intuito de padronizar e identificar fabricantes não descobertos inicialmente.

A Figura 4.3 ilustra a metodologia utilizada para realizar o procedimento de correlação. Nesse método primeiramente foi verificado a categoria na qual o dispositivo pertence. Em seguida foi analisado se o campo *hostname* ou o atributo fabricante contém alguma das palavras-chave. Quando um padrão foi encontrado, o campo fabricante foi atualizado para o valor correspondente. A Tabela 4.3 mostra todas as palavras-chave usadas para realizar o método da correlação.

Tabela 4.2: Palavras-chave utilizadas para categorizar os dispositivos através do fabricante (filtro de fabricante).

Categoria	**Palavras-chave**
smartphone	T&A Mobile Phones
computador	Intel Corporate
impressora	Seiko Epson Corporation
smartwatch	Garmin International; Fitbit, Inc.
dispositivo de rede	Tenda Technology Co., Ltd; Technicolor CH USA Inc
video game	Sony Interactive Entertainment Inc;Nintendo Co.,Ltd; Nintendo Co., Ltd
maquininha de cartão de crédito	PAX Computer Technology(Shenzhen) Ltd

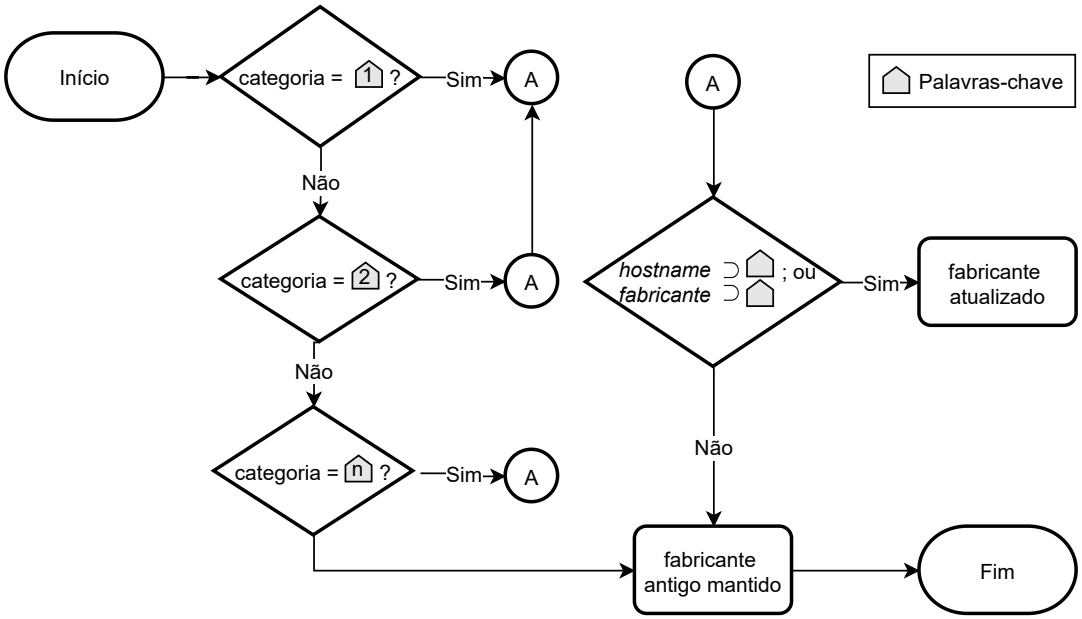

Figura 4.3: Metodologia utilizada para a correlação categoria-*hostname*-fabricante. As palavras-chave empregadas estão na Tabela 4.3.

Tabela 4.3: Palavras-chave utilizadas na correlação categoria-*hostname*-fabricante.

Categoria igual à	**Hostname contém (⊃)**	**Fabricante antigo contém (⊃)**	**Fabricante novo igual à**
smartwatch	applewa	apple	Apple
	gears3; galaxywatch	samsung	Samsung
	amazfit	huami	Huami
	fitbit	fitbit	Fitbit
	garmin	garmin	Garmin
impressora	hp	hewlett packard	HP
	epson	epson	Epson
chromecast	chromecast	google	Google
smart TV	lg	lg	LG
	sony	sony	Sony
	samsung; tizen	samsung	Samsung
video game	wii	nintendo	Nintendo
	xbox	microsoft	Microsoft
	ps3; ps4; playstation; play station	sony	Sony
Google Home	google	google	Google
TV *box*	apple	apple	Apple
	directv	directv	Directv
dispositivo de rede	tl-wa; tl-wr	tp-link	TP-LINK
	technicolor	technicolor	Technicolor
	tenda	tenda	Tenda
	mw300re	mercusys	Mercusys
	intelbras	intelbras	Intelbras
computador	dell; inspiron; latitude; vostro	dell	Dell
	sony, vaio	sony	Sony
	samsung	samsung	Samsung
	lg	lg	LG
	imac; macbook; air	apple	Apple
	aspire	acer	Acer
	asus	asus	Asus
tablet	ipad; apple	apple	Apple
	galaxy	samsung	Samsung
smartphone	redmi; xiaomi; pocophone; miphone	xiaomi	Xiaomi
	iphone; apple	apple	Apple
	sm-j410g; sm-j260m; galaxy; J7-Prime; J7-Neo	samsung	Samsung
	asus; zenfone	asus	Asus
	lg; k40s; k-12; k12; k50s; k-50s	lg	LG
	huawei	huawei	Huawei
	moto	motorola	Motorola
	blackberry	blackberry	Blackberry
	oneplus	oneplus	Oneplus

4.2 Modelo de tráfego total

Uma vez finalizado o procedimento de identificação dos tipos de equipamentos existentes no dados coletados, foram selecionadas cinco categorias de dispositivos IoT para o desenvolvimento deste trabalho, são eles: câmera, impressora, *video game*, *chromecast* e *smart* TV.

Para realizar a escolha das categorias foram excluídas aquelas que possuem equipamentos multipropósito (p. ex. *smartphone*, computador e *tablet*). Em seguida, dentre as categorias restantes foram selecionadas aquelas com maior quantidade de dispositivos e as mais vulneráveis a infecção por *botnets* de IoT.

O modelo de tráfego total foi o nome atribuído ao método responsável por criar o *dataset* que foi usado para treinamento e teste dos classificadores de detecção de ataques. A técnica adotada baseou-se em MENDONÇA *et al.* [15], a qual consiste resumidamente em agregar o tráfego de *upload* real dos equipamentos e o de ataque (descrito na seção 3.2).

Devido algumas particularidades do *dataset* utilizado nesta dissertação, este método diferencia-se do empregado naquele trabalho por considerar um percentual mínimo de amostras para um dispositivo ser selecionável como um bot, realizar controle de ausência de dados no *dataset*, inserção dos dados de ataques nos dispositivos infectados de maneira independente (não sincronizado) e controle de *loops* infinitos.

A seguir será detalhado o procedimento que foi repetido para cada categoria de dispositivo estudada. A fim de facilitar o entendimento, a Tabela 4.4 contém a definição dos termos e variáveis que serão utilizados durante a explicação.

Em primeiro lugar os dados de ataque foram re-amostrados pela média amostral da escala de segundos para minutos, ou seja, foram somadas todas as amostras durante 60 segundos e esse total dividido pelo total de amostras. Esse procedimento foi realizado para deixar os dados de tráfego real e de ataque com a mesma taxa de amostragem.

Com o intuito de aumentar a chance de deteccção de ataques de curta duração foi utilizado o conceito de janela deslizante. Dessa forma, o dado obtido de cada equipamento individualmente foi dividido em janelas (*slots*) deslizantes de tempo de cinco minutos, conforme ilustrado na Figura 4.4. Dessa forma, a janela 1 possui amostras do minuto 1 ao 5, o segundo *slot* do minuto 2 ao 6 e assim sucessivamente. Esse processo foi repetido tanto para o tráfego de *upload* (bytes e pacotes) quanto *download*.

Caso existam amostras faltantes como mostrado na Figura 4.4, onde ocorre ausência de dados nos minutos 6, 20 e 21, são criados *slots* inválidos sempre que a janela englobar o minuto perdido. No exemplo em questão, a inexistência da amostra no tempo 6 originou os *slots* inválidos 2 ao 6. Essa situação influencia nos próximos

passos do modelo e será explicada posteriormente.

Tabela 4.4: Definição das variáveis e termos utilizados.

Termo/Variável	**Descrição**
$\mathcal{A}$	Conjunto composto por todos os dispositivos pertencentes a categoria estudada.
$\mathcal{B}$	Conjunto composto pelos dispositivos que foram infectados pelo malware.
$\mathcal{C}$	Conjunto formado pelos dispositivos que satisfazem a condição $\rho_k > \tau$.
τ	Percentual mínimo de amostras necessárias para um dispositivo ser selecionável como um *bot*.
ρ_k	Percentual de amostras existente no dispositivo k.
i	Dispositivo que foi infectado (*bot*), onde i = 1, 2, ..., $\|\mathcal{B}\|$.
j	Identificador do ataque que foi adicionado ao tráfego normal do *bot* i, onde j = 1, 2, ..., 4704.
Vetor de ataque	Tipo de ataque (Mirai UDP, Mirai UDP-PLAIN, Mirai TCP SYN, Mirai TCP ACK, BASHLITE UDP, BASHLITE TCP SYN ou BASHLITE TCP ACK).
Trace de ataque	Tráfego gerado pelo ataque durante a sua execução.
v_j	Vetor de ataque selecionado para o ataque j
h_j	Trace selecionado para o ataque j
d_j	Duração do ataque j em minutos. Essa informação é extraída a partir do trace de ataque selecionado.
t_j	Instante de tempo onde o ataque j foi iniciado.
Perfil de ataque (a_{ij})	Característica do ataque que será inserido no *bot* i. O perfil de ataque é composto por v_j, h_j e d_j.
ω	Tamanho da janela de tempo deslizante
ω_{ini}	A primeira janela que contém o instante de tempo inicial do ataque.
ω_{fin}	A última janela que contém o instante de tempo final do ataque.

No segundo passo do modelo de tráfego total foi realizada a inserção de dados de ataque no tráfego real coletado do ISP. Nessa etapa foram selecionados aleatoriamente no máximo 5% dos equipamentos pertencentes a categoria analisada para criar um conjunto $\mathcal{B}$ composto pelos dispositivos infectados pelo *malware*, onde $\mathcal{B} \subset \mathcal{A}$ e $\mathcal{A}$ representa todos os equipamentos pertencentes a classe estudada. O valor de 5% foi baseado em AUCHARD [45]. Naquele trabalho, foi relatado que em 2016, por volta de 5% dos roteadores dos clientes de uma empresa de telecomunicações na Alemanha sofreram tentativas de infecção pelo *malware* Mirai.

Os dados obtidos do tráfego do *Wi-Fi* apresentam a particularidade de possuir muitas amostras faltantes, provavelmente devido o equipamento ser desligado ou removido da rede. O percentual de ausência de dados varia tanto entre categorias, quanto entre elementos de uma mesma classe. Para evitar a tentativa de adição de ataques em um dispositivo com muitas lacunas em seu tráfego, foi adotado um *threshold* (τ) do percentual mínimo de amostras de 40% para tornar um equipamento

selecionável como um *bot*. Assim, seja ρ_k o percentual de amostras existentes no dispositivo k e $\mathcal{C}$ o conjunto formado pelos dispositivos que satisfazem a condição $\rho_k > \tau$, onde $\mathcal{C} \subset \mathcal{A}$, a cardinalidade de $\mathcal{B}$ ($|\mathcal{B}|$) foi definida da seguinte forma:

$$|\mathcal{B}| = \begin{cases} 0,05 \cdot |\mathcal{A}| & \text{, se } |\mathcal{C}| \geq 0,05 \cdot |\mathcal{A}| \quad (4.1) \\ |\mathcal{C}| & \text{, se } 2 \leq |\mathcal{C}| < 0,05 \cdot |\mathcal{A}| \quad (4.2) \\ 1 & \text{, se } |\mathcal{C}| \leq 1 \quad (4.3) \end{cases}$$

A partir da cardinalidade de $\mathcal{B}$, seus elementos foram obtidos da seguinte maneira: a) escolhidos aleatoriamente dentre os componentes do conjunto $\mathcal{C}$ (caso 4.1); b) composto por todos os elementos do conjunto $\mathcal{C}$ (caso 4.2); c) selecionado o dispositivo com maior ρ_k, se $|\mathcal{B}| = 1$ (caso 4.3).

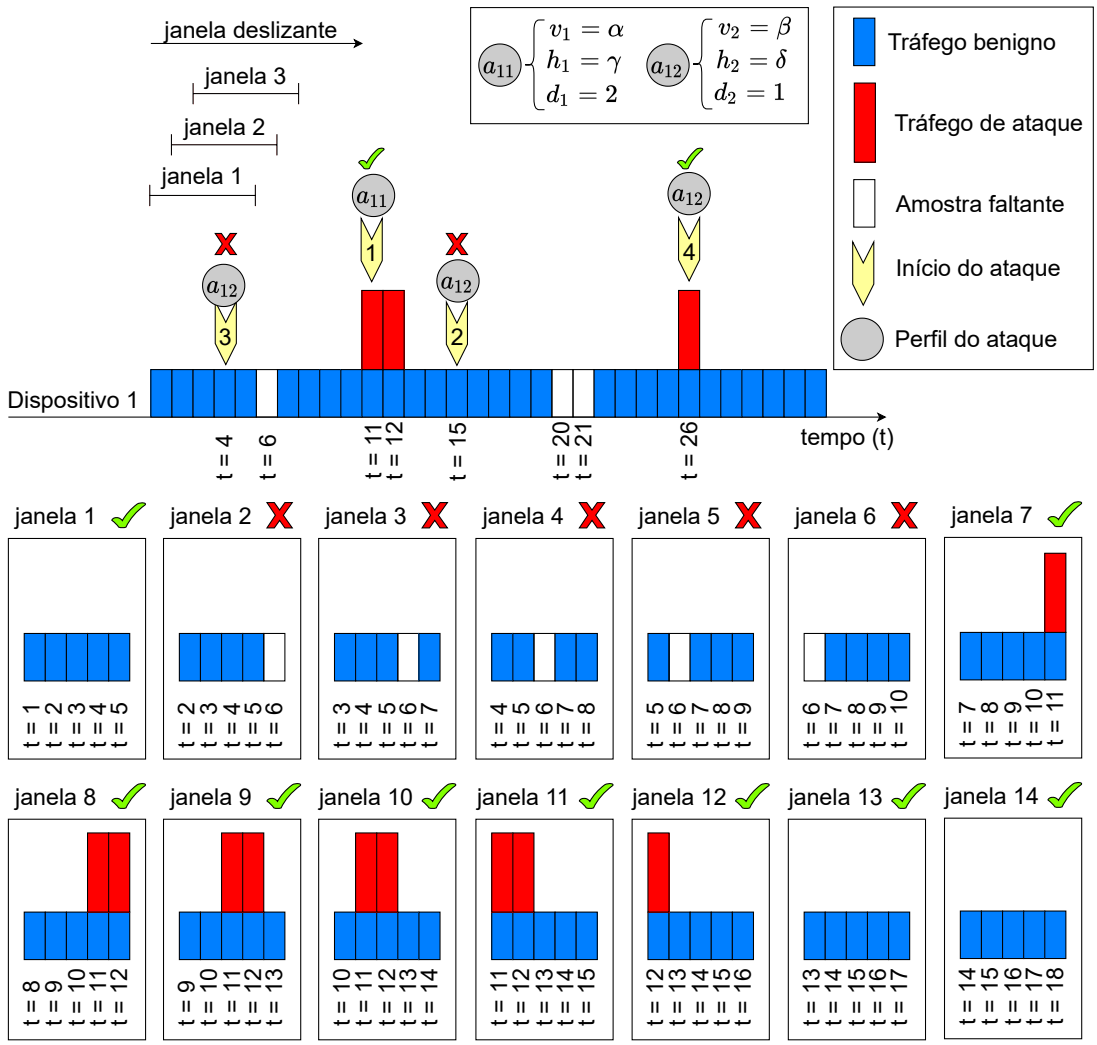

Figura 4.4: Modelo de tráfego total. Essa Figura apresenta como foi realizada a combinação do tráfego de ataque com o tráfego benigno dos dispositivos utilizando o conceito de janela deslizante.

Os *bots* agem seguindo ordens do Servidor de Comando e Controle podendo

efetuar ataques em qualquer horário do dia e com qualquer duração. Além disso, não é obrigatório que todos os equipamentos infectados realizem ataques no mesmo instante de tempo. Segundo MARZANO *et al.* [2], "os operadores de *botnets*, em geral, trabalham em rajadas, emitindo vários comandos em uma curta sessão seguida por longos períodos de inatividade". Ainda naquele trabalho foi observado que o intervalo entre sessões do Servidor de Comando e Controle possui mediana aproximada de 2700 segundos. Assim, para cada *bot* utilizado neste trabalho foi considerado uma média de 32 ataques por dia, totalizando aproximadamente 4704 traces de ataques durante o período analisado.

Para cada ataque j que foi adicionado ao tráfego normal do dispositivo infectado i, foi selecionado aleatoriamente e de maneira uniforme, com reposição, um dos sete vetores de ataques estudados e um dos novecentos traces dos *malwares* disponíveis. Além disso, também foi sorteado o instante de tempo t_j onde o ataque j foi iniciado.

Antes de realizar a inserção dos dados de ataque é essencial identificar todas as janelas que vão receber informações de ataques e verificar se todas elas são válidas. Outrossim, também deve-se observar se alguma janela já possui dados de ataque adicionados. A Figura 4.4 exemplifica a metodologia completa.

Seja ω o tamanho da janela de tempo deslizante ($\omega = 5$ minutos), a_{ij} o perfil do ataque j no dispositivo i, v_j, h_j e d_j o vetor, o trace e a duração do ataque em minutos, selecionados para o ataque j, respectivamente. Os retângulos vermelhos representam o tráfego de ataque, enquanto os azuis ilustram o tráfego natural gerado pelo equipamento.

