Um Método para Detecção de Vulnerabilidades Através da Análise do Tráfego de Rede IoT

Uelinton Q. Brezolin [1], Nelson G. Prates Jr. [2], Andressa Vergütz [1], Michele Nogueira[2,1]

[1]Departamento de Informática
Universidade Federal do Paraná (UFPR)

[2]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)

{uqbrezolin,avergutz}@inf.ufpr.br

{nelsonprates,michele}@dcc.ufmg.br

Resumo. *A Internet das Coisas (do inglês, Internet of Things - IoT) compreende dispositivos sem fio com recursos computacionais limitados. Ela é alvo de ataques que exploram vulnerabilidades como a transferência de dados sem criptografia. A detecção convencional de vulnerabilidades ocorre a partir de bases de dados que listam as vulnerabilidades mais comuns (do inglês, common vulnerabilities and exposures – CVEs). Porém, essas bases são limitadas a vulnerabilidades conhecidas, o que na maioria das vezes não é o caso para o contexto da IoT. Este trabalho propõe MANDRAKE, um método para detecção de Vulnerabilidades através da análise do tráfego de Rede IoT e técnicas de aprendizado de máquina. A avaliação de desempenho foi conduzida em um cenário de casa inteligente com duas bases de dados, uma gerada experimentalmente e outra disponibilizada na literatura. Os resultados apontaram até 99% de precisão na detecção de vulnerabilidades na criptografia do tráfego.*

1. Introdução

Com o avanço tecnológico das redes sem fio e da Internet das Coisas (do inglês, *Internet of Things* - IoT), um número cada vez maior de dispositivos é conectado à Internet [Xie et al. 2017]. Existe uma ampla variedade de dispositivos IoT, como *smartwatch*, monitor de pressão do sangue, lâmpada inteligente e termostato, que são interconectados para coletar e monitorar dados rotineiros dos usuários e ambientes, melhorando a qualidade de vida. Tais dispositivos permitem diferentes aplicações e cenários, como hospitais, cidades e casas inteligentes. Por exemplo, a IoT oferece serviços de atendimento de urgência que monitoram a saúde das pessoas diariamente e notificam os transeuntes próximos sobre emergências. A melhoria da qualidade de vida facilitada pela IoT tem aumentado em grande proporção a diversidade e quantidade de dispositivos.

Os dispositivos IoT têm chamado atenção de atacantes. Sua capacidade computacional limitada deixa pouco espaço para segurança integrada, aumentando as ameaças contra a IoT. As restrições computacionais, a heterogeneidade e a baixa capacidade energética dos dispositivos IoT resultam em uma série de vulnerabilidades de segurança de complexa identificação [Bedhief et al. 2016]. A Forbes aponta as vulnerabilidades da IoT como uma das cinco principais ameaças de segurança atual [Forbes 2021]. Tais vulnerabilidades envolvem ausência de criptografia na troca de mensagens, uso de portas

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

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

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

A avaliação do método MANDRAKE seguiu duas abordagens: *i)* através de um cenário realista experimental e *ii)* orientado a traços da IoT. Na primeira abordagem, o cenário experimental contém dispositivos IoT, conectados em um ponto de acesso *WiFi*, gerando tráfego conforme as suas respectivas aplicações e fabricantes (exemplo, *stream*

de vídeo). Na segunda abordagem, a avaliação do método utiliza o conjunto de dados *IoT Traffic Traces* que contém o tráfego de uma casa inteligente [Sivanathan et al. 2019]. Com base no conjunto de dados da rede do ambiente experimental e do *dataset*, extraem-se as características de rede, criam-se as tuplas de fluxo, calcula-se a entropia para rotulagem dos dados e classifica-se os fluxos existentes vulneráveis. A classificação dos fluxos vulneráveis seguiu dois cenários: duas classes e multi-classes. No cenário de duas classes, classificam-se os fluxos como vulneráveis e não vulneráveis. No cenário multi-classes, o método aponta se o dispositivos IoT possui fluxo vulnerável. Além disso, com o propósito de tornar mais realista a avaliação, classificam-se os fluxos vulneráveis seguindo tanto uma abordagem *offline*, quanto em *stream* de treinamento dos dados. Por fim, para obter precisão na detecção de tráfego vulnerável, avaliaram-se empiricamente todos os fluxos do tráfego para determinação do limiar de vulnerabilidade utilizado na análise da entropia. Os resultados apontaram até 99% de precisão na detecção de vulnerabilidades na IoT em ambas as abordagens de avaliação.

Este artigo procede como segue. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 detalha o método proposto. A Seção 4 descreve a metodologia de avaliação, apresenta e discute os resultados obtidos. A Seção 5 conclui o trabalho.

