UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO SISTEMAS DE INFORMAÇÃO

Bruno Ribeiro da Silva

Abordagem de detecção de intrusão para ambientes IoT baseados em computação em névoa

Florianópolis
2019

Bruno Ribeiro da Silva

Abordagem de detecção de intrusão para ambientes IoT baseados em computação em névoa

Trabalho Conclusão do Curso de Graduação em Sistemas de Informação do Centro Tecnológico da Universidade Federal de Santa Catarina como requisito para a obtenção do Título de Bacharel em Sistemas de Informação.
Orientador: Prof. Dr. Carlos Becker Westphall
Coorientador: Me. Cristiano Antonio de Souza

Florianópolis
2019

Ficha de identificação da obra

Silva, Bruno Ribeiro da

Abordagem de detecção de intrusão para ambientes IoT baseados em computação em névoa / Bruno Ribeiro da Silva ; orientador, Carlos Becker Westphall, coorientador, Cristiano Antonio de Souza, 2019.

83 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Sistema de Informação, Florianópolis, 2019.

Inclui referências.

1. Sistema de Informação. 2. Internet of Things. 3. Fog Computing. 4. Intrusion Detection. I. Westphall, Carlos Becker. II. Souza, Cristiano Antonio de. III. Universidade Federal de Santa Catarina. Graduação em Sistema de Informação. IV. Título.

Bruno Ribeiro da Silva

Abordagem de detecção de intrusão para ambientes IoT baseados em computação em névoa

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de Bacharel e aprovado em sua forma final pelo Curso de Sistemas de Informação.

Florianópolis, 6 de dezembro de 2019.

_____

Prof. Dr. Cristian Koliver
Coordenador do Curso

Banca Examinadora:

_____

Prof. Dr. Carlos Becker Westphall
Orientador
Universidade Federal de Santa Catarina

_____

Me. Cristiano Antonio de Souza
Coorientador
Universidade Federal de Santa Catarina

_____

Prof.ª Dra. Carla Merkle Westphall
Universidade Federal de Santa Catarina

_____

Prof. Dr. Jorge Werner
Universidade Federal de Santa Catarina

Dedico esse trabalho aos meus pais, que sempre me apoiaram e à Janaina, minha linda esposa.

AGRADECIMENTOS

Agradeço à minha linda esposa Janaina, por todo apoio e carinho, mesmo nos momentos da minha ausência durante estudos e atividades durante o curso.

Eu agradeço imensamente à toda Universidade Federal de Santa Catarina, seus professores e profissionais. Todos os anos como estudante nessa universidade foram mágicos.

Agradeço especialmente ao Cristiano Antonio de Souza, cuja orientação e motivação constante foi vital para realização desse trabalho de conclusão de curso.

Estendo o agradecimento aos meus colegas de curso com quem interagi na realização de tarefas e trabalhos ao longo do curso.

“Não há conhecimento que não seja poder.”
Ralph Waldo Emerson, 1870

RESUMO

A quantidade de equipamentos da IoT tem crescido consideravelmente nos últimos anos, a ponto de que a comunicação na internet entre máquinas supera a comunicação máquina homem. Esse novo conceito traz novos desafios, principalmente para área de segurança da informação, dada a sua natureza heterogênea e o surgimento de novas arquiteturas que suportam esse paradigma. A computação em nevoeiro vem sendo utilizada como uma camada intermediária para dar mais robustez às aplicações da IoT, através do fornecimento de armazenamento, processamento e serviços mais próxima dos dispositivos sensores. Este trabalho tem como objetivo avaliar a aplicabilidade de sistemas de detecção de intrusão na IoT através da camada de nevoeiro, utilizando de ferramentas já tradicionais para a segurança de redes de computadores.

Palavras-chave: Internet of Things. Fog Computing. Intrusion Detection.