Observa-se na Figura 4.4 que o primeiro ataque que foi adicionado aos dados do dispositivo 1 possuía as seguintes configurações: $v_1 = \alpha$, $h_1 = \gamma$ e $d_1 = 2$ e $t_1 = 11$. Uma vez conhecido o perfil do ataque e o instante de tempo no qual o mesmo se inicia, foram identificadas a primeira e última janela que deveriam conter informações do ataque. Seja ω_{ini} a primeira janela que contém o instante de tempo inicial do ataque (t_1), $\omega_{ini} = t_1 - \omega + 1$ e seja ω_{fin} a última janela que contém o instante de tempo final do ataque, $\omega_{fin} = t_1 + d_1 - 1$. Todas as janelas no intervalo $[\omega_{ini}, \omega_{fin}]$ conterão amostras de ataque e, portanto, devem ser válidas, ou seja, possuir dados do tráfego benigno do dispositivo em todos os tempos que fazem parte da janela analisada.

Em virtude de todos os requisitos terem sido satisfeitos (todos as janelas válidas e inexistência de sobreposição de ataques), o tráfego do ataque 1 foi somado ao tráfego do dispositivo 1 no instante de tempo e janelas correspondentes. Esse procedimento foi aplicado apenas ao tráfego de *upload*, pois independente da resposta enviada pela vítima, o *bot* continuará atacando o seu alvo. Dessa forma, o tráfego de *download* recebido pelo *bot* pode ser considerado desprezível.

Para o segundo ataque do *bot* 1 foi selecionado aleatoriamente um novo perfil do ataque ($v_2 = \beta$, $h_2 = \delta$ e $d_2 = 1$) e o instante inicial do ataque $t_2 = 15$. Nesse caso,

não foi possível adicionar o ataque no instante previsto ($t_2 = 15$), pois as janelas 11 e 12 já possuem tráfego de ataque. Quando essa situação ocorre, o perfil do ataque é mantido e uma nova posição do início do ataque é sorteada ($t_2 = 4$). Novamente não foi permitido inserir o ataque em $t_2 = 4$ devido a algumas janelas no intervalo $[\omega_{ini}, \omega_{fin}]$ não serem válidas e o valor inicial para a janela ($\omega_{ini} = 0$) não ser um valor válido, pois o menor valor de ω_{ini} é 1. Mais uma vez um novo instante inicial para o ataque foi escolhido ($t_2 = 26$), e dessa vez sem óbices para combinação dos dados de ataque ao tráfego dos equipamentos. A fim de evitar *loops* infinitos são realizadas no máximo cem tentativas para adição dos dados de ataque ao tráfego normal do dispositivo.

Esse procedimento foi repetido para todo o conjunto $\mathcal{B}$. Após sua conclusão, o *dataset* agregado está pronto para ser utilizado como entrada dos classificadores para detecção de ataques que serão descritos na próxima seção.

4.3 Classificadores para detecção de ataques

Nesta seção será descrito como foram desenvolvidos os modelos capazes de detectar a presença de ataques de negação de serviço em cada janela de tempo, utilizando apenas informações do contador de bytes e pacotes dos dispositivos. Para cada tipo de equipamento estudado foi repetido o método ilustrado na Figura 4.5. Dessa forma, foi obtido um classificador específico para cada categoria de dispositivo.

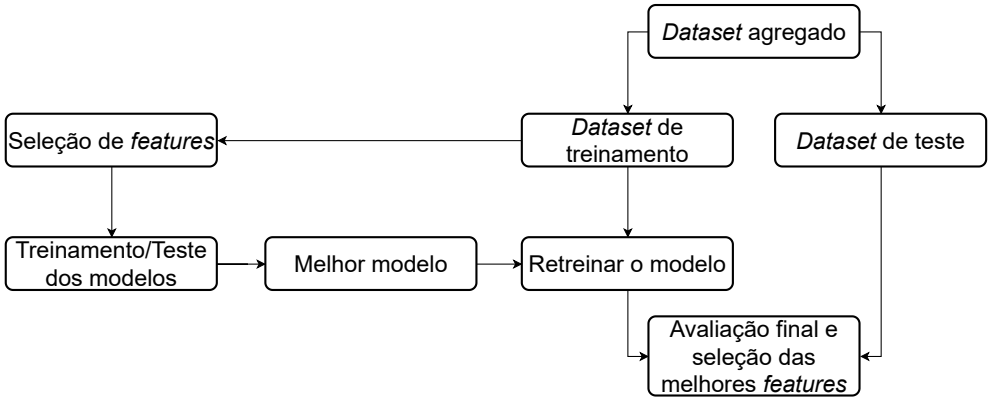

Figura 4.5: Metodologia utilizada para os classificadores de ataques de negação de serviço.

Em primeiro lugar, a partir do contador de bytes e pacotes existente no *dataset* agregado (Seção 4.2) foram derivadas para cada minuto dentro de cada *slot* de cinco minutos, quatro métricas de rede: up rate bps (taxa de *upload* em bits por segundo), up rate pps (taxa de *upload* em pacotes por segundo), up/down ratio bps (razão entre o tráfego de *upload* e *download* em bps) e up/down ratio pps (razão entre o tráfego de *upload* e *download* em pps).

Em seguida, para cada janela de tempo e para cada uma das métricas de rede foram calculadas as seguintes estatísticas: valor máximo, diferença entre o máximo e o mínimo, média, mediana e desvio padrão, totalizando vinte *features* por janela de tempo, as quais foram usadas como entrada dos classificadores (Tabela 4.5). Adicionalmente, cada *feature* foi normalizada utilizando o *z-score* que é dado pela equação:

$$z_i = \frac{x_i - \mu}{\sigma} \tag{4.4}$$

Onde x_i corresponde ao valor atual da *feature* em uma janela de tempo i e μ e σ representam respectivamente, a média e o desvio padrão da *feature* considerando todas as janelas de tempo. Dessa forma, cada *feature* terá média 0 e um desvio padrão igual a 1 [46].

Outras métricas (taxa de *download* em bps ou pps) e estatística (valor mínimo) também foram testadas, porém não houve ganho de performance nos modelos de detecção de ataques.

Tabela 4.5: *Features* extraídas do *dataset*.

Métrica de rede	**Estatísticas**
up rate bps	max, max-min, média, mediana, desvio padrão
up rate pps	max, max-min, média, mediana, desvio padrão
up/down ratio bps	max, max-min, média, mediana, desvio padrão
up/down ratio pps	max, max-min, média, mediana, desvio padrão

O próximo passo realizado foi dividir o *dataset* agregado em dois conjuntos. O primeiro deles chamado *dataset* de treinamento contendo aproximadamente 75% dos dias analisados, enquanto o segundo denominado *dataset* de teste incluindo os 25% restantes. Assim, os dados de treinamento abrangem o período de 01 de fevereiro a 21 de maio de 2020 (111 dias) e os dados de teste de 22 de maio a 27 de junho (36 dias).

O *dataset* de treinamento foi empregado para realizar a seleção de *features* (*feature selection*), a seleção do modelo (*model selection*) e o retreinamento do melhor modelo. Por sua vez, o *dataset* de teste foi usado para avaliar o desempenho do melhor classificador.

Com o intuito de eliminar *features* redundantes, simplificar o modelo e reduzir o custo computacional foi realizada a seleção de *features*. Para realizar essa tarefa foi empregado o algoritmo *Sequential Forward Selection* (SFS) [47], o qual permitiu identificar as variáveis mais importantes para cada categoria de dispositivo.

Seja $X = \{x_1, x_2, ..., x_p\}$ o espaço de *features* existente de dimensão p e deseja-se obter um conjunto $Y = \{y_i \mid i = 1, 2, ..., k; y_i \in X\}$ de dimensão k, onde $k < p$. O método SFS é inicializado com $Y = \emptyset$. Em seguida, a cada passo o algoritmo

testa todas as variáveis do conjunto X (dentre aquelas ainda não adicionadas em Y) e verifica qual delas que maximiza a função de critério (*criterion function*) quando adicionada. Uma vez encontrada essa *feature*, ela é adicionada ao conjunto Y. Esse procedimento é repetido até $|Y| = k$. A função de critério é composta por um algoritmo de classificação e uma métrica de desempenho (p. ex. Árvore de decisão e acurácia) [47].

As métricas acurácia, *precision*, *recall* e *F1 score* são comumente usadas para avaliação de modelos e são calculadas através das equações abaixo:

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.5}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.6}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.7}$$

$$F1\ Score = \frac{2 \cdot precision \cdot recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2} \cdot (FP + FN)} \tag{4.8}$$

Onde TP, TN, FP e FN correspondem respectivamente, a quantidade de verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos.

A métrica acurácia indica o desempenho do modelo com relação aos verdadeiros positivos e verdadeiros negativos. No entanto, deve-se ressaltar que essa medida pode ser ilusória quando o *dataset* possuir um desbalanceamento entre as categorias (p. ex. número de janelas com tráfego benigno $\gg$ número de janelas com tráfego de ataques).

A métrica *precision* (valor preditivo positivo) mede a precisão do modelo em detectar um ataque. Esse tipo de medida é importante em cenários onde um número alto de falsos positivos não é desejável, ou seja, o modelo deve identificar um ataque apenas quando ele tiver realmente certeza, mesmo que para isso deixe de identificar alguns ataques. Por sua vez, o *recall* (sensibilidade) mede a probabilidade de um ataque ser detectado. Esse tipo de métrica é importante em cenários onde um número alto de falsos negativos não é desejável, ou seja, em circunstâncias onde é preferível identificar um ataque em uma janela onde só existe tráfego normal.

Por fim, o *F1 score* combina as métricas *precision* e *recall* para medir a performance do classificador. Esse tipo de medida facilita a interpretação pois apenas uma métrica é avaliada ao invés de duas. No entanto, diferentemente de uma média aritmética, no *F1 score* os valores menores possuem maior influência.

Para a seleção de *features* o algoritmo SFS foi configurado para encontrar as dez *features* mais relevantes utilizando validação cruzada com 5-*fold* (5-*fold cross-validation*), os classificadores Floresta Aleatória (RF – *Random Forest*), Árvore de Decisão (DT – *Decision Tree*) e K-vizinhos mais próximos (KNN – *K-Nearest Neigh-*

bor) e as métricas de desempenho acurácia e *F1-Score*. Desse modo, o método SFS foi executado para cada par classificador-métrica (seis vezes) para cada categoria de dispositivo. Destaca-se que ao final do procedimento é possível saber a performance do modelo com o acréscimo de cada *feature* individualmente.

A partir do resultado do SFS foram definidos quatro conjuntos de *features*. Primeiramente para cada uma das funções de critério empregadas foi identificada a quantidade de *features* que resultou no melhor desempenho. A Figura 4.6 mostra a pontuação (acurácia) obtida pelo modelo conforme as *features* foram sendo selecionadas pelo SFS, para os dispositivos categorizados como câmera. Observa-se que utilizando a RF, DT e KNN, o melhor desempenho foi obtido com 5, 5 e 9 *features*, respectivamente.

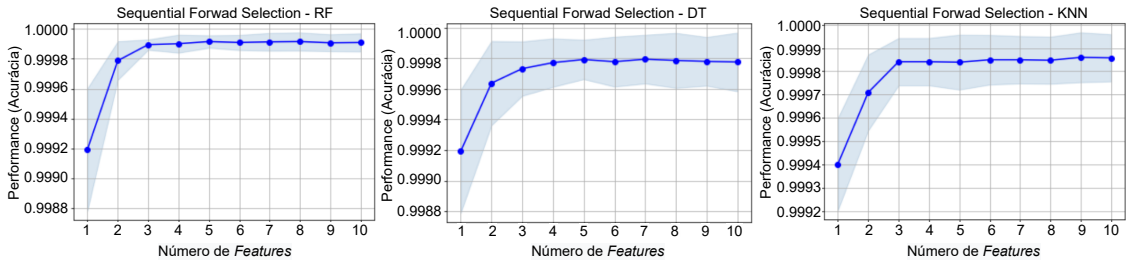

Figura 4.6: Pontuação das *features* selecionadas pelo algoritmo SFS para os dispositivos categorizados como câmera, considerando a métrica de avaliação acurácia e com os algoritmos de classificação (RF, DT e KNN). O eixo x indica a quantidade de *features* utilizadas e o eixo y a pontuação obtida pelo modelo.

Em seguida foi verificado qual dos três modelos resultou na melhor performance (RF, DT ou KNN). Nota-se que a *Random Forest* com cinco *features* atingiu uma acurácia superior aos outros dois algoritmos. Dessa forma, o primeiro conjunto de *features* foi composto pelas cinco primeiras *features* selecionadas pelo SFS com a RF. Analogamente, o segundo conjunto foi gerado substituindo a métrica acurácia pelo *F1 score*.

O terceiro e quarto conjuntos foram obtidos através de um ranqueamento das *features* comuns à pelo menos dois classificadores. As variáveis pertencentes a todos os modelos foram consideradas as mais relevantes, seguidas pelas comuns à dois deles. Como método de desempate foi avaliada a importância (posicionamento) das *features* em cada classificador. Posteriormente foram selecionadas as seis variáveis melhores ranqueadas com relação à acurácia para o conjunto número três e com relação ao *F1 score* para o quarto grupo.

A Tabela 4.6 mostra as 10 *features* selecionadas pelo SFS, utilizando a métrica acurácia e os algoritmos de classificação (RF, DT e KNN), para os dispositivos categorizados como câmera. Nesse caso, as seis *features* melhores ranqueadas foram: up rate pps (max), up rate bps (max - min), up rate pps (mean), up rate bps (std),

up rate bps (max) e up/down ratio pps (max).

Tabela 4.6: Resultado da seleção das 10 *features* mais relevantes para os dispositivos categorizados como câmera, utilizando o algoritmo SFS com a métrica de avaliação acurácia e com os algoritmos de classificação (RF, DT e KNN).

	Rank	***Features (RF)***	***Features (DT)***	***Features (KNN)***
Acurácia	1	up rate pps (max)	up rate pps (max)	up rate pps (max)
	2	up rate bps (max - min)	up/down ratio pps (max)	up rate bps (max)
	3	up rate pps (mean)	up rate pps (mean)	up rate pps (median)
	4	up rate bps (std)	up rate bps (max - min)	up/down ratio bps (median)
	5	up/down ratio pps (std)	up/down ratio bps (mean)	up rate bps (max-min)
	6	up rate bps (median)	up rate bps (std)	up rate pps (mean)
	7	up/down ratio bps (max)	up/down ratio bps (max)	up rate bps (mean)
	8	up rate pps (max - min)	up/down ratio bps (median)	up rate bps (std)
	9	up rate bps (max)	up/down ratio bps (std)	up/down ratio pps (median)
	10	up/down ratio pps (max)	up/down ratio bps (max - min)	up rate pps (std)

Em seguida os quatro conjuntos de *features* foram utilizados para realizar o *model selection*. Nessa etapa foram testados seis classificadores diferentes, configurados com os valores dos hiper-parâmetros *default* da biblioteca scikit-learn [48] do Python. Os modelos utilizados foram a *Random Forest*, Árvore de Decisão, Regressão Logística (LR – *Logistic Regression*), *Naive Bayes* Gaussiano (GNB – *Gaussian Naive Bayes*), *Perceptron* multi-camadas (MLP – *Multi-layer Perceptron*) e K-vizinhos mais próximos.

Cada um dos seis classificadores foram avaliados utilizando validação cruzada com 5-*fold* em quatro cenários distintos. Em cada um deles o modelo usou um conjunto de *features* diferentes e o classificador com melhor performance na métrica *F1 score* considerando todos os testes foi selecionado.

Por último, o modelo vencedor de cada categoria de dispositivo foi empregado para selecionar as melhores *features*. Para isso, o classificador foi retreinado usando todo o *dataset* de treinamento nos mesmos quatro cenários e avaliado no *dataset* de testes. Nessa etapa foi avaliado o desempenho do modelo e selecionado o conjunto de *features* no qual o classificador apresentou maior *F1 score*.

Capítulo 5

Resultados e Discussões

Neste capítulo primeiramente serão apresentadas algumas características dos dados utilizados para o desenvolvimento deste trabalho, tais como período da coleta, tipos de equipamentos existentes e seus respectivos consumos de tráfego (Seção 5.1). Posteriormente serão mostrados os resultados dos modelos desenvolvidos para identificar ataques de negação de serviço em algumas categorias de dispositivos IoT (Seção 5.2).

5.1 Conjunto de dados

O conjunto de dados usados neste estudo engloba o período de 1 de fevereiro a 27 de junho de 2020, totalizando 147 dias. Nesse intervalo foram coletadas amostras do tráfego da WLAN de vários clientes de um ISP. A cada minuto são contabilizados o total de bytes e pacotes recebidos (*download*) e enviados (*upload*) pela rede Wi-Fi por cada dispositivo individualmente. Ao todo foram obtidas informações geradas por 20.036 equipamentos pertencentes à 806 usuários. Esses dispositivos possuem funcionalidades e fabricantes diversos conforme mostrado nas Tabelas 5.1 e 5.2.

Observa-se na Tabela 5.1 que com a metodologia proposta baseada em filtros, foi possível categorizar 14254 dispositivos (71,28% do total analisado). Destaca-se os *smartphones* e dispositivos *Android* como os equipamentos mais recorrentes, com um total de 6507 e 5841, respectivamente. Se for levado em consideração que a maioria dos dispositivos que rodam o Sistema Operacional *Android* no Brasil são *smartphones*, o total real desta categoria é próximo à 10000, o que representaria aproximadamente metade dos equipamentos coletados. No entanto, 5755 dispositivos (28,72% do total) não foram identificados com o método empregado, pois em sua grande maioria possuem " * " ou nomes próprios que foram atribuídos por seus proprietários no campo *hostname*. Dessa forma, não fornecem nenhuma informação extra sobre qual categoria o equipamento pertence.

Além disso, também foi realizada a identificação dos fabricantes reais dos dispositivos. Esse procedimento foi executado em duas etapas, onde na primeira foi aplicado os filtros de OUI nos endereços MAC de cada dispositivo e na segunda etapa foi efetuada a correlação categoria-*hostname*-fabricante. A Tabela 5.2 ilustra um exemplo do resultado obtido com essa técnica.

Tabela 5.1: Resultado da categorização dos dispositivos do conjunto de dados.