2. Trabalhos Relacionados

Na literatura, diferentes trabalhos detectam vulnerabilidades de segurança em dispositivos conectados à Internet. Uma detecção convencional de vulnerabilidades ocorre a partir de bases de dados que listam as vulnerabilidades mais comuns (do inglês, *common vulnerabilities and explosures* – CVEs). Porém, essas bases são limitadas a vulnerabilidades conhecidas (como versões desatualizadas de *software*), o que na maioria das vezes não é o caso para o contexto da IoT. Outras mecanismos tomam como base a análise do tráfego da rede para detectar vulnerabilidades, o que diminui os custos operacionais e simplifica o processo de detecção, pois não requer acesso ao dispositivo físico [Jia et al. 2017, Yi et al. 2019, Huang et al. 2020]. No geral, tais mecanismos empregam técnicas que reconhecem padrões dos dispositivos e da rede por meio de algoritmos de aprendizado de máquina [Sonnekalb 2019, Chernis and Verma 2018, Huang et al. 2020], e técnicas que mapeiam a rede por meio de grafos [Jia et al. 2018, Fang et al. 2019]. Outros trabalhos seguem fontes de dados de níveis diferentes, como o de *firmware* e *software*. Entretanto, as soluções baseadas no nível de *firmware* analisam um conjunto de instruções operacionais que são programadas diretamente no *hardware* [Sachidananda et al. 2020, He et al. 2020], e no nível de *software* as propostas baseiam-se em uma sequência de instruções a serem seguidas e/ou executadas na manipulação, redirecionamento ou modificação do *hardware* do dispositivo [Lin et al. 2020, Zhang 2020].

No nível de *firmware*, [Sachidananda et al. 2020] apresentaram uma estrutura de análise estática para detectar vulnerabilidade do dispositivo IoT. [He et al. 2020] projetaram uma plataforma híbrida para detectar vulnerabilidades no *firmware* IoT, que integra detecção estática *offline* e detecção dinâmica *online*. [Medeiros et al. 2016] propuseram uma ferramenta de análise estática que aprende a detectar vulnerabilidades em aplicações *web*. A ferramenta usa um modelo de sequência para aprender a caracterizar vulnerabilidades com base num conjunto de "fatias" de código-fonte. Entretanto, estes trabalhos não consideram o tráfego de rede na detecção das vulnerabilidades, necessitando assim de acesso aos componentes de *software* e *hardware* dos dispositivos.

Dentre os trabalhos que detectam vulnerabilidades baseados na análise de tráfego com o auxílio de grafos ou aprendizado de máquina, [Jia et al. 2018] apresentaram um mecanismo baseado em grafos para identificar vulnerabilidades na comunicação de dispositivos IoT. Todavia, o mecanismo obtém informações da carga útil (*payload*) do pacote, o que compromete a privacidade dos dados. [Fang et al. 2019] demonstraram uma estrutura formal de camada cruzada para revelar de forma abrangente as vulnerabilidades na IoT. Essa estrutura gera um grafo que representa todos os componentes essenciais da IoT, desde ambientes físicos de baixo nível até processos de tomada de decisão de alto nível, permitindo capturar os caminhos correspondentes a um potencial ataque. Todavia, a estrutura possui um custo de memória excessivo por mapear o caminho do ataque. [Yi et al. 2019] apresentaram um método fundado na construção de grafos de ataque que estabelece um modelo de avaliação de segurança de rede para analisar as vulnerabilidades na IoT. [Jia et al. 2017] apresentaram um sistema de detecção semiautomática de vulnerabilidades no cenário de casas inteligentes antes dos dispositivos IoT usados serem vendidos. Entretanto, o sistema se comporta de forma invasiva, usando arquivos de configurações dos aplicativos dos dispositivos para detecção. [Bhatia et al. 2019] apresentaram uma abordagem baseada em aprendizado de máquina não supervisionado para detecção de anomalias centradas em redes IoT. Os autores mostraram que classificadores não supervisionados baseados em *Autoencoder*, quando treinados em dados de tráfego benigno, são eficazes na modelagem do comportamento da rede e na detecção de anomalias. Porém, a abordagem foca somente em tráfego utilizando o *Transmission Control Protocol* (TCP) e a detecção ocorre após a ocorrência do ataque. Portanto, os trabalhos são ou restritos a protocolos.