ABSTRACT

The quantity of IoT equipment have considerably grown is the last years, to a point that the communication between machines in the internet surpass the communication man-machine. This new concept brings new challenges, primarily to the information security area, given its heterogenous nature and new architectures that support this paradigm. The fog computing has been utilized as an intermediate layer to give more strength to IoT applications, through supply of storage, processing and services. By having this centralizing characteristic, this work aims to evaluate the applicability of Intrusion Detection systems in IoT through the fog layer, utilizing the already traditional tools for computer networks security.

Keywords: Internet of Things. Fog Computing. Intrusion Detection.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IoT Internet of Things
SoC System on a Chip
RAM Random Access Memory
CPU Central Processing Unit
USB Universal Serial Bus
SD Secure Digital
TRRS Tip Ring Ring Sleeve
RCA Radio Corporation of America
GPIO General Purpose Input/Output
CSI Camera Serial Interface
IMU Inertial Measurement Unit
MQTT Message Queuing Telemetry Transport
NFC Near Field Communication
RFID Radio-Frequency Identification
IEEE Institute of Electrical and Electronics Engineers
UFSC Universidade Federal de Santa Catarina

SUMÁRIO

1 INTRODUÇÃO

A Internet of Things (IoT) é um novo paradigma que define equipamentos inteligentes, tecnologias e serviços que conectam objetos convencionais do nosso dia-a-dia, permitindo que estes possam se comunicar e cooperar. Os avanços tecnológicos dos microchips tornaram possível embutir uma antena em praticamente qualquer coisa, viabilizando a ideia de se ter um computador em qualquer lugar, a qualquer hora. A IoT tem tido um papel importante na proliferação de equipamentos conectados, espera-se que até 2020 tenham-se 212 bilhões de unidades implantadas pelo mundo (AL-FUQAHA, GUIZANI, *et al.*, 2015).

A IoT está presente em diversos segmentos de mercado, tendo impacto tanto em termos econômicos quanto financeiros. Através da IoT, equipamentos são capazes de cooperar para a tomada de decisão. Os equipamentos inteligentes podem interromper o fornecimento de calor, iluminação ou resfriamento em casas inteligentes, após a ocorrência de determinado evento, por exemplo. Em cidades inteligentes, a IoT pode alavancar o uso eficiente de energia, e contribuir para o melhor fluxo das vias através do controle de semáforos e locais de estacionamento. A segurança material e pessoal também pode ser aprimorada com o emprego de sistemas de detecção de fumaça inteligentes. Na saúde, sensores podem ser empregados para monitorar as condições físicas de pessoas, monitorando pressão sanguínea, temperatura do corpo e sinalizando uma equipe médica na necessidade de intervenção.

Essa presença pervasiva da IoT é garantida através do uso de equipamentos que utilizam baterias e que possuem CPU de baixo consumo. Esse baixo consumo, porém, reflete em baixo poder computacional e por esse motivo muitos serviços oferecidos pela IoT recorrem ao apoio da computação em nuvem. Na nuvem ocorrem a agregação e processamento dos dados gerados pela IoT. Apesar dessa arquitetura atender a necessidade de processamento, introduz o problema da latência, isso porque os datacenters dos provedores de serviços em nuvem são centralizados e em muitos casos distantes da origem dos dados.

Entre a camada de nuvem e de IoT existe a camada de nevoeiro, ou Fog computing. Essa camada intermediária traz para perto dos equipamentos da IoT o processamento e armazenamento, diminuindo a latência e viabilizando novos serviços. Nesse contexto, a nuvem passa a ser utilizada como camada gerencial, que agrega os dados enquanto é no fog que parte ou todo o processamento dos dados ocorre.