Categoria	**Total**	**Categoria**	**Total**
smartphone	6507	maquininha de cartão de crédito	78
dispositivo *Android*	5841	impressora	65
*	3910	dispositivo de rede	50
não identificado	1845	TV *box*	20
computador	998	câmera	15
smart TV	253	*Google Home*	3
tablet	151	dispositivo de som	2
chromecast	107	*Raspberry*	2
video game	103	*Ipod*	1
smartwatch	85		

Tabela 5.2: Resultado da primeira e segunda etapa do procedimento de identificação dos fabricantes da categoria *video game*.

Primeira Etapa	
Fabricante	**Total**
Sony Interactive Entertainment Inc.	34
Microsoft Corporation	32
Microsoft	11
Nintendo Co.,Ltd	8
Nintendo Co., Ltd.	6
Hon Hai Precision Ind. Co.,Ltd.	3
não identificado	2
Samsung Electronics Co.,Ltd	1
Motorola Mobility LLC, a Lenovo Company	1
Liteon Technology Corporation	1
Cloud Network Technology (Samoa) Limited	1
AzureWave Technology Inc.	1

Segunda Etapa	
Fabricante	**Total**
Microsoft	45
Sony	41
Nintendo	14
Hon Hai Precision Ind. Co.,Ltd.	1

A Tabela 5.2 indica claramente que utilizar apenas a primeira etapa é insuficiente para se obter uma informação precisa, pois em alguns casos, o OUI não reflete o correto fabricante do equipamento (p. ex. *Liteon Technology Corporation* e *Cloud Network Technology (Samoa) Limited*). Além disso, alguns fabricantes possuem diferentes nomes cadastrados (p. ex. *Microsoft* e *Microsoft Corporation)* e também

determinados endereços MAC não podem ser identificados pelo OUI, provavelmente devido um novo intervalo de valores atribuído à determinado fabricante.

Adicionalmente, nota-se que o procedimento proposto na segunda etapa melhora consideravelmente o resultado obtido após os filtros de OUI, pois ao final da primeira etapa foram encontrados doze fabricantes (incluindo os duplicados) e após a correlação observa-se apenas quatro, dos quais apenas um deles não corresponde a um fabricante real de *video game*. Assim, o método sugerido demonstrou ser uma técnica eficiente para aperfeiçoar a classificação dos filtros de OUI.

Destaca-se que todos os filtros utilizados para categorização do tipo de equipamento e atribuição dos fabricantes dos dispositivos basearam-se na premissa que os equipamentos vêm com o campo *hostname* com um padrão de fábrica, o qual muitas vezes não é alterado por seus usuários devido falta de interesse ou dificuldade em editar essa propriedade.

Ainda nesse contexto, mesmo que o usuário modifique o atributo *hostname* é natural que coloque um novo valor que seja de fácil reconhecimento para ele, como por exemplo seu nome, tipo de equipamento, modelo ou uma combinação deles. Além disso, o crescente número de dispositivos IoT e a diversidade de suas aplicações torna praticamente impossível identificar todos eles. Mesmo assim, os resultados obtidos com a metodologia baseada em filtros demonstrou que a proposição inicial permite a identificação da maior parte dos equipamentos coletados.

5.1.1 Caracterização e padrões de tráfego dos dispositivos IoT

Para a continuação deste trabalho foram selecionados os dispositivos pertencentes as categorias: câmera, impressora, *video game*, *chromecast* e *smart* TV.

Em seguida, para cada tipo de equipamento foram obtidas algumas estatísticas relacionadas ao seu conjunto de dados. As Figuras 5.1(a) e 5.2(a) apresentam a função de distribuição acumulada (CDF – *Cumulative Distribution Function*) por minuto do tráfego de *upload* de cada categoria e as Figuras 5.1(b) e 5.2(b) mostram a CDF do 95º percentil do tráfego de *upload* de cada dispositivo individualmente (13 câmeras, 65 impressoras, 103 *video games*, 107 *chromecasts* e 253 *smart* TVs). Os resultados da Figura 5.1 estão em bits/s e os da Figura 5.2 em pacotes/s. Além disso, a fim de possibilitar uma melhor visualização do comportamento das curvas foi utilizada escala logarítmica no eixo x em todas as figuras citadas acima.

Observa-se nas Figuras 5.1(a) e 5.2(a) que em 90% das amostras o tráfego de *upload* de todas as categorias de dispositivos foi inferior à 150 kbps e 110 pps, com exceção das câmeras que foi abaixo de 450 kbps. Por sua vez, as Figuras 5.1(b) e 5.2(b) evidenciam que 95% dos equipamentos de cada categoria possuem 95% das

medições de *upload* menores do que 1 Mbps e 420 pps, excluindo os *video games* que apresentaram valores inferiores à 1,6 Mbps e 1400 pps.

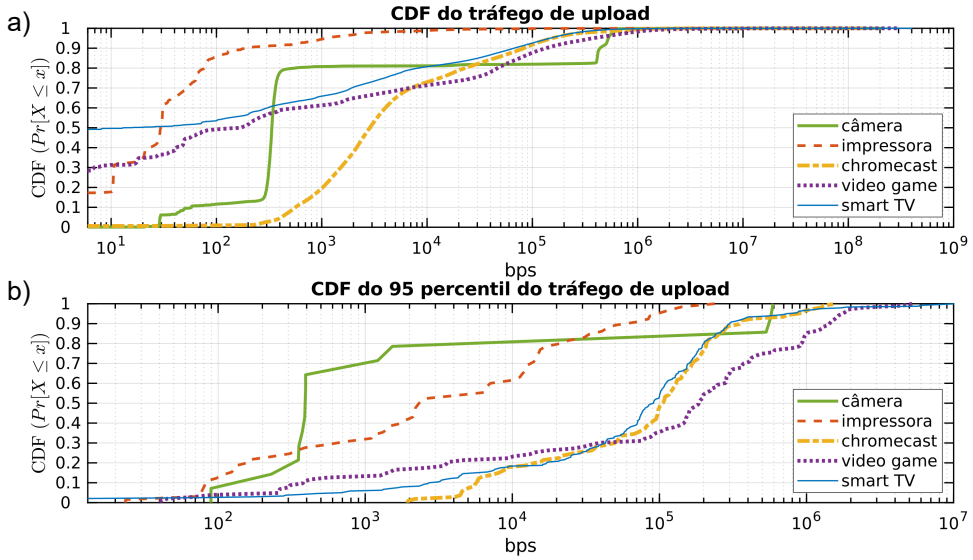

Figura 5.1: a) CDF do tráfego de *upload* por minuto para cada categoria de dispositivo. b) CDF do 95º percentil do tráfego de *upload* de todos os dispositivos pertencentes à cada uma das categorias. O eixo x está em escala logarítmica e representa a taxa em bps.

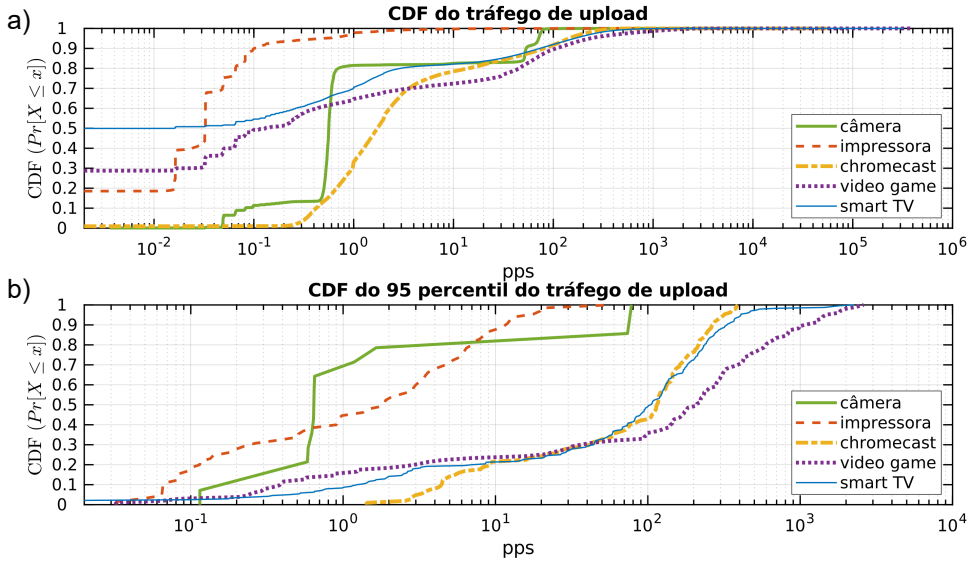

Figura 5.2: a) CDF do tráfego de *upload* por minuto para cada categoria de dispositivo. b) CDF do 95º percentil do tráfego de *upload* de todos os dispositivos pertencentes à cada uma das categorias. O eixo x está em escala logarítmica e representa a taxa em pps.

Adicionalmente, as Figuras 5.3, 5.4, 5.5, 5.6 e 5.7 ilustram a média do tráfego de *upload* e *download* por minuto em bps, para as categorias câmera, impressora, *video game*, *chromecast* e *smart* TV no período de 01/02 a 27/06/2020. Analogamente, as Figuras A.1, A.2, A.3, A.4 e A.5 (Anexo A) mostram a mediana do tráfego de *upload* e *download* das cinco categorias de dispositivos estudadas considerando o

mesmo período.

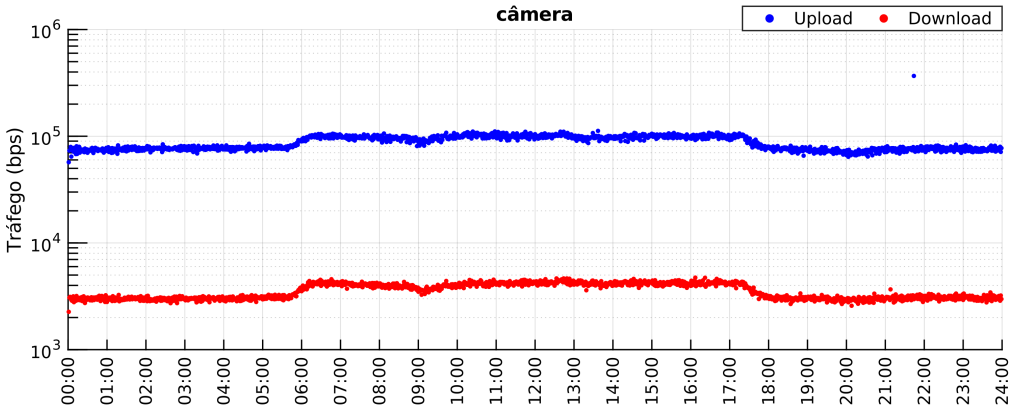

Figura 5.3: Média do tráfego de *upload* e *download* por minuto para os dispositivos categorizados como câmera no período de 01/02 a 27/06/2020. O eixo y está em escala logarítmica e representa a taxa em bps e o eixo x indica as horas ao longo do dia.

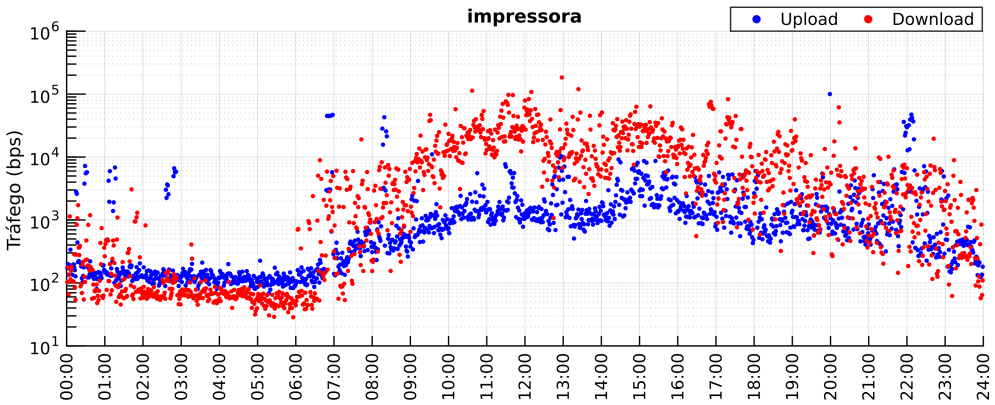

Figura 5.4: Média do tráfego de *upload* e *download* por minuto para os dispositivos categorizados como impressora no período de 01/02 a 27/06/2020. O eixo y está em escala logarítmica e representa a taxa em bps e o eixo x indica as horas ao longo do dia.

É possível notar que existe uma grande diferença na quantidade de bits enviados e recebidos por cada dispositivo, como esperado. No entanto, o padrão temporal do tráfego de *upload* e *download* é semelhante para cada categoria estudada. Além disso, constata-se que cada tipo de dispositivo possui um padrão de uso diferente ao longo do dia.

Dispositivos identificados como câmeras apresentaram maior utilização da rede de 06:00 às 17:30, chegando a taxas médias de 100 kbps para o *upload* e 4,2 kbps para o *download*. Esses picos podem ser explicados pela provável circulação maior de pessoas nesse período. Cabe ressaltar que o tráfego de *upload* da câmera é superior ao de *download* devido à esse dispositivo ter como função capturar e enviar imagens para outros dispositivos conectados a Internet.

Por outro lado, as impressores demonstraram um aumento na utilização de 07:00 às 10:00 e um uso mais intenso de 10:00 às 20:00, alcançando tráfegos médios de 2 kbps e 30 kbps para *upload* e *download*, respectivamente.

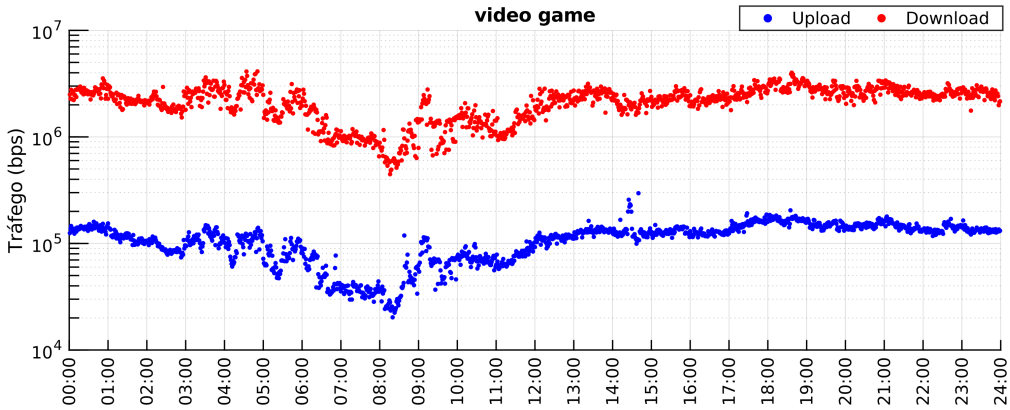

Figura 5.5: Média do tráfego de *upload* e *download* por minuto para os dispositivos categorizados como *video game* no período de 01/02 a 27/06/2020. O eixo y está em escala logarítmica e representa a taxa em bps e o eixo x indica as horas ao longo do dia.

Por sua vez, os equipamentos categorizados como *video games* indicaram um baixo consumo da rede pela manhã e um maior volume de utilização de 12:00 às 02:00, com taxas médias de 150 kbps para *upload* e 3 Mbps para *download*. As variações observadas na média do tráfego no período de 03:00 às 11:00 são justificadas pela presença de alguns *outliers* que ocorreram nesse horário e distorceram a média. Essa situação fica comprovada na Figura A.3 (Anexo A), a qual apresenta a mediana do tráfego de *upload* e *download* por minuto em bps dos *video games*.

Por fim, os aparelhos *chromecast* e *smart* TV apresentaram durante o pico de consumo taxas médias próximas, tanto de *upload* (60 kbps para ambos) quanto de *download* (1,7 Mbps e 1,6 Mbps, respectivamente). No entanto, divergiram quanto ao padrão temporal de utilização durante o dia. Enquanto os *chromecast* revelaram um uso moderado da rede de 08:30 às 21:00 (entre 0,7 e 1 Mbps de *download*) e alto volume de tráfego de 21:00 às 02:00 (1,7 Mbps de *download*), as *smart* TV indicaram um padrão de utilização praticamente constante das 08:00 às 02:00 (1,2 Mbps de *download*), com um leve aumento do consumo no período compreendido entre às 15:00 e 02:00 (1,6 Mbps de *download*). Embora sejam equipamentos destinados a mesma finalidade ficou constatado que seus proprietários os utilizam de maneira ligeiramente diferente.

Dentre os equipamentos estudados, as câmeras foram o único tipo de dispositivo à apresentar tráfego médio de *upload* superior ao de *download*. Esse fato deve ser levado em consideração para definição das *features* e criação dos modelos de detecção de ataques, pois certamente existe uma relação do tráfego de *upload* e *download* diferente para cada categoria de equipamento.

Além disso, também ficou evidenciado que os dispositivos estudados podem gerar tráfego de *upload* similar as taxas geradas pelos ataques de negação de serviço. Dessa forma, detectar ataques é uma tarefa não trivial e uma política baseada em um limiar (*threshold*) não detecta com eficiência ataques em todas as categorias de dispositivos, conforme será mostrado na Seção 5.2.6.

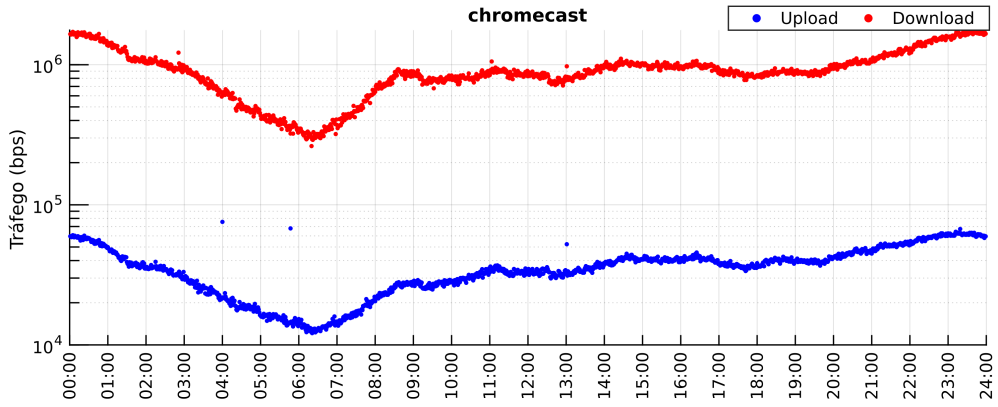

Figura 5.6: Média do tráfego de *upload* e *download* por minuto para os dispositivos categorizados como *chromecast* no período de 01/02 a 27/06/2020. O eixo y está em escala logarítmica e representa a taxa em bps e o eixo x indica as horas ao longo do dia.