Tendo em vista a importância da criptografia, alguns trabalhos classificam o tráfego vulnerável sem criptografia. Por exemplo, [Wang et al. 2011] projetaram um classificador para distinguir o tráfego da Internet em diferentes tipos de conteúdo usando técnicas de aprendizado de máquina. [Rezaei and Liu 2019] apresentaram uma estrutura geral para classificação de tráfego baseada em aprendizado profundo (*deep leraning*). Porém, estas técnicas não distinguem tráfego criptografado e não criptografado. [Puhan et al. 2014] apresentaram uma abordagem de detecção de memória sem criptografia usando análise dinâmica de fluxo de dados. Entretanto, não se mostrou eficaz na detecção utilizando o *Transport Layer Security* (TLS). [Ma et al. 2020] treinaram um modelo baseado no algoritmo *K-nearest neighbor* (KNN) que necessita apenas de uma pequena quantidade de dados para classificação refinada de fluxo de rede criptografados. Contudo, não foi demonstrado como é detectado o tráfego criptografado.

Nesse sentido, existem trabalhos que detectam vulnerabilidades no nível de *firmware*, *software* e rede. Outros trabalhos exploram a detecção de vulnerabilidades, utilizando abordagens em grafos e aprendizado de máquina. Porém, na maioria dos casos, os trabalhos necessitam de acesso aos componentes de *software* e *firmware* dos dispositivos. Em contrapartida, os trabalhos que não necessitam de acesso ao dispositivo IoT possuem abordagens com alto custo de memória para a execução e detectam a vulnerabilidade após o ataque ser efetivado. Já os trabalhos que fazem a classificação de tráfego vulnerável se mostram funcionais. Porém, eles possuem restrições em protocolos ou não diferenciam tráfego vulnerável. Dessa forma, um método de detecção de vulnerabilidade eficaz, sem a necessidade de acesso aos componentes dos dispositivos, que identifica dispositivos que geram tráfego vulnerável e com o custo de processamento inferior se torna indispensável

para a segurança de redes IoT, pois incrementa as técnicas de detecção da literatura.

3. O Método MANDRAKE

Este trabalho apresenta o método MANDRAKE, do inglês *a **M**ethod for vulner**A**bilities detectio**N** base**D** on the IoT netwo**R**k p**A**c**KE**t traffic*. Seu objetivo consiste em detectar vulnerabilidades a partir da análise do tráfego de rede IoT e técnicas de aprendizado de máquina. Particularmente, o método busca identificar tráfego de rede não criptografado e complementa as estratégias e soluções existentes na literatura. O método rotula automaticamente parte da base de dados e utiliza o mínimo possível de instâncias rotuladas a fim de construir um classificador capaz de identificar se o tráfego possui fluxos vulneráveis e sua origem. Para isso, o método segue três fases de execução mostradas na Figura 1: *(1)* pré-processamento dos dados, *(2)* cálculo da entropia e *(3)* classificação com base em algoritmos de aprendizagem de máquina. Para atingir sucesso no funcionamento do método MANDRAKE, no contexto da IoT de uma casa inteligente, sugere-se o posicionamento do mesmo em um dispositivo que possua acesso a todo o tráfego transmitido na rede, como o *gateway*. As próximas subseções detalham as fases do método.

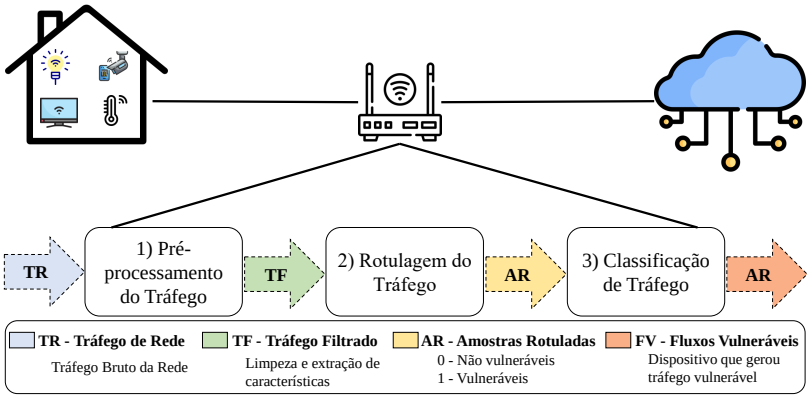


Figura 1. Fases de Execução do Método MANDRAKE

3.1. Pré-processamento do Tráfego

Esta fase compreende três principais ações: *(i)* coleta e filtragem do tráfego da rede, *(ii)* organização das amostras de tráfego em fluxos e *(iii)* extração das características dos pacotes dos fluxos de rede. Na ação *(i)*, o método proposto coleta e filtra o tráfego da rede sem fio por meio de um *sniffer*, em modo promíscuo, em uma placa de rede ou do roteador de borda (*sink node*, *gateway* ou *access point* - AP) da própria infraestrutura almejada. Portanto, o *sniffer* utiliza uma interface de rede sem fio (por exemplo, *Wi-Fi* ou IEEE 802.15.4), em modo promíscuo, e programada para coletar pacotes que trafegam de acordo com a pilha de protocolos *Transmission Control Protocol/Internet Protocol* (TCP/IP). Durante a captura, o *sniffer* filtra os pacotes e seleciona apenas os pacotes provenientes de comunicações através da Internet e com as características que permitem as análises. Dessa forma, o filtro descarta os pacotes correspondentes a *broadcast* da rede local e pacotes com a carga útil (*payload*) vazia.