A IoT por ser um conceito emergente, ainda não possui consolidados alguns sérios aspectos, como o de segurança. A IoT tem natureza pervasiva, possui dispositivos que capturam informações e atuam no meio físico que estão inseridos. Dados médicos e identificáveis de pessoas transitam pela IoT, assim como dados de sistemas de alarmes, de controles de estoque, de sensores magnéticos de vias públicas e de outros serviços críticos. Portanto, dados sensíveis dos usuários são coletados, processados, transmitidos e armazenados através dos componentes de computação em névoa e IoT. Por isso, um ambiente IoT sem a devida segurança pode ser utilizado para causar sérios danos as vítimas.

Em setembro de 2016 uma série de ciberataques aos servidores de uma empresa chamada Dyn causaram a indisponibilidade de diversos grandes serviços da internet, como Amazon, Netflix e GitHub. Esses ataques ocorreram através de uma botnet chamada Mirai. A botnet Mirai, no ápice da sua atividade, chegou a produzir 600 Gbps de fluxo de dados que eram utilizados em ataques de negação de serviço. Durante o período de sua atividade, cerca de 65.000 novos equipamentos de IoT foram incluídos na botnet em 20 horas, chegando num número entre 200.000 e 300.000. Quase 50% dos equipamentos afetados estavam concentrados no Brasil, Colômbia e Vietnã. Entre os dispositivos da IoT afetados encontravam roteadores, impressoras, câmeras IP e dispositivos de gravação de vídeo (ANTONAKAKIS et al., 2017).

Em redes de computadores convencionais, Sistemas de Detecção de Intrusão (em inglês, Intrusion Detection Systems - IDS) são empregados como recurso de segurança capazes de detectar tentativas de invasões, ações suspeitas e atividades que possam prejudicar a integridade da rede e dos nodos. A pesquisa de segurança em IoT, porém, ainda é bastante recente e por essa razão ainda não existem sistemas de IDS consolidados.

Nesse trabalho, serão conduzidos estudos para a avaliação de ferramentas de IDS tradicionais no contexto da Fog, visto que esta está em posição estratégica e normalmente já age como um gateway para os equipamentos da IoT.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O objetivo geral deste trabalho é propor uma abordagem para detecção de intrusão para ambientes IoT baseados no contexto de Fog Computing utilizando IDS tradicionais.

1.1.2 Objetivos Específicos

Para alcançar o objetivo geral deste trabalho os seguintes objetivos específicos foram definidos:

- propor abordagem experimental de detecção de intrusão para ambientes IoT baseados em Fog computing;
- avaliar o IDS Suricata no método proposto;
- avaliar o IDS Snort no método proposto;
- comparar o desempenho relacionado a CPU e RAM de ambas as ferramentas tradicionais de detecção supracitadas.

1.2 ESTRUTURA DO TEXTO

No capítulo 2, são conceituados os temas IoT, Fog computing e IDS. O capítulo 3 contém a revisão sistemática da bibliografia e o estudo do estado da arte no tema de segurança de IoT através da Fog. O desenvolvimento do trabalho é detalhado no capítulo 4, enquanto a análise de resultados, a conclusão e discussão de trabalhos futuros são abordados nos capítulos 5 e 6 respectivamente.

2 CONCEITOS FUNDAMENTAIS

Nessa seção são apresentados os principais conceitos relacionados com a proposta deste trabalho, citando seus aspectos tecnológicos e relevância no contexto de tecnologia em geral.

2.1 INTERNET OF THINGS

A Internet of Things (IoT, Internet das Coisas em português) é um paradigma que tem a premissa de computação ubíqua através de coisas e objetos, dotados de sensores que através de esquemas de endereçamento único são capazes de interagir e cooperar para um propósito comum (ATZORI, IERA e MORABITO, 2010).

Não há ainda na literatura, entretanto, um consenso quanto a definição formal de IoT, uma vez que esse paradigma tem muitas visões diferentes, que variam principalmente do fato de que os órgãos reguladores, empresas e pesquisadores focam em diferentes aspectos tecnológicos (ATZORI, IERA e MORABITO, 2010).