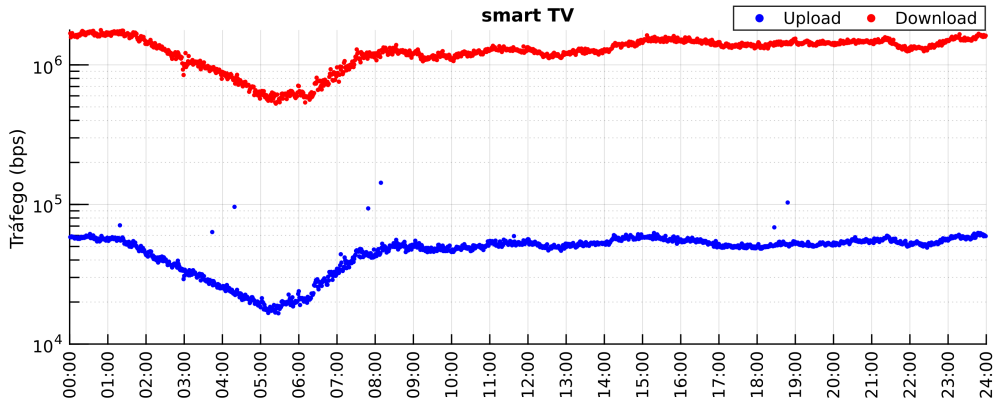

Figura 5.7: Média do tráfego de *upload* e *download* por minuto para os dispositivos categorizados como *smart* TV no período de 01/02 a 27/06/2020. O eixo y está em escala logarítmica e representa a taxa em bps e o eixo x indica as horas ao longo do dia.

5.1.2 Impacto das medidas de prevenção do coronavírus no comportamento dos usuários

Nesta seção foram analisadas as mudanças nos hábitos dos clientes do ISP provocadas pelas medidas de restrições impostas pela prefeitura de Nova Friburgo para enfrentamento da pandemia da Covid-19. Como forma de combater a disseminação

do coronavírus ficou autorizado o funcionamento apenas de serviços essenciais no município de Nova Friburgo a partir de 23 de março de 2020 [49].

Para possibilitar a análise dos impactos das medidas restritivas, o conjunto de dados de cada categoria de dispositivo (as mesmas utilizadas na seção 5.1.1) foi dividido em três períodos: 01/02 à 22/03/2020 (antes da quarentena), 23/03 à 21/05/2020 (período de suspensão das atividades) e 22/05 à 27/06/2020 (relaxamento da quarentena).

Destaca-se que a retomada gradual das atividades na cidade de Nova Friburgo iniciou em 28 de maio de 2020 [50], porém, para este estudo foi considerado a partir do dia 22 de maio com o intuito de coincidir com o período usado para testes dos modelos de detecção de ataques (seção 5.2). A influência dessa alteração é pequena, pois esse conjunto engloba quase em sua totalidade apenas o período de relaxamento da quarentena.

As alterações no comportamentos dos usuários foram avaliadas através do tráfego médio em bps para cada categoria de equipamento durante os três períodos considerados. As Figuras 5.8(a), 5.9(a), 5.10(a), 5.11(a) e 5.12(a) mostram a taxa média do tráfego de *upload* ao longo do dia para as câmeras, impressoras, *video games*, *chromecast* e *smart* TV, respectivamente. Analogamente, as Figuras 5.8(b), 5.9(b), 5.10(b), 5.11(b) e 5.12(b) ilustram a taxa de *download* para os mesmos tipos de dispositivos.

Adicionalmente, as Figuras A.6, A.7, A.8, A.9 e A.10 (Anexo A) mostram a mediana do tráfego de *upload*(a) e *download*(b) das cinco categorias de dispositivos estudadas.

Para os dispositivos categorizados como câmera, observa-se uma clara mudança em seu comportamento. Antes da quarentena esses equipamentos apresentavam tráfego médio constante (durante todo o dia) de 650 bps de *upload* e 400 bps de *download* e após o início das medidas restritivas o seu tráfego médio modificou, passando a gerar taxas de 110 kbps (4,5 kbps) de *upload* (*download*) das 18:00 às 06:00 e um pico de consumo de 06:00 às 18:00 de 150 kbps e 6 kbps de *upload* e *download*, respectivamente.

Em particular, essa mudança de comportamento foi provocada por apenas um cliente do ISP que possui três câmeras instaladas. Destaca-se que duas delas já estavam conectadas na rede desde fevereiro e geravam tráfego constante durante todo o dia, semelhante as câmeras dos outros usuários até o dia 28 de março de 2020. Por sua vez, o outro equipamento foi adicionado no dia 29 do mesmo mês e já apresentou o comportamento em dois níveis (um nível de consumo das 18:00 às 06:00 e outro de 06:00 às 18:00).

Através dos dados coletados não é possível inferir se esse cliente é residencial ou empresarial. No entanto, é possível afirmar que após o início da quarentena, o

tráfego gerado por suas câmeras sofreu uma nítida alteração no padrão de tráfego, provavelmente causado por uma maior circulação de pessoas nos locais onde as câmeras encontram-se instaladas.

Assim sendo, para esse cliente em particular, as medidas de prevenção do coronavírus provocaram mudanças em seus hábitos. Com relação aos demais usuários, acredita-se que as suas câmeras são utilizadas para monitoramento de bebês e por isso não sofreram alteração no seu tráfego médio com a quarentena.

Com relação às impressoras, nota-se que o padrão de utilização é parecido para os três períodos considerados, apresentando um maior consumo das 08:00 às 20:00. Contudo, os valores do tráfego de *upload* e *download* nessa mesma faixa de horário é bastante diferente antes e depois da quarentena. No período de 01/02 à 22/03 observa-se um consumo médio nos horários de pico de 0,5 (5,1) kbps *upload* (*download*) e posteriormente passou para valores aproximados de 2,3 kbps de *upload* e 36 kbps de *download*.

Desse modo, para estes clientes do ISP constata-se que houve um aumento de documentos impressos ocasionados provavelmente pelo *home office* após o início das medidas restritivas.

Por sua vez, para os video games foram avaliadas a média e a mediana durante os três períodos analisados, representadas nas Figuras 5.10 e A.8 (Anexo A), respectivamente. Essas duas estatísticas foram analisadas devido a presença de alguns *outliers*, conforme já relatado na Seção 5.1.1. Observa-se que o tráfego médio de *upload* e *download* é maior nos períodos 23/03 à 21/05 e 22/05 à 27/06. No entanto, a mediana do tráfego indica um consumo de *upload* e *download* parecido nos períodos anteriores as medidas de restrição (em azul) e após a liberação de algumas atividades (em verde).

Esse resultado sugere que houve uso mais intenso dos *video games* durante o período de confinamento, mas após o seu termino o padrão de uso voltou aquele pré-confinamento.

Por fim, para os equipamentos *chromecast* e *smart* TV nota-se que a partir de 23 de março de 2020, o consumo médio nas taxas de *upload* e *download* aumentou e permaneceu elevado até o período final da análise (27/06). Esse fato é um indício de que a maioria desses clientes do ISP estão ficando mais tempo em casa do que o habitual.

Os resultados alcançados nesta seção demonstraram que o isolamento social provocado pelas medidas de prevenção ao coronavírus afetaram o comportamento dos usuários e ocasionaram aumento de tráfego na rede do ISP.

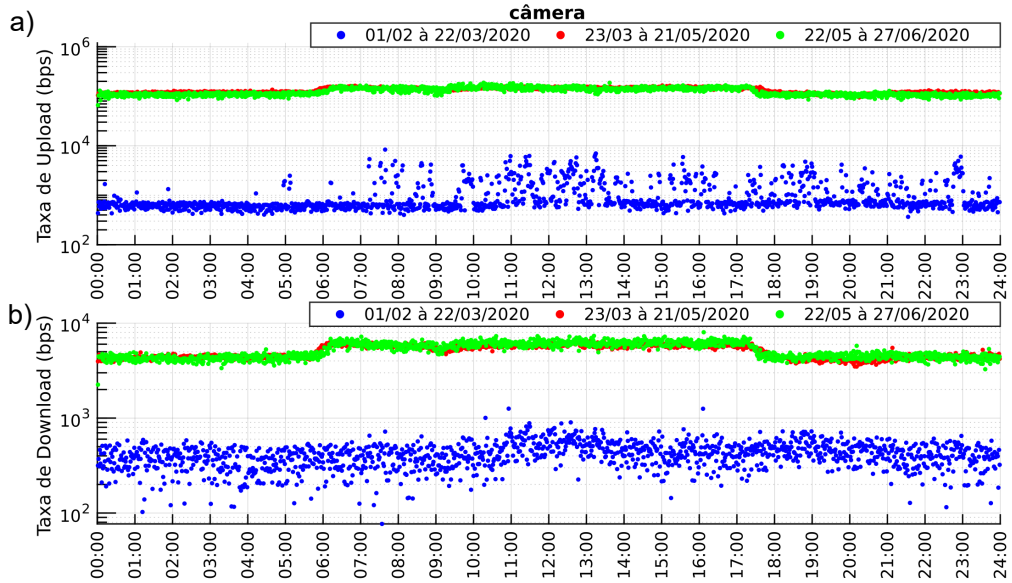

Figura 5.8: Média do tráfego por minuto para os dispositivos categorizados como câmera, considerando três períodos: 01/02 à 22/03/2020, 23/03 à 21/05/2020 e 22/05 à 27/06/2020. O eixo y está em escala logarítmica e representa a taxa em bps e o eixo x indica as horas ao longo do dia. a) Média do tráfego de *upload*. b) Média do tráfego de *download*.

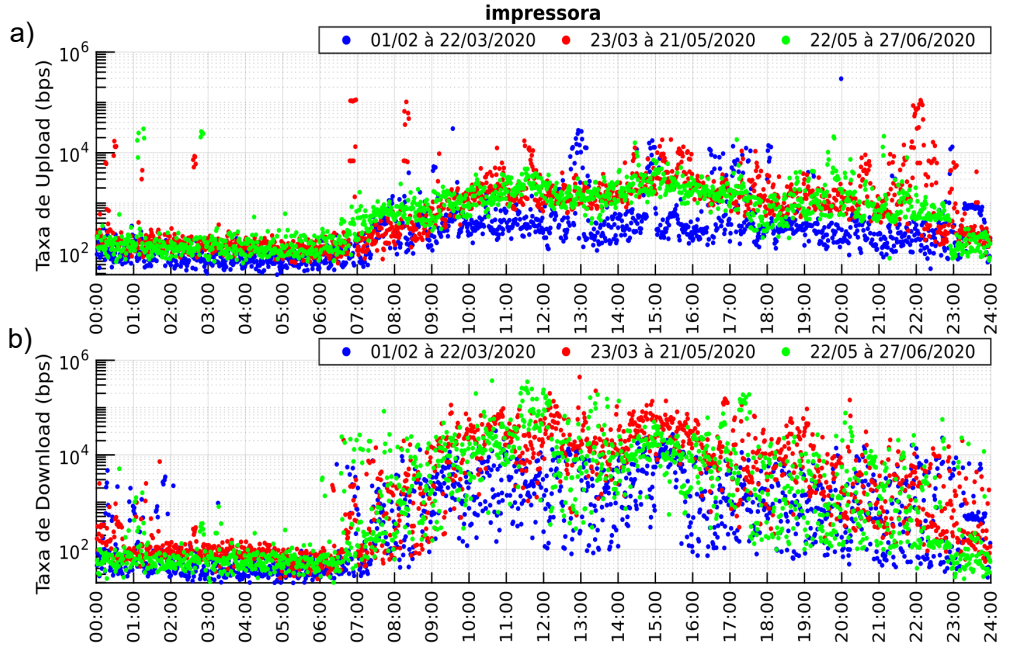

Figura 5.9: Média do tráfego por minuto para os dispositivos categorizados como impressora, considerando três períodos: 01/02 à 22/03/2020, 23/03 à 21/05/2020 e 22/05 à 27/06/2020. O eixo y está em escala logarítmica e representa a taxa em bps e o eixo x indica as horas ao longo do dia. a) Média do tráfego de *upload*. b) Média do tráfego de *download*.

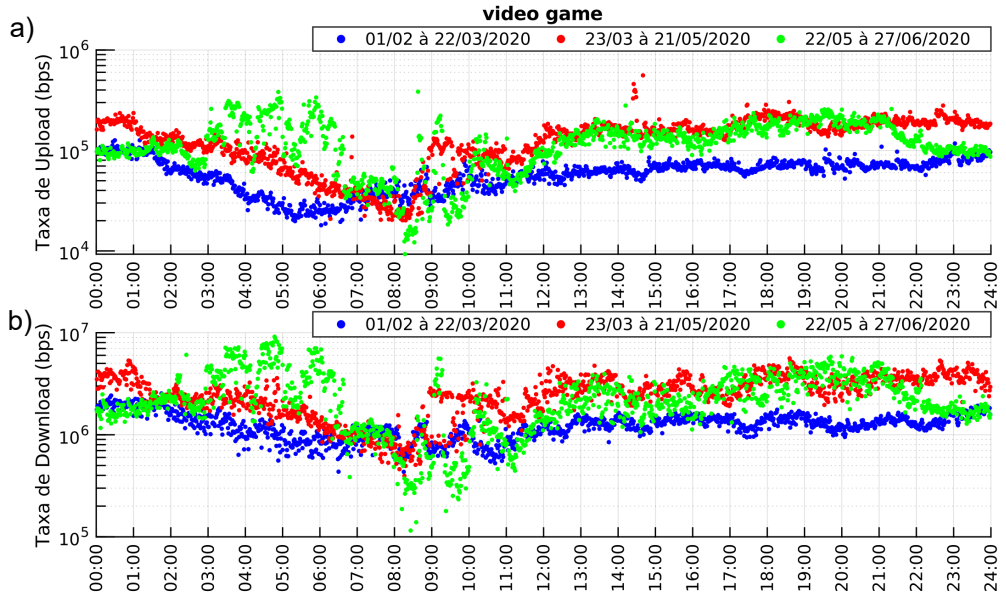

Figura 5.10: Média do tráfego por minuto para os dispositivos categorizados como *video game*, considerando três períodos: 01/02 à 22/03/2020, 23/03 à 21/05/2020 e 22/05 à 27/06/2020. O eixo y está em escala logarítmica e representa a taxa em bps e o eixo x indica as horas ao longo do dia. a) Média do tráfego de *upload*. b) Média do tráfego de *download*.

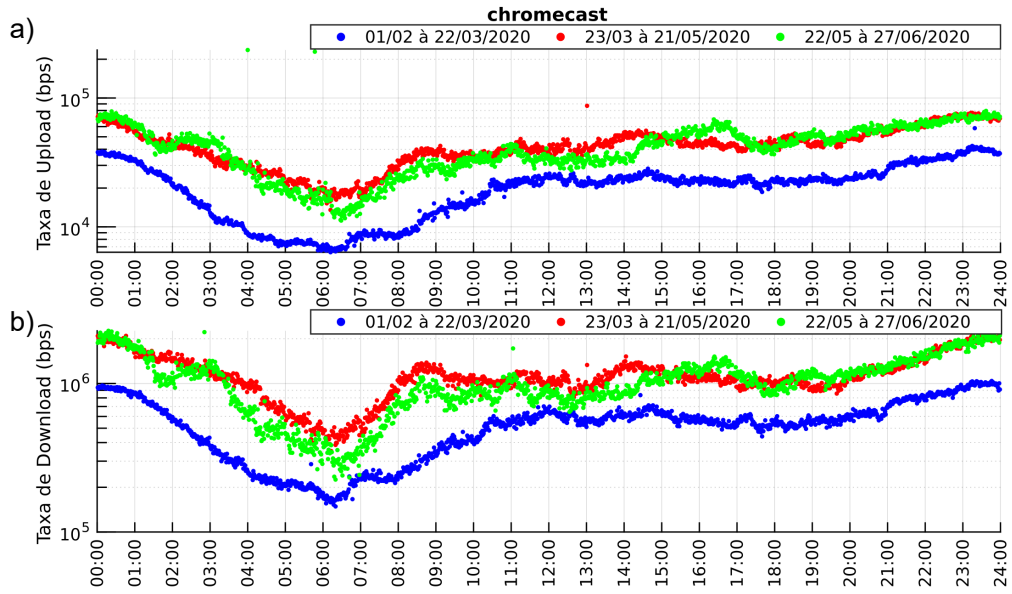

Figura 5.11: Média do tráfego por minuto para os dispositivos categorizados como *chromecast*, considerando três períodos: 01/02 à 22/03/2020, 23/03 à 21/05/2020 e 22/05 à 27/06/2020. O eixo y está em escala logarítmica e representa a taxa em bps e o eixo x indica as horas ao longo do dia. a) Média do tráfego de *upload*. b) Média do tráfego de *download*.