A ação *(ii)* toma como base o tráfego da rede capturado e separa as amostras do tráfego bruto em fluxos, tanto TCP quanto UDP, conforme a tupla: IP de origem, IP de

destino, porta de origem, porta de destino e protocolo da camada de transporte. Cada amostra de fluxo representa um conjunto de pacotes provenientes de uma comunicação de um dispositivo IoT com um serviço ou aplicação através da Internet.

A partir dos fluxos criados, na ação *(iii)* consiste em calcular uma série de características estatísticas sobre os conjuntos de amostras de pacotes pertencentes ao mesmo fluxo e transforma em uma única amostra de estatísticas do fluxo. As características estatísticas seguem a média (μ), a soma (sum) e o desvio padrão (DP) das características de rede como duração do fluxo, instante de tempo *timestamp* de início e fim, e tamanho dos pacotes. No final, cada dispositivo tem suas próprias amostras de fluxos com suas propriedades estatísticas.

3.2. Rotulagem de Tráfego

A fase 2 do método é responsável por rotular os fluxos de tráfego em vulnerável (1) e não vulnerável (0). Para isso, esta fase toma como entrada os fluxos do tráfego filtrados na fase anterior, porém desconsidera as medidas estatísticas na rotulagem. A rotulagem do tráfego vulnerável utiliza apenas os dados oriundos dos fluxos (seguindo a mesma tupla) e a carga útil (*payload)* de cada pacote pertencente aos fluxos. Sendo assim, seleciona-se um conjunto de pacotes pertencentes a cada fluxo filtrado. Para cada pacote do conjunto selecionado, é calculada uma estimativa de aleatoriedade. Esta estimativa de aleatoriedade é dada pela fórmula da **entropia** que está relacionada ao grau de desordem de uma variável de entrada [Shannon 1948]. A fórmula da entropia calcula a aleatoriedade do pacote de rede com base na sua carga útil atribuindo uma nota entre 0 e 1. Essa nota representa a probabilidade do pacote estar criptografado ou não. Quanto mais próximo do valor 1, mais aleatoriedade é encontrada, ou seja, aumenta a possibilidade do pacote de rede estar utilizando técnicas de criptografia. Em contrapartida, uma baixa aleatoriedade na carga útil indica uma provável ausência de criptografia.

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

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

Quanto mais próximo de 1 a probabilidade do tráfego estar criptografado aumenta. Porém, de acordo com [Dorfinger et al. 2011], os valores acima de $0,7$ estimam corretamente a probabilidade do pacote estar criptografado. O método MANDRAKE contabiliza os pacotes que possuem notas superiores e inferiores à $0,7$, onde uma condição no conjunto de valores estimado pela entropia separa os pacotes com notas superiores e inferiores a $0,7$. Neste condicionamento, o método calcula a taxa de pacotes criptografados em

cada fluxo (exemplo, analisa se entre 70% e 90% dos pacotes do mesmo fluxo possuem criptografia) e assim, estima-se que o tráfego do fluxo não está criptografado, isto é, o tráfego é vulnerável. Por fim, esta fase rotula as amostras com base no endereço MAC de cada dispositivo e na definição de fluxos vulneráveis ("***1***" para as amostras consideradas vulneráveis ou "***0***" para as amostras consideradas não-vulneráveis).

3.3. Classificação do Tráfego Vulnerável

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

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

4. Avaliação de Desempenho

A avaliação de desempenho do método MANDRAKE seguiu duas abordagens: *i)* através de um cenário realista experimental (Exp) e *ii)* orientado a traços da IoT (IoT). Na primeira abordagem, o cenário experimental contém dispositivos IoT, conectados em um ponto de acesso *WiFi*, gerando tráfego conforme as suas respectivas aplicações e fabricantes (exemplo, *stream* de vídeo). Na segunda abordagem, a avaliação do método utiliza o conjunto de dados *IoT Traffic Traces* que contém o tráfego de uma casa inteligente [Sivanathan et al. 2019]. Em ambas as abordagens, a avaliação utiliza como entrada o tráfego da rede IoT de uma casa inteligente.