A padronização da arquitetura da IoT é necessária para que se crie um ambiente competitivo para as organizações (AL-FUQAHA, GUIZANI, *et al.*, 2015).

A IoT oferece uma excelente oportunidade de investimento para fabricantes de equipamentos e desenvolvedores. Até o fim do ano de 2020 projeta-se que os equipamentos de IoT em operação no mundo chegarão a 212 bilhões de unidades, com potencial de mercado para gerar até 2025 um montante de $6.2 trilhões de dólares (AL-FUQAHA, GUIZANI, *et al.*, 2015).

O potencial oferecido por esse paradigma torna possível o desenvolvimento de um imenso número de aplicações. A IoT tem potencial para facilitar atividades relacionadas ao lar, automatizando rotinas simples, como, abrir um portão ou o acionamento automático do ar condicionado. Além disso, pode automatizar o controle de estoque na indústria, controle de tráfego de vias através de semáforos inteligentes, entre outras atividades. A IoT tem presença importante para as seguintes áreas: transporte e logística, saúde, ambientes inteligentes (casa, escritório, indústria), domínio pessoal e social (ATZORI, IERA e MORABITO, 2010).

2.1.1 Arquitetura da IoT

Segundo (AL-FUQAHA, GUIZANI, *et al.*, 2015), a necessidade de camadas flexíveis para a interação dos bilhões de equipamentos que compõe a IoT é crítica, no entanto, as diferentes propostas ainda não convergiram. De acordo com (GUBBI, BUYYA, *et al.*, 2013), a estrutura básica para a IoT é composta de três componentes: hardware, middleware e apresentação. A camada de hardware é composta de sensores, atuadores e dispositivos para comunicação.

A camada de middleware engloba a camada de armazenamento e processamento de dados. Já a camada de apresentação é acessada por serviços e usuários para a visualização e interpretação dos dados da IoT.

2.1.2 Dispositivos

Os dispositivos da IoT estão presentes em diversos segmentos de mercado e um número crescente de objetos estão sendo conectados à internet para troca de dados e informações. O conceito da IoT é de transformar objetos convencionais em objetos inteligentes, capazes de se comunicar entre si e coordenarem ações (AL-FUQAHA, GUIZANI, *et al.*, 2015). Nesta seção serão detalhados alguns equipamentos que são utilizados na IoT principalmente para o campo de pesquisa, tendo em vista que as organizações por trás desses equipamentos possuem programas de ensino e em alguns casos disponibilizam material de acesso livre e gratuito para o incentivo à pesquisa.

O Raspberry Pi é um pequeno computador Sistema em um Chip (em inglês System-on-a-chip - SoC) de arquitetura ARM utilizado em muitos projetos de IoT. A organização por trás desse projeto, a Raspberry Pi Foundation investe em treinamentos e educação utilizando o pequeno computador. Atualmente existem 8 diferentes modelos desse computador que variam de tamanho e capacidade computacional. A Tabela 1 lista os modelos disponíveis com suas respectivas configurações de hardware.

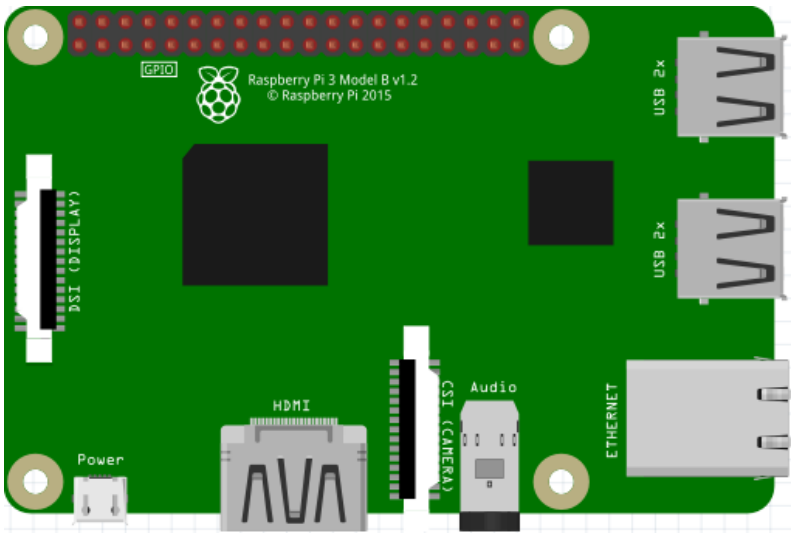
Tabela 1 - Lista de equipamentos Raspberry Pi.