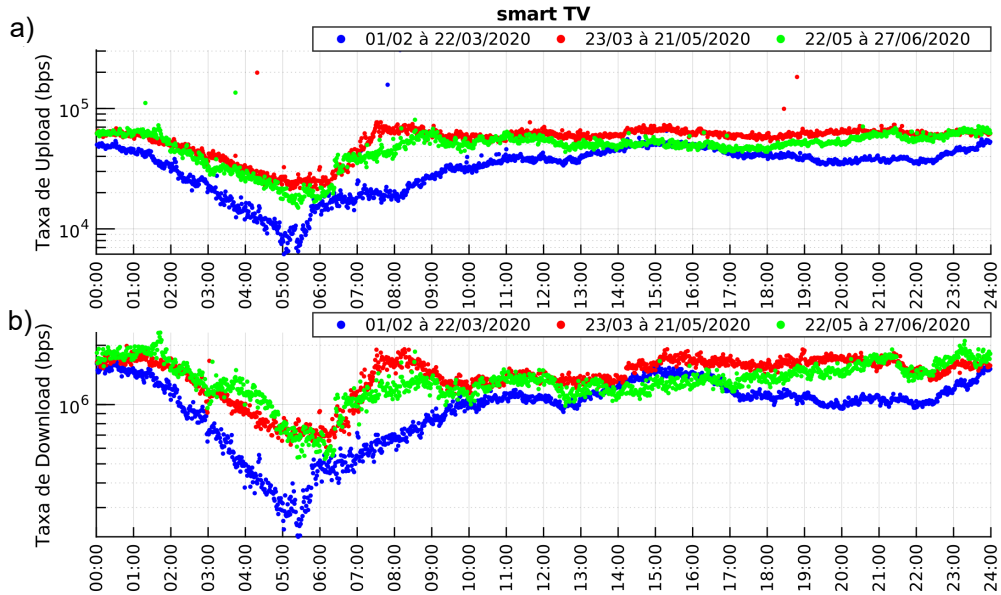

Figura 5.12: Média do tráfego por minuto para os dispositivos categorizados como *smart* TV, considerando três períodos: 01/02 à 22/03/2020, 23/03 à 21/05/2020 e 22/05 à 27/06/2020. O eixo y está em escala logarítmica e representa a taxa em bps e o eixo x indica as horas ao longo do dia. a) Média do tráfego de *upload*. b) Média do tráfego de *download*.

5.2 Detecção de ataques de negação de serviço em dispositivos IoT

Nesta seção serão apresentados os resultados do desempenho de seis modelos de aprendizado de máquina, Random Forest (RF), Árvore de Decisão (DT), Regressão Logística (LR), *Naive Bayes* Gaussiano (GNB), *Perceptron* multi-camadas (MLP) e K-vizinhos mais próximos (KNN) para identificar a presença de ataques de negação de serviço em dispositivos IoT.

Primeiramente na Seção 5.2.1 serão mostrados os conjuntos de *features* utilizados para realizar a seleção do modelo, o qual será retratado na Seção 5.2.2. Posteriormente será apresentado o resultado da avaliação do melhor classificador e da seleção de *features* para cada categoria de dispositivo (Seção 5.2.3). Na Seção 5.2.4 será exibido o ranqueamento por relevância das *features* de cada modelo.

Adicionalmente na Seção 5.2.5 serão apresentados os resultados da avaliação da capacidade de generalização dos classificadores obtidos na Seção 5.2.3. Por fim, na Seção 5.2.6 serão comparadas a performance dos melhores modelos com a utilização de políticas baseadas em *threshold*.

5.2.1 Conjuntos de *features*

Para cada categoria de dispositivo estudada foram selecionados quatro conjuntos de *features* potenciais para detecção de ataques de negação de serviço. Inicialmente o

algoritmo *Sequential Forward Selection* foi utilizado no *dataset* de treinamento para selecionar as 10 *features* mais importantes de um total de 20 *features*. A relevância de cada *feature* foi calculada a partir de um algoritmo de classificação (RF, DT ou KNN) e uma métrica de desempenho (acurácia ou *F1 score*). Também foram testados os algoritmos LR, GNB e MLP, porém o desempenho alcançado com eles foi sempre inferior aos outros três.

Esse procedimento foi repetido seis vezes para cada tipo de equipamento (utilizando um algoritmo de classificação e uma métrica diferente em cada repetição) e o resultado para cada um deles está no Apêndice B. Ressalta-se que devido a complexidade computacional, não foi possível obter as *features* através do algortimo KNN para as *smart* TV. No entanto, essa limitação não teve grande influência no modelo de detecção de ataques para essa categoria de dispositivo.

A partir do resultado anterior foram criados quatro conjuntos de *features*, conforme descrito na seção 4.3. As Tabelas 5.3, 5.4, 5.5, 5.6 e 5.7 mostram os quatro grupos formados para cada tipo de equipamento. Devido o algoritmo KNN não ter sido utilizado para a categoria *smart* TV, o conjunto das *Top 6 features* comuns para a métrica acurácia possui apenas cinco *features*.

Tabela 5.3: Conjuntos de *features* selecionadas para os dispositivos categorizados como câmera.

	Top 6 features	**Melhor algoritmo de classificação**
Acurácia	up rate pps (max) up rate bps (max - min) up rate pps (mean) up rate bps (std) up rate bps (max) up/down ratio pps (max)	up rate pps (max) up rate bps (max - min) up rate pps (mean) up rate bps (std) up/down ratio pps (std)
F1 score	up rate pps (max) up rate bps (max - min) up rate pps (mean) up rate bps (std) up/down ratio pps (max) up/down ratio bps (median)	up rate pps_max up rate bps (max - min) up rate pps (mean) up/down ratio bps (max) up rate bps (std)

5.2.2 Seleção do modelo

Para obter o melhor modelo de detecção de ataques, para cada categoria de dispositivo, os classificadores RF, DT, LR, GNB, MLP e KNN foram avaliados em quatro cenários distintos. Em cada um deles foi empregado um dos conjuntos de *features* obtido na Seção 5.2.1.

Os experimentos foram realizados utilizando validação cruzada de 5-fold no conjunto de treinamento e para cada cenário foi calculado o *F1 score* de cada classifica-

dor. O algoritmo de melhor performance nessa métrica foi eleito o vencedor de cada rodada. Após avaliar os quatro cenários, o modelo com mais vitórias foi selecionado.

A Tabela 5.8 apresenta os classificadores escolhidos para cada categoria de dispositivo e a média da métrica *F1 score* nos quatro cenários. Observa-se que a *Random Forest* foi o melhor modelo para todos os tipos de equipamentos.

Tabela 5.4: Conjuntos de *features* selecionadas para os dispositivos categorizados como impressora.

	Top 6 features	**Melhor algoritmo de classificação**
Acurácia	up rate pps (max) up rate bps (mean) up rate bps (median) up rate bps (max) up/down ratio pps (median) up rate bps (std)	up rate pps (max) up rate bps (mean) up rate bps (median) up rate bps (max) up/down ratio pps (median) up rate pps (std) up/down ratio bps (median)
F1 score	up rate pps (max) up/down ratio bps (max) up rate bps (mean) up rate bps (std) up rate bps (median) up/down ratio pps (median)	up rate pps (max) up rate bps (mean) up rate bps (median) up rate pps (mean) up/down ratio bps (max) up/down ratio pps (median) up rate bps (max - min) up rate pps (std) up/down ratio bps (mean)

Tabela 5.5: Conjuntos de *features* selecionadas para os dispositivos categorizados como *video game*.

	Top 6 features	**Melhor algoritmo de classificação**
Acurácia	up rate pps (std) up rate bps (std) up rate bps (mean) up/down ratio bps (max - min) up/down ratio bps (max) up/down ratio bps (mean)	up rate pps (std) up rate bps (max - min) up rate bps (mean) up rate bps (std) up/down ratio bps (max - min)
F1 score	up rate pps (std) up rate bps (std) up rate bps (mean) up/down ratio bps (max) up/down ratio bps (max - min) up/down ratio pps (median)	up rate pps (std) up rate bps (max - min) up rate bps (mean) up rate bps (std)

5.2.3 Avaliação dos modelos e seleção das *features*

Para cada categoria de dispositivo, o classificador selecionado na Seção 5.2.2 (*Random Forest)* foi retreinado utilizando todo o *dataset* de treinamento (111 dias) e

avaliado no *dataset* de teste (36 dias). Os dados de treinamento e teste possuem todos os vetores de ataques Mirai e Bashlite (Seção 3.2.3).

Tabela 5.6: Conjuntos de *features* selecionadas para os dispositivos categorizados como *chromecast*.

	Top 6 features	**Melhor algoritmo de classificação**
Acurácia	up rate pps (std) up rate bps (std) up/down ratio bps (std) up/down ratio bps (median) up rate bps (mean) up/down ratio bps (max)	up rate pps (std) up rate bps (std) up rate bps (mean)
F1 score	up rate pps (std) up rate bps (std) up/down ratio bps (max) up rate bps (mean) up/down ratio bps (mean) up/down ratio bps (std)	up rate pps (std) up rate bps (std) up rate bps (mean) up rate bps (max)

Tabela 5.7: Conjuntos de *features* selecionadas para os dispositivos categorizados como *smart* TV.

	Top 6 features	**Melhor algoritmo de classificação**
Acurácia	up rate pps (std) up rate bps (std) up/down ratio pps (mean) up/down ratio bps (std) up/down ratio bps (max - min)	up rate pps (std) up rate bps (std) up rate pps (max - min) up/down ratio pps (mean)
F1 score	up rate pps (std) up rate bps (std) up/down ratio pps (mean) up rate bps (mean) up/down ratio bps (median) up/down ratio bps (mean)	As mesmas features da acurácia

O desempenho dos modelos foi medido através das métricas acurácia, *precision*, *recall* (sensibilidade), *F1 score* e taxa de falso positivo (FPR – *False Positive Rate*) em quatro cenários distintos (Seção 5.2.1).

A taxa de falso positivo ou taxa de alarme falso indica a probabilidade de um ataque ser identificado em um *slot* com tráfego benigno $\left(\frac{FP}{FP+TN}\right)$. As definições das demais métricas já foram apresentadas na Seção 4.3.

Ao final dos experimentos, o conjunto de *features* que resultou no maior *F1 score* foi selecionado. A Tabela 5.9 mostra o conjunto de *features* escolhidas para cada categoria de equipamento analisada e o desempenho do modelo selecionado (*Random Forest*).

Adicionalmente foram avaliadas a probabilidade de detecção de cada vetor de ataque individualmente (*recall* ou sensibilidade). Nesse contexto, os classificadores

utilizaram o *dataset* de teste com apenas um dos vetores de ataques por vez. A Tabela 5.10 apresenta a performance de cada modelo para identificar um ataque específico.

Tabela 5.8: Resultado da seleção do melhor modelo para cada categoria de dispositivo.

Categoria	**Melhor modelo**	**Média *F1 score***
câmera	*Random Forest*	0,999476
impressora	*Random Forest*	0,998836
video game	*Random Forest*	0,997173
chromecast	*Random Forest*	0,995869
smart TV	*Random Forest*	0,997936

Tabela 5.9: Resultado do desempenho dos modelos e da seleção de features.

	Features	**Acurácia**	***Precision***	***Recall***	***F1 score***	**FPR**
câmera	up rate pps (max) up rate bps (max - min) up rate pps (mean) up rate bps (std) up/down ratio pps (std)	0,997558	0,977678	0,999105	0,988275	0,002620
impressora	up rate pps (max) up rate bps (mean) up rate bps (median) up rate bps (max) up/down ratio pps (median) up rate pps (std) up/down ratio bps (median)	0,999827	0,998342	0,997706	0,998024	0,000076
video game	up rate pps (std) up rate bps (max - min) up rate bps (mean) up rate bps (std)	0,999549	0,998842	0,994281	0,996557	0,000081
chromecast	up rate pps (std) up rate bps (std) up rate bps (mean) up rate bps (max)	0,999663	0,999665	0,993079	0,996361	0,000016
smart TV	up rate pps (std) up rate bps (std) up/down ratio pps (mean) up/down ratio bps (std) up/down ratio bps (max - min)	0,999874	0,998847	0,996239	0,997542	0,000030

Os resultados obtidos comprovaram que os classificadores são capazes de identificar uma grande quantidade de *slots* com ataques (*recall* superior à 0,99), independente do vetor de ataque empregado. Para o caso geral, todos os modelos demonstraram sua robustez, apresentando excelentes valores de acurácia (acima de 0,99), alta certeza na detecção de ataques (*precision* superiores à 0,97 e taxas de

alarme falso menores do que 0,0026).

Destaca-se em relação à *precision* que o classificador de ataques em equipamentos categorizados como câmeras foi o que apresentou um desempenho ligeiramente inferior (0,97 contra 0,99 dos outros modelos). Dessa forma, embora seja capaz de detectar mais ataques que os demais classificadores (*recall* superior à 0,999), esse modelo identifica *slots* sem ataques como malicioso com maior frequência (menor *precision*).

Em particular, essa categoria de dispositivo foi a única que gerou tráfegos de *upload* por minuto (média e mediana) superiores à taxa de *download*. Essa característica provavelmente contribuiu para uma maior dificuldade em detectar ataques corretamente, pois a *feature* (desvio padrão da razão da taxa de *upload* e *download* é a terceira mais importante para esse modelo (*Gini index* de 0,2192), praticamente empatada com a segunda, conforme será demonstrado na Seção 5.2.4.

Os classificadores empregados neste trabalho utilizaram entre quatro e sete *features*, o que representa no pior caso, aproximadamente metade das *features* usado pelo modelo proposto por MENDONÇA *et al.* [15] (doze *features*). Quanto ao desempenho dos classificadores, ele é similar nos dois trabalhos.

A grande vantagem dos modelos deste trabalho em relação ao de MENDONÇA *et al.* [15] é que eles possibilitam a identificação exata do dispositivo IoT malicioso. Dessa forma, o ISP pode efetuar ações de bloqueio apenas no equipamento infectado, evitando assim, a interrupção do tráfego de toda a rede do seu cliente.

Tabela 5.10: *Recall* por vetor de ataque.

	câmera (***recall***)	**impressora** (***recall***)	***video game*** (***recall***)	***chromecast*** (***recall***)	***smart* TV** (***recall***)
Mirai UDP *flood*	0,999086	0,996777	0,992544	0,990993	0,996067
Mirai UDP-PLAIN *flood*	0,999145	0,997285	0,994310	0,991713	0,996111
Mirai TCP SYN *flood*	0,998675	1,000000	0,994800	0,992521	0,994342
Mirai TCP ACK *flood*	1,000000	1,000000	0,996386	0,996058	0,997151
BASHLITE UDP *flood*	0,998746	0,994911	0,993512	0,994564	0,996915
BASHLITE TCP SYN *flood*	0,999551	0,996578	0,993752	0,991903	0,995517
BASHLITE TCP ACK *flood*	0,998613	0,999064	0,994819	0,993656	0.997553

5.2.4 Avaliação das *features*

No que se segue foi avaliada a importância relativa de cada *feature* utilizando a métrica *Gini index*. A Tabela 5.11 apresenta o resultado alcançado para cada categoria de dispositivo. Observa-se que para as câmeras e impressoras, o atributo mais importante foi o valor máximo da taxa de *upload* em pps (0,410 e 0,301, respectivamente). Em segundo lugar o *Gini index* indicou a média do tráfego de *upload* em

pps (0,219) para as câmeras e o desvio padrão da taxa de *upload* em pps para as impressoras (0,294).

Tabela 5.11: Resultado da métrica Gini index para cada categoria de dispositivo.

	Features	***Gini index***
câmera	up rate pps (max)	0,410320
	up rate pps (mean)	0,219606
	up/down ratio pps (std)	0,219291
	up rate bps (std)	0,148971
	up rate bps (max - min)	0,001811
impressora	up rate pps (max)	0,301594
	up rate pps (std)	0,294996
	up rate bps (max)	0,223306
	up rate bps (mean)	0,149177
	up rate bps (median)	0,026893
	up/down ratio pps (median)	0,003568
	up/down ratio bps (median)	0,000466
video game	up rate pps (std)	0,516705
	up rate bps (std)	0,283804
	up rate bps (max - min)	0,189322
	up rate bps (mean)	0,010169
chromecast	up rate pps (std)	0,543268
	up rate bps (std)	0,292599
	up rate bps (max)	0,160630
	up rate bps (mean)	0,003503
smart TV	up rate pps (std)	0,489741
	up rate bps (std)	0,284236
	up/down ratio bps (max - min)	0,109531
	up/down ratio bps (std)	0,096531
	up/down ratio pps (mean)	0,019961

Por sua vez, para as categorias *video game*, *chromecast* e *smart* TV as *features* mais relevantes foram o desvio padrão do tráfego de *upload* em pps (0,516; 0,543 e 0,489, respectivamente), seguido pelo desvio padrão da taxa de *upload* em bps (0,283; 0,292 e 0,284, respectivamente).

Nota-se que as *features* relacionadas a estatística desvio padrão estão sempre entre as melhores ranqueadas. Assim sendo, os resultados obtidos indicam que a variabilidade do tráfego dentro de cada *slot* em relação à sua média é bastante relevante para identificar ataques nos equipamentos estudados.

5.2.5 Detecção de ataques desconhecidos

Com o intuito de avaliar o desempenho do classificador para identificação de ataques inéditos, inicialmente foi realizado o treinamento do modelo em um *dataset* contendo apenas vetores de ataques BASHLITE e posteriormente o classificador foi testado em um *dataset* com somente ataques Mirai. Essa configuração foi utilizada devido a botnet Mirai ter sido criada após a BASHLITE [2]. Dessa forma, os experimentos realizados simulam o surgimento de um novo *malware*.

A Tabela 5.12 mostra a performance dos modelos para detecção de ataques desconhecidos. Como esperado, o desempenho geral reduziu, porém nota-se que os classificadores continuam com uma alta taxa de identificação de ataques (*recall* elevado). No entanto, o modelo dos video-games foi o que apresentou maior variação na sua sensibilidade, variando o *recall* de 0,993 (com o treinamento utilizando ataques Mirai e BASHLITE) para 0,964 (com o treinamento usando somente o BASHLITE). Quanto a *precision*, os valores obtidos são similares aos dos testes realizados na Seção 5.2.3 quando foram empregados todos os tipos de ataques no treinamento. Esses valores são provocados pelo baixo número de falsos positivos.

Por sua vez, a Tabela 5.13 ilustra o desempenho dos classificadores para detectar individualmente cada vetor de ataque Mirai (*dataset* de teste com apenas um tipo de ataque por vez).

Tabela 5.12: Resultado do desempenho dos modelos para detecção ataques desconhecidos. O classificador foi treinado utilizando um *dataset* com ataques BASHLITE e testado em um *dataset* com ataques Mirai.