Com base no conjunto de dados da rede do ambiente experimental e do *dataset*, extraem-se as características de rede, criam-se as tuplas de fluxo, calcula-se a entropia para rotulagem dos dados e classificam-se os fluxos. A motivação para criação de um ambiente experimental se deve à necessidade de avaliar o método MANDRAKE em um ambiente realista, neste caso, uma casa inteligente. A limitação no número de dispositivos IoT no ambiente experimental motivou o uso de um conjunto de dados também de uma

casa inteligente, permitindo a análise em uma maior diversidade de dispositivos. Além disso, o uso de um conjunto de dados facilita a reprodutibilidade do trabalho.

Em ambas as abordagens de avaliação, considera-se a execução do método MANDRAKE no ponto de acesso da casa inteligente. Uma vez que nas redes sem fio, os dispositivos IoT transmitem dados pessoais em forma de pacotes padronizados por protocolos populares, não é possível saber quem está recebendo ou "farejando" (*sniffing*/ouvindo sem autorização) o tráfego da rede. Neste contexto, um dispositivo vulnerável não emprega corretamente as técnicas de criptografia e pode servir como entrada para usuários mal-intencionados realizar vetores de ataques, controlar os dispositivos remotamente, roubar os dados pessoais e/ou causar prejuízos financeiros.

Abordagem de Avaliação 1: Detalhes do Cenário Experimental

No cenário experimental, os dispositivos estabelecem uma conexão com os serviços na Internet por meio de uma rede local *Wi-Fi*. Tais dispositivos possuem as configurações de fábrica e operam conforme suas respectivas aplicações. A Figura 2 é uma foto tirada do ambiente experimental com os dispositivos considerados nas análises. As câmeras de monitoramento transmitiram as gravações constantemente para o servidor na nuvem de cada fabricante. O *smartphone* acessa as gravações das câmeras por meio das aplicações em nuvem. A *smartTV* conecta-se com o serviço de *stream* do *Yotube* e o *Playstation* 4 conectado a um jogo *online*. Enquanto os dispositivos geram tráfego, um laptop contendo o sistema operacional *Ubuntu Desktop* na versão 20.04 LTS observa a rede sem fio por meio de uma placa de rede e da ferramenta *Wireshark* [Orebaugh et al. 2007]. O tráfego de rede foi coletado por 2 horas, gerando um total de 638166 pacotes e 2,9 GB de dados.

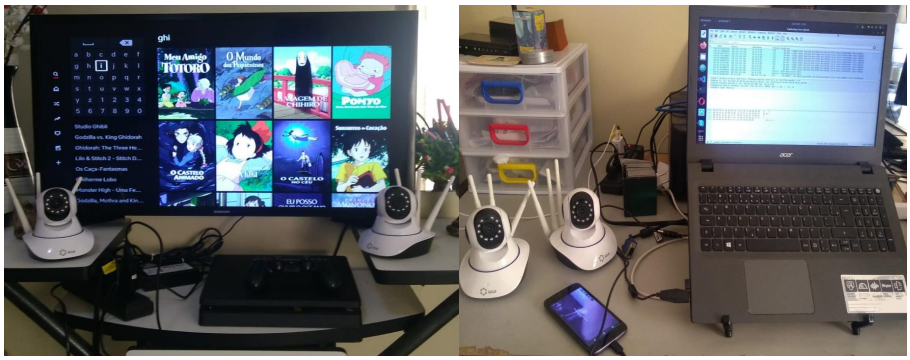


Figura 2. Cenário Experimental

Abordagem de Avaliação 2: Características do Conjunto de Dados

O conjunto de dados *IoT Traffic Traces* [Sivanathan et al. 2019] possui a coleta do tráfego da rede de uma casa inteligente composta por 28 dispositivos IoT e não-IoT. A coleta do tráfego da rede ocorreu do dia 22 de setembro de 2016 à 12 de outubro de 2016, totalizando 20 dias de captura de tráfego de rede, 29.3 GB de dados e 1629879 pacotes. A captura foi dividida em 20 arquivos *pcaps* referentes a cada dia de coleta. O *dataset* contém o registro do tráfego de rede de diferentes dispositivos IoT e não-IoT, como assistentes virtuais, lâmpada inteligente, sensor de movimento, monitor de pressão arterial, monitor de bebê, balança Wi-Fi, notebooks, sensor de movimento, smartphones, porta-retrato digital,

caixa de som portátil, interruptor de luz inteligente, câmeras de monitoramento Wi-Fi, *gateway*, impressoras, entre outros. Por ser um conjunto de dados composto por diferentes dispositivos, as características do tráfego da rede apresentam variabilidade de um dispositivo para o outro, como diferentes tamanhos de pacotes. Desse modo, o *dataset* possui uma ampla diversidade com dados sensíveis coletados pelos dispositivos, viabilizando a avaliação da detecção de vulnerabilidades do método proposto.