Modelo	SoC	Frequência	RAM	USB	Ethernet	Wireless	Bluetooth
Pi Model A+	BCM2835	700MHz	512MB	1	Não	Não	Não
Pi Model B+	BCM2835	700MHz	512MB	4	100Base-T	Não	Não
Pi 2 Model B	BCM2836/7	900MHz	1GB	4	100Base-T	Não	Não
Pi 3 Model B	BCM387A0/B0	1200MHz	1GB	4	100Base-T	802.11n	4.1
Pi 3 Model B+	BCM2837B0	1400MHz	1GB	4	1000Base-T	802.11ac/n	4.2
Pi Zero	BCM2835	1000MHz	512MB	1	Não	Não	Não
Pi Zero W	BCM2835	1000MHz	512MB	1	Não	802.11n	4.1
Pi Zero WH	BCM2835	1000MHz	512MB	1	Não	802.11n	4.1

Fonte: Raspberry Pi Foundation.

A Figura 1 ilustra uma placa desse computador e suas conexões. O Raspberry Pi não possui entrada para disco convencional de computador, como por exemplo SATA, em vez disso utiliza de cartão SD. O equipamento possui saída de vídeo HDMI, enquanto alguns modelos também possuem saída de vídeo RCA e também uma entrada diretamente na placa para conexão de telas LCD, por exemplo. A alimentação do sistema é feita através de uma porta micro USB. Possui entrada de áudio TRRS de 3.5mm, também conhecida no Brasil como "P2". A conexão de rede é de 10/100Mb. Possui até 4 entradas USB de 2.0. A adição de novos recursos ao sistema pode ser também realizada através do conector GPIO de até 40 pinos, de acordo com o modelo. Por fim há também a possibilidade de conexão de uma câmera através da conexão CSI.

Figura 1 - Ilustração de uma placa Raspberry Pi 3 Model B.



Fonte: aplicação Fritzing.

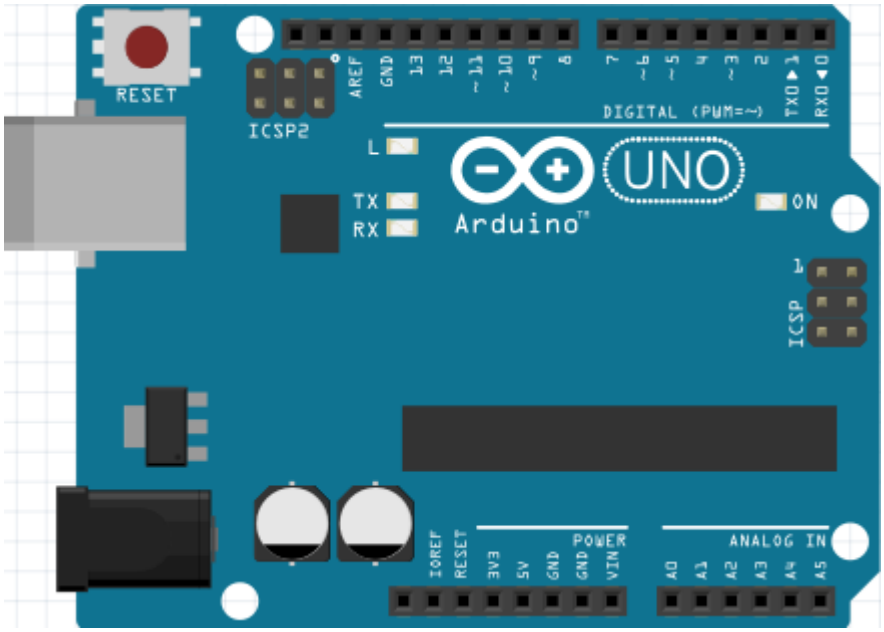
Outra placa bastante utilizada em projetos de IoT é o Arduino. Ela consiste em uma plataforma que compõe hardware e software open-source. Possui sensores e entradas que podem controlar luzes LED, acionar motores ou interagir com serviços online. Possui uma linguagem de desenvolvimento própria, assim como uma IDE.