	Acurácia	***Precision***	***Recall***	***F1 score***	**FPR**
câmera	0,997313	0,959766	0,997858	0,978441	0,002722
impressora	0,999926	0,999548	0,997517	0,998531	0,000012
video game	0,998614	0,999073	0,964997	0,981739	0,000036
chromecast	0,999766	0,999452	0,991843	0,995633	0,000015
smart TV	0,999890	0,998892	0,993640	0,996259	0,000016

Tabela 5.13: *Recall* por vetor de ataque para ataques desconhecidos.

	câmera (recall)	***impressora (recall)***	***video game (recall)***	***chromecast (recall)***	***smart TV (recall)***
Mirai UDP flood	0,997257	0,994359	0,983940	0,989706	0,994297
Mirai UDP-PLAIN flood	0,998717	0,997285	0,989758	0,991160	0,994849
Mirai TCP SYN flood	0,998234	0,999112	0,882028	0,990919	0,989974
Mirai TCP ACK flood	0,998075	1,000000	0,994880	0,995521	0,996267

Os resultados indicam que para algumas categorias de dispositivos, os ataques Mirai UDP *flood* foram os mais difíceis de identificar, enquanto para outros equipamentos foi o Mirai TCP SYN *flood*. Nesse cenário, destaca-se o desempenho inferior

do classificador (*video-game*) para detectar ataques TCP SYN *flood* (*recall* de 0,88).

Essa dificuldade pode ser explicada pelo fato do *video-game* ser a categoria de dispositivos que possui a maior quantidade de equipamentos com tráfego de *upload* elevado, tanto em bps quanto em pps. Além disso, conforme relatado por MENDONÇA *et al.* [14], a assinatura dos ataques TCP SYN *flood* BASHLITE e Mirai são bem diferentes. Esse fato fica bem claro quando é analisado o tráfego gerado por cada um deles.

Analisando o boxplot do TCP SYN *flood* do BASHLITE e do Mirai (Figura 3.4), nota-se que o 25º percentil da taxa de *upload* do TCP SYN *flood* do Mirai em bps é superior ao 50º percentil (mediana) do BASHLITE. Outrossim, o intervalo interquartílico do TCP SYN *flood* do Mirai é maior que o do BASHLITE tanto no tráfego de *upload* em bps quanto em pps.

Adicionalmente destaca-se o ótimo desempenho dos classificadores na identificação de ataques não implementados no BASHLITE, como o Mirai UDP-PLAIN *flood*, indicando que os modelos possuem um bom poder de generalização. Ainda assim, quanto mais variantes de ataques forem utilizadas para treinamento dos modelos, mais robusto e eficiente ele ficará.

5.2.6 Avaliação de políticas de *threshold*

No que se segue foram realizados experimentos para testar o desempenho de políticas de *threshold* para detectar ataques de negação de serviço em dispositivos IoT. Nesta Seção foram testados valores de *thresholds* variando do 5º até o 95º percentil (com incremento de 5) da taxa de *upload* em bps e pps, considerando o tráfego gerado por todos os vetores de ataques estudados.

Exemplificando, para a taxa de *upload* em bps, agrupamos em um mesmo conjunto, todas as amostras (por minuto) do tráfego de *upload* em bps de todos os sete vetores de ataques. Em seguida, calculamos os percentis (5º, 10º, 15º, ..., 95º) desse conjunto de dados. Esse mesmo procedimento foi repetido para a taxa de *upload* em pps.

O modelo de detecção baseado em políticas de *threshold* considera que existe um ataque em uma amostra específica se ela possuir valores acima do limiar. Inicialmente foram realizados experimentos considerando apenas um *threshold* (em bps ou pps) e posteriormente foi utilizada a abordagem com os dois limiares.

A Tabela 5.14 mostra o melhor resultado do *F1 score* para cada categoria de dispositivo, utilizando o modelo de detecção baseado em políticas de *threshold*. Nota-se um *F1-score* médio de 0,924 contra 0,995 com a *Random Forest* (Tabela 5.9). Em particular, para a categoria *video-game* a performance caiu consideravelmente (*F1-score* de 0,763). Assim sendo, essa abordagem de detecção possui desempenho

inferior quando comparada à *Random Forest*.

Para a categoria *video-game* a melhor política foi utilizar o *threshold* do 5º percentil em bits/s (840 kbps) e do 35º percentil em pacotes/s (2684 pps). Para todos os outros equipamentos as melhores performances foram obtidas com apenas o limiar do 5º percentil em pacotes/s (1051 pps). Caso esse mesmo *threshold* seja aplicado nos *video-games*, a taxa de detecção de ataques aumenta (*recall* de 0,9484), porém tráfegos benignos são classificados como maliciosos com mais frequência (*precision* de 0,5960).

Adicionalmente, a Figura 5.13 exemplifica o que ocorre quando utilizamos maiores valores de *threshold* tanto em bps (Figura 5.13(a)) quanto em pps (Figura 5.13(b)). Esse comportamento foi observado em todas as categorias estudadas, ou seja, considerar maiores valores de *threshold* tanto em bps quanto em pps, provocam uma diminuição na quantidade de ataques detectados e na quantidade de tráfego legítimo confundido como malicioso.

Tabela 5.14: Resultado do desempenho dos modelos com políticas de threshold

	Acurácia	***Precision***	***Recall***	***F1 score***	**FPR**
câmera	0,998172	1	0,945613	0,972046	0
impressora	0,999237	1	0,945337	0,971900	0
video game	0,991769	0,951733	0,637033	0,763216	0.000687
chromecast	0,999236	0,999314	0,950424	0,974256	0.000010
smart TV	0,999000	0,932638	0,951936	0,942188	0.000593

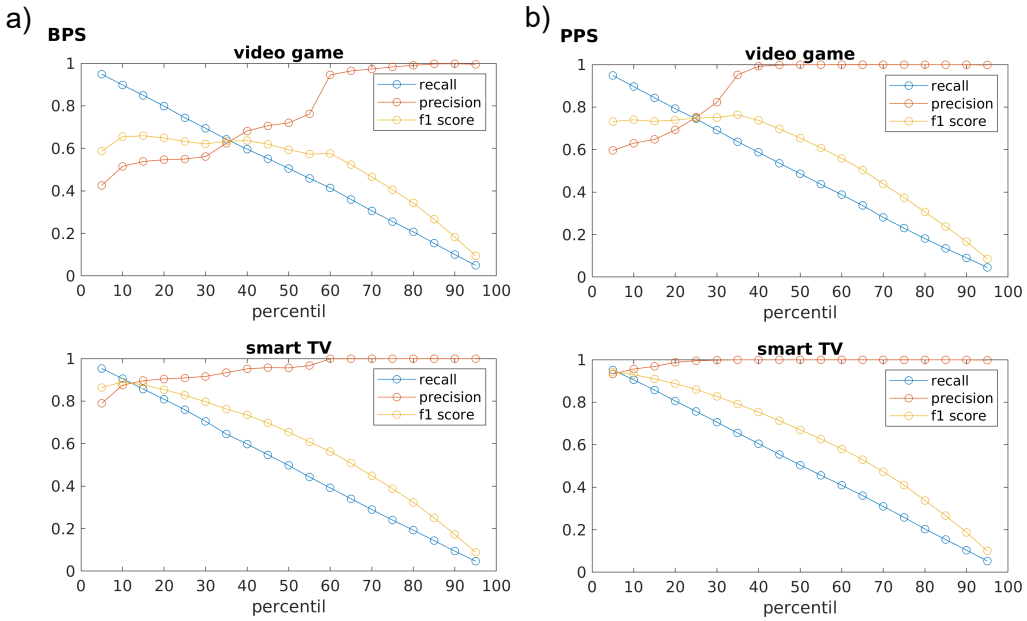

Figura 5.13: Desempenho do modelo com diferentes valores de *threshold* para as categorias *video game* e *smart* TV. O eixo x representa o percentil usado como *threshold* e o eixo y a performance das métricas. a) *Threshold* em bps. b) *Threshold* em pps.

Capítulo 6

Conclusões

O crescimento acelerado do número de dispositivos IoT conectados à Internet, aliado a vulnerabilidade ainda presente nesse tipo de equipamento e a potencialidade de emprego desses dispositivos em *botnets* capazes de realizar ataques DDoS em larga escala, motivam diversos estudos nessa área. Além disso, os *hackers* estão sempre criando novas estratégias para infecção dos dispositivos e também inovando nas formas de efetuar seus ataques. Esse contexto levanta a necessidade do desenvolvimento de novas ferramentas para identificar essas ameaças e mitigar os efeitos dos seus ataques.

Neste trabalho foi proposta uma abordagem simples, de baixo custo computacional, para detectar ataques de negação de serviço em dispositivos IoT, sem interferir na privacidade dos usuários. A metodologia proposta utiliza técnicas de aprendizado de máquina supervisionadas para identificar tráfego malicioso no conjunto de dados a partir de informações do contador de bytes e pacotes de uma categoria de equipamento IoT específica. Dessa forma, o método proposto cria um modelo de detecção de ataques para cada tipo de dispositivo IoT.

Em parceria com um ISP foram coletados dados de tráfego a cada minuto de 806 clientes durante um período de cinco meses. O *dataset* obtido possui dados de 20.036 equipamentos de diferentes naturezas. Através de experimentos laboratoriais foram simulados diversos ataques de negação de serviço a partir do código fonte das *botnets* de IoT BASHLITE e Mirai, criando assim um conjunto de dados de ataques. Em seguida foi criado um novo *dataset* com o tráfego agregado desses dois conjunto de dados, o qual foi utilizado para treinamento dos modelos.

A fim de permitir a avaliação do método foi realizada a categorização dos equipamentos existentes no *dataset* a partir dos endereços MAC e dos *hostnames* dos dispositivos. Os resultados mostraram que essa abordagem permitiu identificar a maioria dos equipamentos.

Para o desenvolvimento deste trabalho foram selecionadas as categorias câmera, impressora, *video game*, *chromecast* e *smart* TV. Devido o padrão de tráfego de

cada uma delas, também foi definida uma metodologia para selecionar conjuntos de *features* mais relevantes para cada tipo de dispositivo estudado. Essa técnica permitiu reduzir o custo computacional dos modelos e ainda assim produziu excelentes resultados.

Diferentes modelos de aprendizado de máquina foram testados para as categorias escolhidas e o algoritmo da *Random Forest* foi o que apresentou o melhor desempenho para todas elas. Os resultados demostraram uma boa performance dos classificadores para diferentes vetores de ataques utilizando entre quatro e sete *featuares*. Além disso, os modelos também apresentaram um bom poder de generalização ao detectar com eficiência ataques não existentes no conjunto de treinamento. Dessa forma, a solução proposta se mostrou promissora para identificar novas variantes ou novos tipos de *botnets*. Por fim, a abordagem proposta também apresentou melhor performance do que políticas baseadas em *threshold*, o que comprova que a detecção de ataques DDoS não é trivial.

As perspectivas deste trabalhos incluem: a) reduzir o intervalo de coleta dos dados de minutos para segundos e avaliar o método nesse novo cenário; b) combinar o modelo desenvolvido por MENDONÇA *et al.* [15] com o deste trabalho e avaliar a efetividade dessa abordagem (detecção *online* e *offline*); c) avaliar critérios para definir quando o modelo precisa ser retreinado devido alteração no padrão do tráfego dos dispositivos.

Referências Bibliográficas

[1] ANTONAKAKIS, M., APRIL, T., BAILEY, M., et al. "Understanding the Mirai Botnet". In: *26th USENIX Security Symposium (USENIX Security 17)*, pp. 1093–1110, Vancouver, BC, ago. 2017. USENIX Association. ISBN: 978-1-931971-40-9. Disponível em: <https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis>.

[2] MARZANO, A., ALEXANDER, D., FONSECA, O., et al. "The Evolution of Bashlite and Mirai IoT Botnets". In: *2018 IEEE Symposium on Computers and Communications (ISCC)*, pp. 00813–00818, 2018. doi: 10.1109/ISCC.2018.8538636.

[3] ASHTON, K. "That "Internet of Things" Thing", *RFID Journal*, v. 22, 2009. Disponível em: <https://ci.nii.ac.jp/naid/20000912385/en/>.

[4] IDC. "Worldwide Spending on the Internet of Things Will Slow in 2020 Then Return to Double-Digit Growth, According to a New IDC Spending Guide". 2020. Disponível em: <https://www.idc.com/getdoc.jsp?containerId=prUS46609320>. Acessado em: 15/12/2020.

[5] LUETH, K. L. "State of the IoT 2020: 12 billion IoT connections, surpassing non-IoT for the first time". 2020. Disponível em: <https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/>. Acessado em: 15/12/2020.

[6] IDC. "IoT Growth Demands Rethink of Long-Term Storage Strategies, says IDC". 2020. Disponível em: <https://www.idc.com/getdoc.jsp?containerId=prUS46609320>. Acessado em: 15/12/2020.

[7] ROOPAK, M., TIAN, G. Y., CHAMBERS, J. "An Intrusion Detection System Against DDoS Attacks in IoT Networks". In: *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0562–0567, 2020. doi: 10.1109/CCWC47524.2020.9031206.

[8] KOLIAS, C., KAMBOURAKIS, G., STAVROU, A., et al. "DDoS in the IoT: Mirai and other botnets", *Computer*, v. 50, pp. 80–84, 01 2017. doi: 10.1109/MC.2017.201.

[9] CISCO. "Cisco Annual Internet Report (2018–2023) White Paper". 2020. Disponível em: <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html>. Acessado em: 20/02/2021.

[10] MENSCHER, D. "Exponential growth in DDoS attack volumes". 2020. Disponível em: <https://cloud.google.com/blog/products/identity-security/identifying-and-protecting-against-the-largest-ddos-attacks>. Acessado em: 20/02/2021.

[11] SHIELD, A. "Threat Landscape Report – Q1 2020". 2020. Disponível em: <https://aws-shield-tlr.s3.amazonaws.com/2020-Q1_AWS_Shield_TLR.pdf>. Acessado em: 20/02/2021.

[12] KOTTLER, S. "February 28th DDoS Incident Report". 2018. Disponível em: <https://github.blog/2018-03-01-ddos-incident-report/>. Acessado em: 20/02/2021.

[13] AKAMAI. "State of the Internet/Segurança de 2021: Como se adaptar ao imprevisível". 2021. Disponível em: <https://www.akamai.com/br/pt/multimedia/documents/state-of-the-internet/soti-security-a-year-in-review-report-2020.pdf>. Acessado em: 19/04/2021.

[14] MENDONÇA, G., SANTOS, G., DE SOUZA E SILVA, E., et al. "Uma abordagem para detecção de DDoS a partir de roteadores domésticos". In: *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pp. 834–847, Porto Alegre, RS, Brasil, 2019. SBC. doi: 10.5753/sbrc.2019.7406. Disponível em: <https://sol.sbc.org.br/index.php/sbrc/article/view/7406>.

[15] MENDONÇA, G., SANTOS, G. H. A., DE SOUZA E SILVA, E., et al. "An Extremely Lightweight Approach for DDoS Detection at Home Gateways". In: *2019 IEEE International Conference on Big Data (Big Data)*, pp. 5012–5021, 2019. doi: 10.1109/BigData47090.2019.9006548.

[16] AKAMAI. "Q3 2016 State of the Internet – Security Report". 2016. Disponível em: <https://www.akamai.com/uk/en/multimedia/documents/state-of-the-internet/q3-2016-state-of-the-internet-security-report.pdf>. Acessado em: 19/01/2020.

[17] AKAMAI. "2020 State of the Internet / Security: Gaming — You Can't Solo Security". 2020. Disponível em: <https://www.akamai.com/br/pt/multimedia/documents/state-of-the-internet/soti-security-gaming-you-cant-solo-security-report-2020.pdf>. Acessado em: 19/01/2020.

[18] GUPTA, B., BADVE, O. P. "Taxonomy of DoS and DDoS attacks and desirable defense mechanism in a cloud computing environment", *Neural Computing and Applications*, v. 28, n. 12, pp. 3655–3682, 2017.

[19] CHANG, W., MOHAISEN, A., WANG, A., et al. "Understanding adversarial strategies from bot recruitment to scheduling". In: *International Conference on Security and Privacy in Communication Systems*, pp. 397–417. Springer, 2018.

[20] ÖZÇELIK, M., CHALABIANLOO, N., GÜR, G. "Software-Defined Edge Defense Against IoT-Based DDoS". In: *2017 IEEE International Conference on Computer and Information Technology (CIT)*, pp. 308–313, 2017. doi: 10.1109/CIT.2017.61.

[21] NEIRA, A., MEDEIRO, A., NOGUEIRA, M. "Identificação Antecipada de Botnets por Aprendizagem de Máquina". In: *Anais do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pp. 896–909, Porto Alegre, RS, Brasil, 2020. SBC. doi: 10.5753/sbrc.2020.12333. Disponível em: <https://sol.sbc.org.br/index.php/sbrc/article/view/12333>.

[22] LU, L., FENG, Y., SAKURAI, K. "C&C Session Detection Using Random Forest". In: *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*, IMCOM '17, New York, NY, USA, 2017. Association for Computing Machinery. ISBN: 9781450348881. doi: 10.1145/3022227.3022260. Disponível em: <https://doi.org/10.1145/3022227.3022260>.

[23] MEIDAN, Y., BOHADANA, M., MATHOV, Y., et al. "N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders", *IEEE Pervasive Computing*, v. 17, n. 3, pp. 12–22, 2018. doi: 10.1109/MPRV.2018.03367731.

[24] HOLBROOK, L., ALAMANIOTIS, M. "Internet of Things Security Analytics and Solutions with Deep Learning". In: *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 178–185, 2019. doi: 10.1109/ICTAI.2019.00033.

[25] KIM, J., SHIM, M., HONG, S., et al. "Intelligent Detection of IoT Botnets Using Machine Learning and Deep Learning", *Applied Sciences*, v. 10, n. 19, 2020. ISSN: 2076-3417. doi: 10.3390/app10197009. Disponível em: <https://www.mdpi.com/2076-3417/10/19/7009>.

[26] AMARASINGHE, K., KENNEY, K., MANIC, M. "Toward Explainable Deep Neural Network Based Anomaly Detection". In: *2018 11th International Conference on Human System Interaction (HSI)*, pp. 311–317, 2018. doi: 10.1109/HSI.2018.8430788.