Detalhamento da Rotulagem e Classificação do Tráfego Vulnerável

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

A entropia foi avaliada empiricamente, sendo implementada para medir a aleatoriedade do *payload* dos pacotes da rede. Dessa forma, criaram-se dois ambientes de testes isolados do tráfego normal. Sendo uma aplicação local, enviando e recebendo pacotes e a outra com duas máquinas virtuais em rede trocando informações. Cada pacote trafegado foi analisado individualmente, visualizando seu conteúdo (*payload*). A cada *payload* analisado antes da execução do cálculo da entropia, uma tentativa de decodificação de caracteres é usada, pois o conteúdo do pacote extraído corresponde num conjunto de caracteres (*string*). Assim, essa tentativa de decodificação valida a nota dada pela fórmula da entropia. Quando a tentativa de decodificação é bem sucedida, a nota da entropia automaticamente apresenta um valor inferior a $0,7$, conforme a teoria apresentada por [Dorfinger et al. 2011]. Caso contrário, quando a tentativa de decodificação do *payload* é mal sucedida, a nota da entropia corresponde a um valor superior a $0,7$. Portanto, a tentativa de decodificação demonstrou que o cálculo da entropia é eficiente na detecção de alto índice de aleatoriedade dos pacotes. Após calculado a entropia e rotulado o tráfego, armazena-se em um arquivo final a rotulagem feita pela entropia juntamente com o segundo arquivo que contém as amostras de fluxo com as respectivas medidas estatísticas. Por fim, submete-se aos classificadores este arquivo final, um de cada base Exp e IoT, para validação da entropia e detecção dos fluxos vulneráveis.

A classificação dos fluxos vulneráveis seguiu dois cenários: duas classes e multiclasses. Nas duas classes, classificam-se os fluxos como vulneráveis (1) e não vulneráveis

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

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

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

Algoritmos de Aprendizagem de Máquina e Métricas de Avaliação

Na avaliação, consideram-se os seguintes algoritmos supervisionados: *Adaptive Random Forest* (ADR), *Naive Bayes* (NB), *Multi-Layer Perceptron* (MLP) e *K-Nearest Neighbourn* (KNN), visto que são amplamente utilizados na literatura para classificação de tráfego de rede [Sonnekalb 2019, Sivanathan et al. 2019]. As métricas consideradas envolvem a acurácia, precisão, *recall* e *F1-Score* (também conhecida como *F-Measure*). Estas métricas consideram a taxa de verdadeiros positivos (VP), verdadeiros negativos (VN), falsos positivos (FP) e falsos negativos (FN). Assim, a acurácia se refere à proporção do tráfego de dados classificados corretamente em relação a todas as amostras de tráfego. A precisão (p) estima a porcentagem de verdadeiros positivos ($VP/(VP+FP)$) em todas as amostras de tráfego classificadas como positivas. O *recall* (r) ou revocação consiste na porcentagem de amostras verdadeiras positivas em todas as amostras cuja categoria esperada é positiva ($VP/(VP+FN)$). Finalmente, *F1-Score* estabelece uma relação entre a medida de precisão e *recall* estimando a média harmônica ($2rp/(r+p)$). Portanto, os resultados são baseados nessas métricas.

4.1. Resultados

A Tabela 1 apresenta os resultados relacionados à avaliação empírica do cálculo da entropia, seguindo a aplicação local e a máquina virtual. Em ambos os cenários, os resultados da detecção da vulnerabilidade de ausência de criptografia foram similares e satisfatórios. Precisamente, na aplicação local, a precisão da entropia manteve-se em torno de 90%, identificando corretamente os pacotes criptografados, e 100% de precisão para pacotes sem criptografia. No ambiente da máquina virtual, a precisão da entropia manteve-se em torno de 91%, identificando corretamente os pacotes criptografados e 100% de precisão para pacotes sem criptografia. Vale ressaltar que cada pacote trafegado foi analisado individualmente. Para conteúdo do pacote sem técnica de criptografia a nota da entropia era inferior a $0,7$, caso contrário, a nota era superior a $0,7$. Com base nos resultados