Os hardwares open-source, por exemplo o Arduino, possuem as especificações de projeto das placas disponibilizadas para o público, de modo que empresas e outras organizações podem fazer uso.

A organização por trás do Arduino fornece equipamentos e serviços na nuvem especificamente para o desenvolvimento de IoT. Essa iniciativa se chama Arduino IoT Cloud e é composta pela placa, com sensores que capturam dados do mundo real e podem interagir com meio e serviços na nuvem através dos quais esses dados são visualizados. A Figura 2 ilustra uma placa Arduino modelo UNO WiFi REV2 dessa proposta para IoT.

Esse modelo inclui conexão WiFi com um co-processador de criptografia ECC608. Inclui um giroscópio através de um chip IMU.

Figura 2 - Ilustração de uma placa Arduino Uno.



Fonte: aplicação Fritzing.

2.1.3 Visão Geral de Segurança em IoT

Para (ATZORI, IERA e MORABITO, 2010) a segurança da IoT é um tema complexo porque os equipamentos geralmente não são fisicamente supervisionados, por isso estariam suscetíveis a ataques físicos. Além disso, dada que a maior parte da comunicação é wireless, esta seria facilmente comprometida. Também por ser composta principalmente de equipamentos de baixo poder computacional, não são capazes de implementar recursos complexos de suporte a segurança. Tipicamente algoritmos de criptografia utilizam muitos recursos computacionais e por essa razão não seriam diretamente aplicáveis para IoT. Para (AL-FUQAHA, GUIZANI, *et al.*, 2015), outro fator que dificulta a segurança da IoT é a sua natureza heterogênea.

2.2 CLOUD COMPUTING

A sugestão do que viria a ser a computação em nuvem foi dada nos anos 1960s por John McCarthy quando foi feita a proposta do conceito de "Utilitário de Computação". (SHAW e SINGH, 2014).

A definição de computação em nuvem, segundo o Instituto Nacional de Padrões e Tecnologia (National Institute of Standards and Technology - NIST) é: "Computação em Nuvem é um modelo que permite acesso pela rede ubíquo, conveniente e sob demanda a um conjunto compartilhado de recursos computacionais configuráveis (p. ex., redes, servidores, armazenamento, aplicações, e serviços) que podem ser rapidamente provisionados e liberados com o mínimo esforço de gerenciamento ou interação com o provedor do serviço". (MOGHADDAM, ROHANI, *et al.*, 2015).

A Computação em Nuvem pode ser implantada em três modelos: Pública, Privada e Híbrida (PRAJAPATI, SHARMA e BADGUJAR, 2018).

- Nuvem Pública: esse tipo de nuvem possibilita o acesso a recursos compartilhados, num formato conhecido como multi-inquilinos. O conjunto de recursos providos nesse modelo adotam o formato de pagamento por uso.
- Nuvem Privada: quando usuários ou organizações precisam de um controle mais rígido sobre seus dados e acessos aos mesmos, esse modelo costuma ser adotado. A nuvem privada é geralmente oferecida também por fornecedores de nuvem pública, mas num molde em que os recursos não são compartilhados.
- Nuvem Híbrida: junção dos dois modelos supracitados, nesse caso os recursos mais sensíveis são mantidos em um ambiente mais controlado, na nuvem privada, enquanto outros recursos de armazenamento e processamento são delegados para a nuvem pública, permitindo um cenário mais seguro e com capacidade de escalabilidade.