[27] HWANG, R.-H., PENG, M.-C., HUANG, C.-W. "Detecting IoT Malicious Traffic Based on Autoencoder and Convolutional Neural Network". In: *2019 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, 2019. doi: 10.1109/GCWkshps45667.2019.9024425.

[28] ROOPAK, M., YUN TIAN, G., CHAMBERS, J. "Deep Learning Models for Cyber Security in IoT Networks". In: *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0452–0457, 2019. doi: 10.1109/CCWC.2019.8666588.

[29] DOSHI, R., APTHORPE, N., FEAMSTER, N. "Machine Learning DDoS Detection for Consumer Internet of Things Devices". In: *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 29–35, 2018. doi: 10.1109/SPW.2018.00013.

[30] CHAUDHARY, P., GUPTA, B. B. "DDoS Detection Framework in Resource Constrained Internet of Things Domain". In: *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*, pp. 675–678, 2019. doi: 10.1109/GCCE46687.2019.9015465.

[31] HAMZA, A., GHARAKHEILI, H. H., BENSON, T. A., et al. "Detecting Volumetric Attacks on LoT Devices via SDN-Based Monitoring of MUD Activity". In: *Proceedings of the 2019 ACM Symposium on SDN Research*, SOSR '19, p. 36–48, New York, NY, USA, 2019. Association for Computing Machinery. ISBN: 9781450367103. doi: 10.1145/3314148.3314352. Disponível em: <https://doi.org/10.1145/3314148.3314352>.

[32] ANLIX. "Anlix". 2020. Disponível em: <https://anlix.io/>. Acessado em: 10/04/2020.

[33] GIGALINK. "Gigalink". 2020. Disponível em: <https://www.gigalink.com.br/internet-residencial>. Acessado em: 12/06/2020.

[34] OPENWRT. "OpenWrt: Wireless Freedom". 2020. Disponível em: <https://openwrt.org/about>. Acessado em: 15/12/2020.

[35] KALKAN, K., GÜR, G., ALAGÖZ, F. "Filtering-Based Defense Mechanisms Against DDoS Attacks: A Survey", *IEEE Systems Journal*, v. 11, n. 4, pp. 2761–2773, 2017. doi: 10.1109/JSYST.2016.2602848.

[36] SILVA, S. S. C., SILVA, R. M. P., PINTO, R. C. G., et al. "Botnets: A Survey", *Comput. Netw.*, v. 57, n. 2, pp. 378–403, fev. 2013. ISSN: 1389-1286. doi: 10.1016/j.comnet.2012.07.021. Disponível em: <https://doi.org/10.1016/j.comnet.2012.07.021>.

[37] ANGRISHI, K. "Turning Internet of Things(IoT) into Internet of Vulnerabilities (IoV) : IoT Botnets". 2017.

[38] MARZANO, A., ALEXANDER, D., FAZZION, E., et al. "Monitoramento e Caracterização de Botnets Bashlite em Dispositivos IoT". In: *Anais do XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, Porto Alegre, RS, Brasil, 2018. SBC. Disponível em: <https://sol.sbc.org.br/index.php/sbrc/article/view/2479>.

[39] CERON, J. M., STEDING-JESSEN, K., HOEPERS, C., et al. "Improving IoT Botnet Investigation Using an Adaptive Network Layer", *Sensors*, v. 19, n. 3, 2019. ISSN: 1424-8220. doi: 10.3390/s19030727. Disponível em: <https://www.mdpi.com/1424-8220/19/3/727>.

[40] DING, F. "IoT Malware". 2017. Disponível em: <https://github.com/ifding/iot-malware>. Acessado em: 06/05/2020.

[41] MARGOLIS, J., OH, T. T., JADHAV, S., et al. "An In-Depth Analysis of the Mirai Botnet". In: *2017 International Conference on Software Security and Assurance (ICSSA)*, pp. 6–12, 2017. doi: 10.1109/ICSSA.2017.12.

[42] HARIS, S. H. C., AHMAD, R. B., GHANI, M. A. H. A. "Detecting TCP SYN Flood Attack Based on Anomaly Detection". In: *2010 Second International Conference on Network Applications, Protocols and Services*, pp. 240–244, 2010. doi: 10.1109/NETAPPS.2010.50.

[43] BLENN, N., GHIËTTE, V., DOERR, C. "Quantifying the Spectrum of Denial-of-Service Attacks through Internet Backscatter". In: *Proceedings of the 12th International Conference on Availability, Reliability and Security*, ARES '17, New York, NY, USA, 2017. Association for Computing Machinery. ISBN: 9781450352574. doi: 10.1145/3098954.3098985. Disponível em: <https://doi.org/10.1145/3098954.3098985>.

[44] IEEE. "IEEE OUI". 2020. Disponível em: <http://standards-oui.ieee.org/oui/oui.txt>. Acessado em: 14/03/2020.

[45] AUCHARD, E. "German internet outage was failed botnet attempt: report". 2016. Disponível em: <https://www.reuters.com/article/us-deutsche-telekom-outages-idUSKBN13N12K>. Acessado em: 20/04/2020.

[46] SARANYA, C., MANIKANDAN, G. "A study on normalization techniques for privacy preserving data mining", *International Journal of Engineering and Technology (IJET)*, v. 5, n. 3, pp. 2701–2704, 2013.

[47] RASCHKA, S. "MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack", *The Journal of Open Source Software*, v. 3, n. 24, abr. 2018. doi: 10.21105/joss.00638. Disponível em: <http://joss.theoj.org/papers/10.21105/joss.00638>.

[48] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., et al. "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, v. 12, pp. 2825–2830, 2011.

[49] MUNICIPAL DE NOVA FRIBURGO, P. "Decreto nº 515 de 20 de março de 2020". 2020. Disponível em: <https://plenussistemas.dioenet.com.br/public/uploads/diarios/2020/03/c288aba82af627ae758898ccac08e3c4.pdf>. Acessado em: 18/08/2020.

[50] MUNICIPAL DE NOVA FRIBURGO, P. "Decreto n° 591 de 28 de maio de 2020." 2020. Disponível em: <https://plenussistemas.dioenet.com.br/public/uploads/diarios/2020/05/bdbac95d5b6ed9e4481183542d586882.pdf>. Acessado em: 18/08/2020.

Apêndice A

Mediana do tráfego dos dispositivos IoT

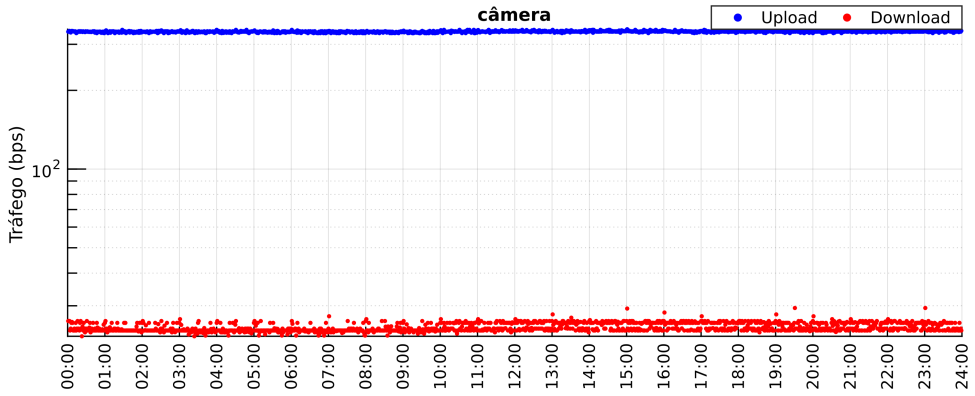

Figura A.1: Mediana do tráfego de *upload* e *download* por minuto para os dispositivos categorizados como câmera, no período de 01/02 a 27/06/2020. O eixo y está em escala logarítmica e representa a taxa em bps e o eixo x indica as horas ao longo do dia.

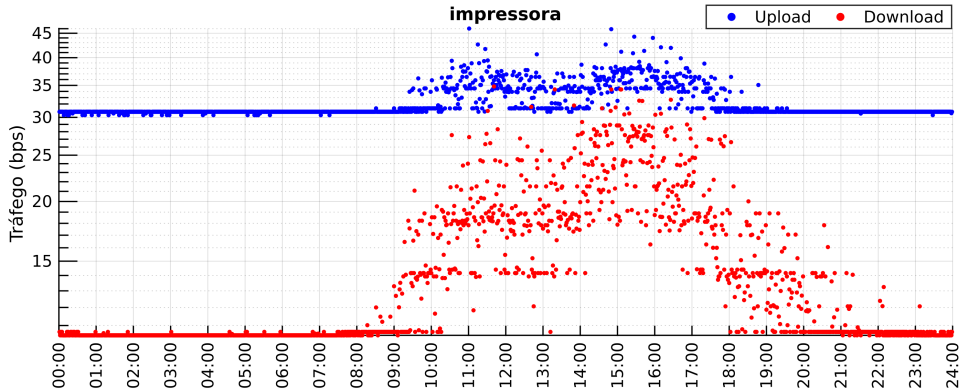

Figura A.2: Mediana do tráfego de *upload* e *download* por minuto para os dispositivos categorizados como impressora, no período de 01/02 a 27/06/2020. O eixo y está em escala logarítmica e representa a taxa em bps e o eixo x indica as horas ao longo do dia.

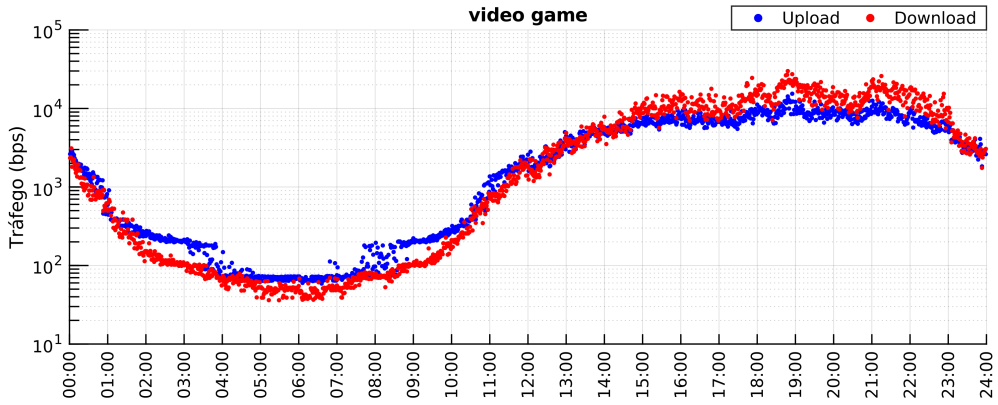

Figura A.3: Mediana do tráfego de *upload* e *download* por minuto para os dispositivos categorizados como *video game*, no período de 01/02 a 27/06/2020. O eixo y está em escala logarítmica e representa a taxa em bps e o eixo x indica as horas ao longo do dia.

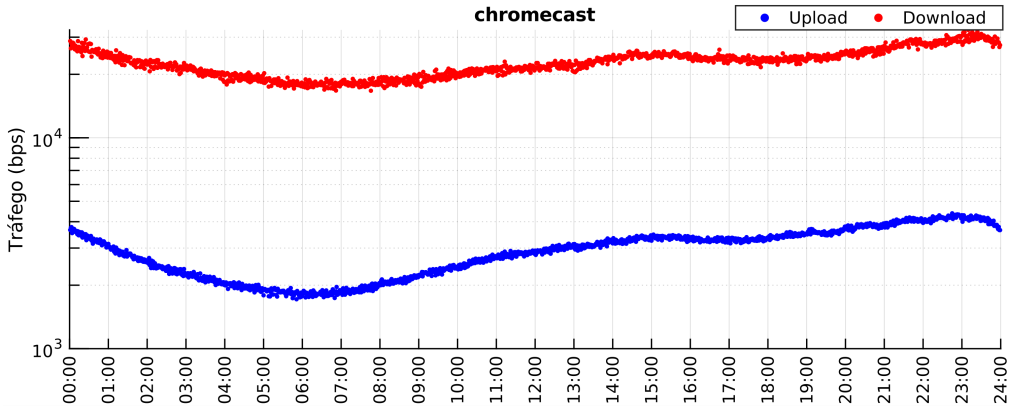

Figura A.4: Mediana do tráfego de *upload* e *download* por minuto para os dispositivos categorizados como *chromecast*, no período de 01/02 a 27/06/2020. O eixo y está em escala logarítmica e representa a taxa em bps e o eixo x indica as horas ao longo do dia.

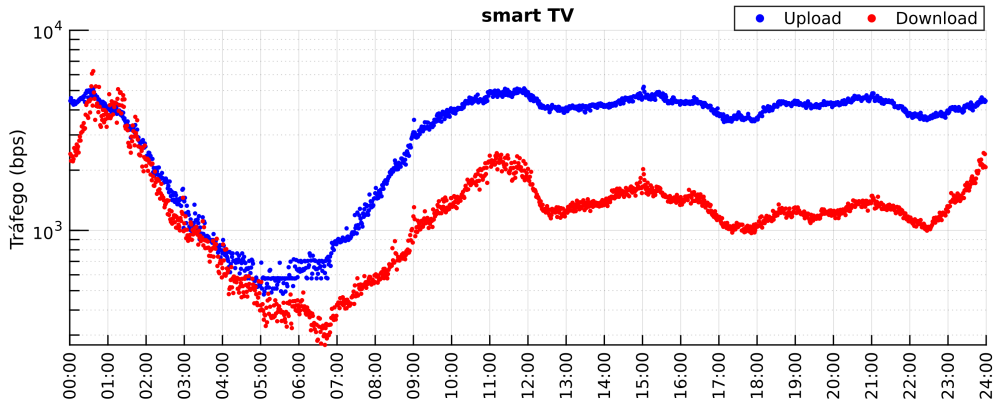

Figura A.5: Mediana do tráfego de *upload* e *download* por minuto para os dispositivos categorizados como *smart* TV, no período de 01/02 a 27/06/2020. O eixo y está em escala logarítmica e representa a taxa em bps e o eixo x indica as horas ao longo do dia.

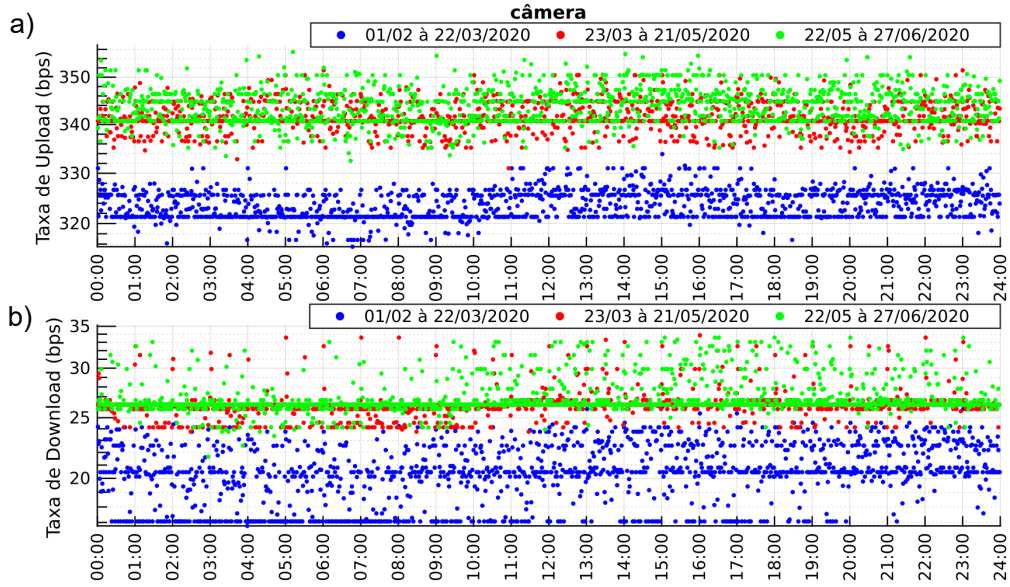

Figura A.6: Mediana do tráfego por minuto para os dispositivos categorizados como câmera considerando três períodos: 01/02 à 22/03/2020, 23/03 à 21/05/2020 e 22/05 à 27/06/2020. O eixo y está em escala logarítmica e representa a taxa em bps e o eixo x indica as horas ao longo do dia. a) Mediana do tráfego de *upload*. b) Mediana do tráfego de *download*.

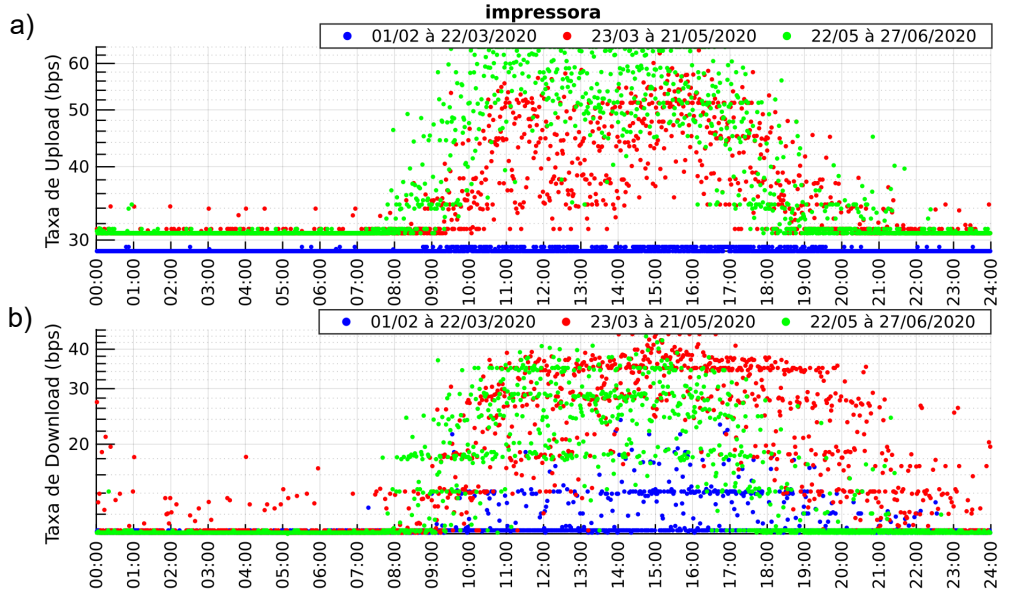

Figura A.7: Mediana do tráfego por minuto para os dispositivos categorizados como impressora considerando três períodos: 01/02 à 22/03/2020, 23/03 à 21/05/2020 e 22/05 à 27/06/2020. O eixo y está em escala logarítmica e representa a taxa em bps e o eixo x indica as horas ao longo do dia. a) Mediana do tráfego de *upload*. b) Mediana do tráfego de *download*.