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

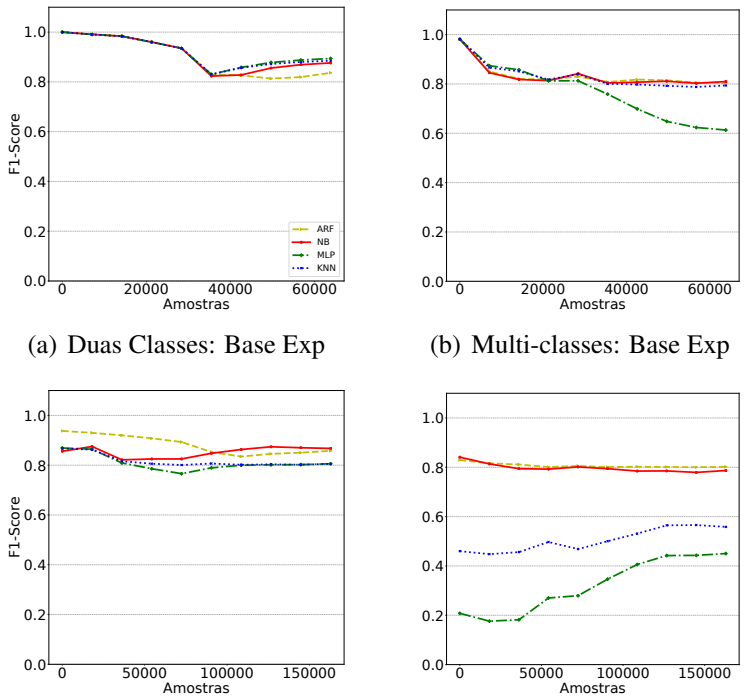
[5]Biblioteca Scikit Multiflow: https://scikit-multiflow.github.io/. Acessado em Fev/2022.

satisfatórios da entropia, pode-se seguir com a detecção de fluxos vulneráveis por meio dos classificadores.

Tabela 1. Avaliação Empírica do Cálculo da Entropia

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

A Figura 3 mostra o desempenho do método MANDRAKE na detecção dos fluxos vulneráveis seguindo os cenários de avaliação de duas classes e multi-classes no modo de aprendizagem *m-st*. Note que os classificadores foram treinados parcialmente com as 100 primeiras instâncias de cada classe e no decorrer da base eles foram treinados novamente toda vez que o limiar de 80% de F1-Store era atingido. As Figuras 3(a) e 3(c) apontam que os classificadores mantiveram o desempenho em ≈ 80% de F1-Score no cenário de classificação de tráfego vulnerável e não vulnerável, quando comparado aos resultados dos cenários multi-classes das Figuras 3(b) e 3(d).



(a) Duas Classes: Base Exp (b) Multi-classes: Base Exp

(c) Duas Classes: Base IoT (d) Multi-classes: Base IoT

Figura 3. Avaliação dos Classificadores em *Stream*

Esses resultados demonstram que a rotulagem da entropia, a qual serviu de entrada para os classificadores detectarem tráfego vulnerável e não vulnerável, obteve sucesso, visto que todos os classificadores obtiveram ≈ 80% de F1-Score. Em contrapartida, o cenário de detecção de dispositivos contendo tráfego vulnerável (multi-classes) apresentou valores próximos ou menores do que 60% de F1-Score, atingindo 20% no algoritmo MLP da Figura 3(d). Isso se deve ao fato da base IoT possuir uma maior quantidade de dispositivos do que a base Exp, o que impactou nos algoritmos KNN e MLP que tomam como base a similaridade dos dados. Por isso, mesmo utilizando todas as instâncias no

treinamento (150000), os algoritmos não conseguiram atingir o limiar de 80% de F1-Score. Além disso, nos cenários multi-classes, o algoritmo NB, que utiliza distribuições de probabilidade, manteve o F1-Score médio de 81,5% e, para manter esse resultado, precisou de 72% da base no treinamento.

A Tabela 2 mostra os resultados considerando todas as métricas, bem como os cenários de duas classes (Duas) e multi-classes (Mult), os modos de aprendizagem *m-off* e *m-st*, e as duas bases Exp e IoT. Os resultados corroboram que o cenário de duas classes (taxa de precisão entre 85% e 99%) apresenta um desempenho melhor do que o multi-classes (taxa de precisão entre 32% e 98%). Em relação aos modos de aprendizagem, observa-se que no modo *m-off* os resultados atingiram valores médios de 91% de F1-Score, relativamente maior que no *m-st* com 79% de F1-Score médio. Estes resultados se justificam, pois o modo *m-off* não respeita a ordem do fluxo ao dividir as amostras e melhora o balanceamento dos dados. No entanto, note que mesmo com essa vantagem o algoritmo NB apresentou acurácia de $\approx$ 60% no modo *m-off*, sendo menos eficiente que os demais classificadores considerados nesta avaliação. Tanto no cenário de classificação com duas classes quanto no multi-classes, o algoritmo RF apresentou as maiores taxas médias de acerto na detecção de fluxos vulneráveis, com $\approx$ 91% de F1-Score, enquanto o algoritmo KNN atingiu a segunda melhor melhor taxa com $\approx$ 85% de F1-Score médio. Portanto, com base nestes resultados, o método MANDRAKE apresentou sucesso na detecção de fluxos vulneráveis no ambiente experimental e orientado a traços.