Além dos diferentes tipos de implantação possíveis para a computação em nuvem, há também os modelos de serviços ofertados que podem ser: Infraestrutura como Serviço (Infrastructure as a service - IaaS), Plataforma como Serviço (Platform as a service PaaS) e Software como Serviço (Software as a service - SaaS).

- IaaS: o provedor desse modelo fornece aos clientes infraestrutura computacional física ou virtual, de modo que os clientes possam instalar sistemas operacionais, middleware e ferramentas que necessitarem.
- PaaS: na plataforma como serviço, o cliente não possui controle sobre a infraestrutura, já que essa é abstraída, mas tem à disposição um ambiente para desenvolvimento e implantação de soluções de software.
- SaaS: são oferecidos softwares hospedados em nuvem, que não requerem instalação, pouca ou nenhuma configuração. O contratante desse modelo não tem controle sobre a infraestrutura ou plataforma.

2.3 FOG COMPUTING

O termo Fog Computing (em português Computação em Nevoeiro) é geralmente associado aos trabalhos de pesquisa da Cisco. Esse conceito recebe variadas definições da indústria e da academia (MUNIR, KANSAKAR e KHAN, 2017). Entretanto, em 2015 surgiu uma coalizão da indústria e academia que fundou a OpenFog Consortium e que define a Fog Computing como: "Fog Computing é uma arquitetura horizontal a nível de sistema que distribui recursos e serviços de computação, armazenamento, controle e rede em qualquer lugar em continuidade a "Nuvem das Coisas" (MUNIR, KANSAKAR e KHAN, 2017).

Na prática, a Fog computing é uma extensão da nuvem, mais próxima das coisas da IoT. A Fog Computing age como um intermediário entre a nuvem e os dispositivos da IoT, trazendo processamento, armazenamento e serviços de rede para perto dos dispositivos finais. Esses dispositivos da Fog são chamados de nodos do nevoeiro. Qualquer dispositivo com capacidade de processamento, armazenamento e conectividade de rede pode ser um nó na Fog. (ATLAM, WALTERS e WILLS, 2018).

Os dispositivos da IoT estão se tornando essenciais para a permitir a troca de dados entre serviços e eletrônicos, mas eles geralmente não são capazes de processar esses dados e fornecê-los a serviços que usem esses dados, por consequência da sua natureza intrínseca de limitação de recursos. Nesse ponto, a Fog Computing pode auxiliar, coexistindo com os equipamentos da IoT e cooperando com a nuvem. (PULIAFITO, MINGOZZI, *et al.*, 2019).

A integração entre a Computação em Nuvem e a IoT permite que dispositivos de baixa capacidade da IoT deleguem atividades complexas para a nuvem, fazendo uso da sua

capacidade computacional e de armazenamento. No entanto, a natureza centralizada da nuvem pode ser um problema para serviços que demandem baixa latência, visto que os equipamentos que geram os dados podem estar longe do data center da nuvem (PULIAFITO, MINGOZZI, *et al.*, 2019).

Gerenciar os dados gerados pela IoT é um dos maiores desafios quando são implantados tais sistemas. Sistemas de IoT baseados na nuvem possuem aspectos de serem sistemas geralmente de larga escala, heterogêneos e, dependendo da localização dos dispositivos da IoT e da nuvem, podem sofrer de alta latência. Uma solução para esses desafios é o emprego da Fog Computing para descentralizar as aplicações, o gerenciamento e a análise dos dados gerados pela IoT (IORGA, FELDMAN, *et al.*, 2018).