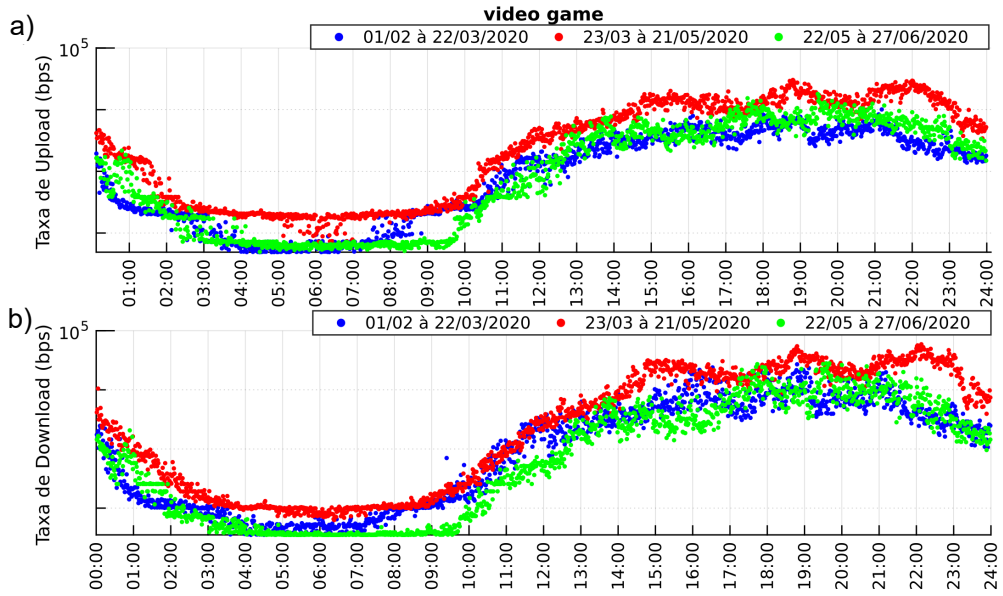

Figura A.8: Mediana do tráfego por minuto para os dispositivos categorizados como *video game* considerando três períodos: 01/02 à 22/03/2020, 23/03 à 21/05/2020 e 22/05 à 27/06/2020. O eixo y está em escala logarítmica e representa a taxa em bps e o eixo x indica as horas ao longo do dia. a) Mediana do tráfego de *upload*. b) Mediana do tráfego de *download*.

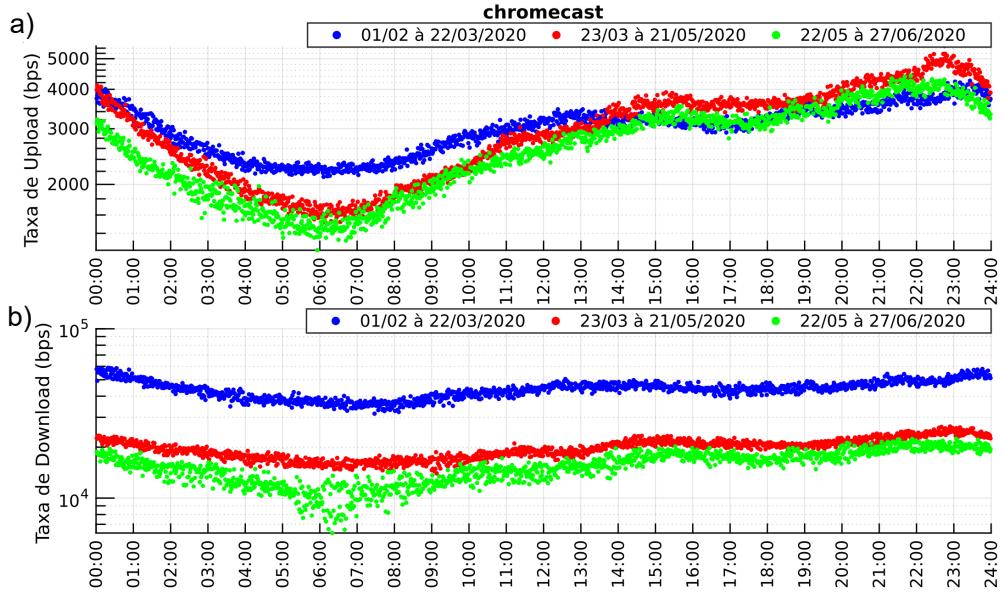

Figura A.9: Mediana do tráfego por minuto para os dispositivos categorizados como *chromecast* considerando três períodos: 01/02 à 22/03/2020, 23/03 à 21/05/2020 e 22/05 à 27/06/2020. O eixo y está em escala logarítmica e representa a taxa em bps e o eixo x indica as horas ao longo do dia. a) Mediana do tráfego de *upload*. b) Mediana do tráfego de *download*.

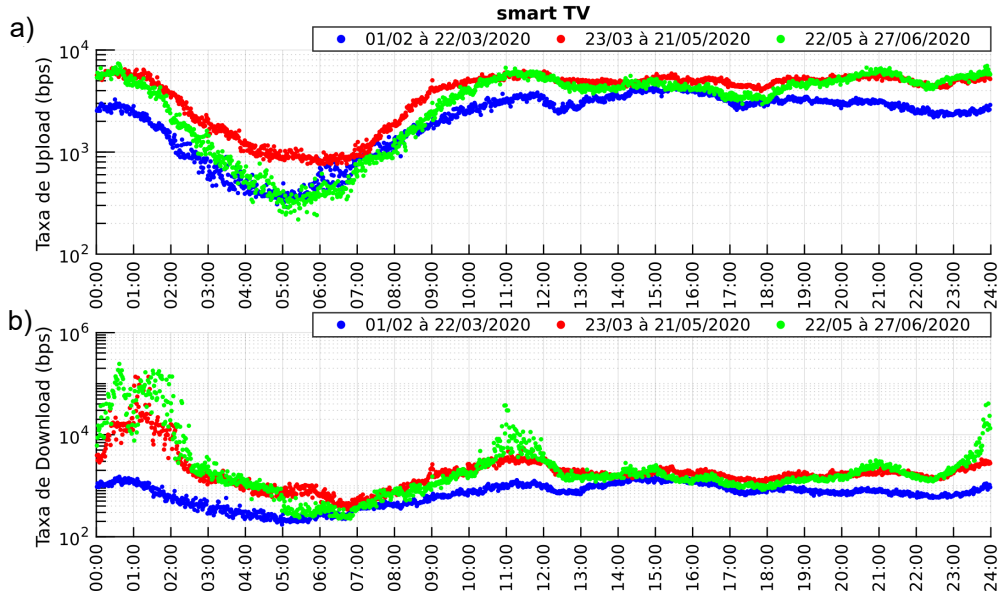

Figura A.10: Mediana do tráfego por minuto para os dispositivos categorizados como *smart* TV considerando três períodos: 01/02 à 22/03/2020, 23/03 à 21/05/2020 e 22/05 à 27/06/2020. O eixo y está em escala logarítmica e representa a taxa em bps e o eixo x indica as horas ao longo do dia. a) Mediana do tráfego de *upload*. b) Mediana do tráfego de *download*.

Apêndice B

Features selecionadas pelo algoritmo *Sequential Forward Selection*

Tabela B.1: Resultado da seleção das 10 *features* mais relevantes para os dispositivos categorizados como câmera, utilizando o algoritmo SFS com as métricas de avaliação acurácia e *F1-score*. Para cada métrica de avaliação o procedimento foi repetido três vezes variando o algoritmo de classificação (*Random Forest* (RF), Árvore de Decisão (DT) e K-vizinhos mais próximos (KNN)).

	Rank	***Features (RF)***	***Features (DT)***	***Features (KNN)***
Acurácia	1	up rate pps (max)	up rate pps (max)	up rate pps (max)
	2	up rate bps (max - min)	up/down ratio pps (max)	up rate bps (max)
	3	up rate pps (mean)	up rate pps (mean)	up rate pps (median)
	4	up rate bps (std)	up rate bps (max - min)	up/down ratio bps (median)
	5	up/down ratio pps (std)	up/down ratio bps (mean)	up rate bps (max-min)
	6	up rate bps (median)	up rate bps (std)	up rate pps (mean)
	7	up/down ratio bps (max)	up/down ratio bps (max)	up rate bps (mean)
	8	up rate pps (max - min)	up/down ratio bps (median)	up rate bps (std)
	9	up rate bps (max)	up/down ratio bps (std)	up/down ratio pps (median)
	10	up/down ratio pps (max)	up/down ratio bps (max - min)	up rate pps (std)
F1 score	1	up rate pps (max)	up rate pps (max)	up rate pps (max)
	2	up rate bps (max - min)	up/down ratio pps (max)	up rate bps (max)
	3	up rate pps (mean)	up rate pps (mean)	up rate pps (median)
	4	up/down ratio bps (max)	up rate bps (max - min)	up/down ratio bps (median)
	5	up rate bps (std)	up/down ratio bps (max - min)	up rate bps (max - min)
	6	up rate bps (median)	up/down ratio bps (median)	up rate pps (mean)
	7	up/down ratio pps (max)	up/down ratio bps (mean)	up rate bps (mean)
	8	up rate bps (mean)	up/down ratio pps (max - min)	up rate bps (std)
	9	up rate bps (max)	up/down ratio pps (std)	up/down ratio pps (median)
	10	up/down ratio bps (max -min)	up rate bps (std)	up rate pps (std)

Tabela B.2: Resultado da seleção das 10 *features* mais relevantes para os dispositivos categorizados como impressora, utilizando o algoritmo SFS com as métricas de avaliação acurácia e *F1-score*. Para cada métrica de avaliação o procedimento foi repetido três vezes variando o algoritmo de classificação (*Random Forest* (RF), Árvore de Decisão (DT) e K-vizinhos mais próximos (KNN)).

	Rank	***Features (RF)***	***Features (DT)***	***Features (KNN)***
Acurácia	1	up rate pps (max)	up rate pps (max)	up rate pps (max)
	2	up rate bps (mean)	up rate bps (std)	up rate bps (mean)
	3	up rate bps (median)	up rate bps (median)	up/down ratio bps (median)
	4	up rate bps (max)	up/down ratio pps (mean)	up rate bps (std)
	5	up/down ratio pps (median)	up/down ratio pps (median)	up rate pps (median)
	6	up rate pps (std)	up rate pps (std)	up rate bps (max)
	7	up/down ratio bps (median)	up rate pps (mean)	up rate bps (max -min)
	8	up rate pps (mean)	up/down ratio pps (std)	up rate bps (median)
	9	up/down ratio bps (mean)	up rate bps (mean)	up/down ratio bps (mean)
	10	up/down ratio bps (max)	up rate bps (max)	up/down ratio pps (median)
F1 score	1	up rate pps (max)	up rate pps (max)	up rate pps (max)
	2	up rate bps (mean)	up rate bps (std)	up rate bps (mean)
	3	up rate bps (median)	up rate bps (median)	up/down ratio pps (median)
	4	up rate pps (mean)	up rate pps (std)	up rate bps (std)
	5	up/down ratio bps (max)	up/down ratio bps (std)	up rate pps (median)
	6	up/down ratio pps (median)	up rate pps (mean)	up/down ratio bps (median)
	7	up rate bps (max - min)	up/down ratio bps (median)	up/down ratio bps (mean)
	8	up rate pps (std)	up/down ratio bps (max - min)	up/down ratio bps (max)
	9	up/down ratio bps (mean)	up rate pps (median)	up/down ratio bps (std)
	10	up/down ratio pps (max - min)	up/down ratio bps (max)	up/down ratio pps (mean)

Tabela B.3: Resultado da seleção das 10 *features* mais relevantes para os dispositivos categorizados como *video game*, utilizando o algoritmo SFS com as métricas de avaliação acurácia e *F1-score*. Para cada métrica de avaliação o procedimento foi repetido três vezes variando o algoritmo de classificação (*Random Forest* (RF), Árvore de Decisão (DT) e K-vizinhos mais próximos (KNN)).

	Rank	***Features (RF)***	***Features (DT)***	***Features (KNN)***
Acurácia	1	up rate pps (std)	up rate pps (std)	up rate pps (std)
	2	up rate bps (max - min)	up rate bps (std)	up rate bps (std)
	3	up rate bps (mean)	up rate pps (median)	up/down ratio bps (max)
	4	up rate bps (std)	up/down ratio pps (max - min)	up rate bps (mean)
	5	up/down ratio bps (max - min)	up rate bps (mean)	up/down ratio bps (max - min)
	6	up/down ratio pps (std)	up/down ratio bps (max - min)	up/down ratio pps (median)
	7	up rate pps (max - min)	up/down ratio bps (mean)	up/down ratio bps (median)
	8	up rate bps (max)	up/down ratio pps (max)	up/down ratio bps (mean)
	9	up rate bps (median)	up/down ratio bps (std)	up/down ratio bps (std)
	10	up rate pps (max)	up/down ratio bps (max)	up rate pps (max)
F1 score	1	up rate pps (std)	up rate pps (std)	up rate pps (std)
	2	up rate bps (max - min)	up rate bps (std)	up rate bps (std)
	3	up rate bps (mean)	up rate bps (mean)	up/down ratio bps (max)
	4	up rate bps (std)	up/down ratio bps (std)	up rate bps (mean)
	5	up/down ratio bps (mean)	up/down ratio pps (max)	up/down ratio bps (max -min)
	6	up rate pps (max)	up/down ratio pps (median)	up/down ratio pps (median)
	7	up/down ratio pps (mean)	up/down ratio bps (max - min)	up/down ratio bps (median)
	8	up rate bps (max)	up/down ratio bps (max)	up/down ratio bps (mean)
	9	up rate pps (max - min)	up rate pps (median)	up/down ratio bps (std)
	10	up rate bps (median)	up/down ratio pps (std)	up rate pps (max)

Tabela B.4: Resultado da seleção das 10 *features* mais relevantes para os dispositivos categorizados como *chromecast*, utilizando o algoritmo SFS com as métricas de avaliação acurácia e *F1-score*. Para cada métrica de avaliação o procedimento foi repetido três vezes variando o algoritmo de classificação (*Random Forest* (RF), Árvore de Decisão (DT) e K-vizinhos mais próximos (KNN)).

	Rank	***Features (RF)***	***Features (DT)***	***Features (KNN)***
Acurácia	1	up rate pps (std)	up rate pps (std)	up rate pps (std)
	2	up rate bps (std)	up/down ratio bps (std)	up rate bps (std)
	3	up rate bps (mean)	up/down ratio bps (median)	up rate bps (mean)
	4	up/down ratio bps (std)	up/down ratio bps (max - min)	up/down ratio bps (max)
	5	up/down ratio bps (median)	up/down ratio bps (max)	up/down ratio bps (max - min)
	6	up rate bps (max - min)	up/down ratio bps (mean)	up/down ratio bps (mean)
	7	up rate pps (max - min)	up rate bps (std)	up/down ratio bps (median)
	8	up rate pps (mean)	up rate bps (max - min)	up/down ratio bps (std)
	9	up/down ratio pps (std)	up rate bps (max)	up/down pps (max)
	10	up/down ratio bps (mean)	up rate bps (mean)	up/down ratio pps (std)
F1 score	1	up rate pps (std)	up rate pps (std)	up rate pps (std)
	2	up rate bps (std)	up/down ratio bps (std)	up rate bps (std)
	3	up rate bps (mean)	up/down ratio bps (median)	up rate bps (mean)
	4	up rate bps (max)	up rate bps (std)	up/down ratio bps (max)
	5	up/down ratio bps (max)	up/down ratio bps (max - min)	up/down ratio bps (max - min)
	6	up rate bps (max - min)	up/down ratio bps (max)	up/down ratio bps (mean)
	7	up rate pps (max - min)	up/down ratio bps (mean)	up/down ratio bps (median)
	8	up/down ratio pps (max - min)	up rate bps (max - min)	up/down ratio bps (std)
	9	up/down ratio bps (mean)	up rate bps (max)	up/down ratio pps (max)
	10	up rate bps (median)	up rate bps (mean)	up/down ratio pps (std)

Tabela B.5: Resultado da seleção das 10 *features* mais relevantes para os dispositivos categorizados como *smart* TV, utilizando o algoritmo SFS com as métricas de avaliação acurácia e *F1-score*. Para cada métrica de avaliação o procedimento foi repetido duas vezes variando o algoritmo de classificação (*Random Forest* (RF) e Árvore de Decisão (DT).

	Rank	***Features (RF)***	***Features (DT)***
Acurácia	1	up rate pps (std)	up rate pps (std)
	2	up rate bps (std)	up rate bps (std)
	3	up rate pps (max - min)	up/down ratio pps (mean)
	4	up/down ratio pps (mean)	up/down ratio bps (max)
	5	up rate bps (mean)	up/down ratio bps (max - min)
	6	up/down ratio bps (std)	up/down ratio bps (std)
	7	up rate bps (max - min)	up/down ratio bps (median)
	8	up/down ratio pps max - min)	up/down ratio pps (median)
	9	up/down ratio bps (max - min)	up/down ratio bps (mean)
	10	up rate pps (mean)	up/down ratio pps (std)
F1 score	1	up rate pps (std)	up rate pps (std)
	2	up rate bps (std)	up rate bps (std)
	3	up rate pps (max - min)	up/down ratio pps (mean)
	4	up/down ratio pps (mean)	up/down ratio bps (max - min)
	5	up rate bps (mean)	up/down ratio bps (median)
	6	up rate pps (mean)	up/down ratio bps (max)
	7	up rate bps (max - min)	up/down ratio pps (median)
	8	up rate pps (max)	up/down ratio bps (std)
	9	up down ratio bps (mean)	up/down ratio bps (mean)
	10	up down ratio bps (median)	up rate bps (mean)