Tabela 2. Resultados de Detecção de Fluxos Vulneráveis do Método MANDRAKE

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

Em resumo, os resultados apontam que o método MANDRAKE torna possível a rotulação da base, identificação da presença de tráfego vulnerável e de seus respectivos dispositivos de origem, sem conhecimento prévio da rede. Cada uma das fases do método foi avaliada experimentalmente com bases realistas. A Fase *(1)* (pré-processamento do tráfego) tornou possível a captura e criação de amostras de fluxos. A Fase *(2)* (rotulagem do tráfego com base na entropia) não apresentou falsos negativos, ou seja, nenhuma

amostra vulnerável foi identificada como não-vulnerável. Este fato confirma a eficiência do cálculo da entropia. Por fim, a Fase *(3)* (classificação dos fluxos vulneráveis) apresentou altas taxas de acerto em diferentes cenários de avaliação e problemas de classificação, com acurácia entre 88,5% e 99% nos problema de duas classes e com acurácia entre 82,1% e 98% no problema multi-classes.

5. Conclusão

Este artigo apresentou o método MANDRAKE, que detecta vulnerabilidades na IoT a partir da análise do tráfego de rede e técnicas de aprendizagem de máquina. O método é simples, eficiente e não requer conhecimento prévio da rede. O método toma como base o cálculo da entropia para rotulação dos fluxos vulneráveis e algoritmos supervisionados para detecção destes fluxos. Sua avaliação seguiu duas abordagens no contexto de casas inteligentes, cenário experimental e orientado a traços, além de diferentes cenários de detecção de vulnerabilidade (modo *offline* e *stream*). Os resultados do método MANDRAKE obtiveram até 99% de precisão na identificação de dispositivos vulneráveis. Além disso, os resultados apontam que é possível rotular e treinar uma base de dados de acordo com a entropia, e assim criar modelos capazes de identificar dispositivos vulneráveis que não possuem técnicas de criptografia na sua comunicação. Como direções futuras, pretende-se avaliar o impacto do processamento desta solução na rede.

Referências

Bedhief, I., Kassar, M., and Aguili, T. (2016). SDN-based architecture challenging the IoT heterogeneity. In *Proc. of the IEEE SCNS*, pages 1–3, Dubai, United Arab Emirates.

Bhatia, R., Benno, S., Esteban, J., Lakshman, T. V., and Grogan, J. (2019). Unsupervised machine learning for network-centric anomaly detection in IoT. In *Proc.of the ACM*, Big-DAMA, page 42–48, New York, NY, USA. ACM.

Chernis, B. and Verma, R. (2018). Machine learning methods for software vulnerability detection. In *Proc. of the ACM, IWSPA*, page 31–39, Tempe, AZ, USA.

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

Fang, Z., Fu, H., Gu, T., Qian, Z., Jaeger, T., and Mohapatra, P. (2019). Foresee: A cross-layer vulnerability detection framework for the IoT. In *Proc. of the IEEE MASS*, pages 236–244, Monterey, CA, USA.

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

He, D., Gu, H., Li, T., Du, Y., Wang, X., Zhu, S., and Guizani, N. (2020). Toward hybrid static-dynamic detection of vulnerabilities in IoT firmware. *IEEE Network*, 2(35):202–207.

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

Jia, X., Li, X., and Gao, Y. (2017). A novel semi-automatic vulnerability detection system for smart home. In *Proc. of the BDIoT*, page 195–199, New York, NY, USA.

Jia, Y., Xiao, Y., Yu, J., Cheng, X., Liang, Z., and Wan, Z. (2018). A novel graph-based mechanism for identifying traffic vulnerabilities in smart home IoT. In *Proc. of the IEEE INFOCOM*, pages 1493–1501, Honolulu, HI, USA.

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

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

Medeiros, I., Neves, N., and Correia, M. (2016). Dekant: A static analysis tool that learns to detect web application vulnerabilities. In *Proc. of the ACM ISSTA*, page 1–11, New York, NY, USA.

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

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

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

Sachidananda, V., Bhairav, S., and Elovici, Y. (2020). Over: Overhauling vulnerability detection for IoT through an adaptable and automated static analysis framework. In *Proc. of the ACM SAC*, page 729–738, New York, NY, USA.

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

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

Sonnekalb, T. (2019). Machine-learning supported vulnerability detection in source code. In *Proc. of the ACM ESEC/FSE*, page 1180–1183, Tallinn, Estonia.

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

Xie, W., Jiang, Y., Tang, Y., Ding, N., and Gao, Y. (2017). Vulnerability detection in IoT firmware: A survey. In *Proc. of the IEEE ICPADS*, pages 769–772, Shenzhen, China.

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

Zhang, B. (2020). A software upgrade security analysis method on network traffic classification using deep learning. In *Proc. of the IEE ICUEMS*, pages 568–574, Zhuhai, China.