A Fog é um modelo em camadas que permite acesso ubíquo a recursos de nuvem escaláveis. Esse modelo facilita a implantação de aplicações e serviços distribuídos e de baixa latência. Os nós da Fog podem ser físicos ou virtuais, estar distribuídos em uma arquitetura horizontal (com suporte a isolamento) ou vertical (com suporte a federação) e geralmente estão entre os dispositivos da IoT e os serviços em nuvem. (IORGA, FELDMAN, *et al.*, 2018).

2.4 INTRUSION DETECTION SYSTEM

Segundo o (NIELES, DEMPSEY e PILLITTERI, 2017), a detecção de intrusão é o processo de monitoramento de um sistema computacional ou de rede afim de detectar sinais de possíveis ou iminentes incidentes de violação de políticas de segurança. Em sua grande maioria, os incidentes são atividades maliciosas, mas há casos em que estes eventos são gerados por uso incorreto de recursos. Para (GHORBANI, LU e TAVALLAEE, 2010), a detecção de intrusão é o processo de identificar (e possivelmente responder) a uma atividade maliciosa que tenha como alvo computadores e recursos de rede. Nesse contexto, qualquer software ou hardware que possa monitorar, identificar ou responder a esses eventos é relevante para um Sistema de Detecção de Intrusão (Intrusion Detection System - IDS).

Uma intrusão, de acordo com (GHORBANI, LU e TAVALLAEE, 2010), é definida como uma tentativa de uso indevido ou de tornar indisponível um recurso computacional.

Intrusões geralmente ocorrem a partir de uma brecha na segurança e por esse motivo devem ser rapidamente detectadas.

Um IDS é um sistema responsável por detectar e responder a essas intrusões. Esses sistemas podem ser empregados em computadores ou em redes de computadores e analisam toda a atividade para encontrar possíveis ataques a segurança dos recursos monitorados. Quando monitoram um computador, analisam seus processos e o comportamento do usuário com o objetivo de encontrar desvios. Em rede, monitoram toda a atividade de comunicação e podem comparar os pacotes transeuntes com um conjunto de ameaças conhecidas ou aplicar inteligência artificial para categorizá-los.

O trabalho de Denning (DENNING, 1987) descreve a automação do processo de detecção através da análise do comportamento de uso de programas. Em seu trabalho, ela sugere que as invasões ocorrem do uso incomum de programas comuns.

Segundo (BISHOP, 2004), existem 4 características importantes para um IDS: detectar uma ampla variedade de intrusões, isto é, ser capaz de detectar incidentes que ocorram dentro e fora do ambiente, sendo também capaz de detectar ataques conhecidos assim como novos ataques. A detecção do evento deve ser oportuna, isto é, nem sempre é possível que seja analisado todo o fluxo de uma rede em tempo real, visto que isso poderia causar um gargalo na rede. Dessa forma, a detecção deve ocorrer tão logo quanto possível. Quanto a informação de que um evento foi verificado, ela deve ser simples e de fácil leitura, já que um IDS pode ser responsável por um elevado número de sistemas, a interface com o usuário é um ponto importante. O último ponto ressaltado trata da acurácia do IDS. Um IDS precisa ser preciso, porque falsos positivos podem ser custosos e fazer com que o sistema fique desacreditado se esses casos forem recorrentes. Já um falso negativo pode ter impacto ainda maior, isso porque incidentes de segurança deixam de ser notificados.

2.4.1 Método de Detecção

Os IDS podem caracterizar um objeto de análise como sendo ou não característico de um incidente com base em diferentes modelos de detecção. Uma abordagem de detecção usada por IDS constitui de um conjunto já conhecido de incidentes. Cada novo objeto de análise é submetido à uma comparação para averiguar se suas características são identificadas em algum exemplo de incidente. Outro modelo de IDS aborda o problema de forma diferente:

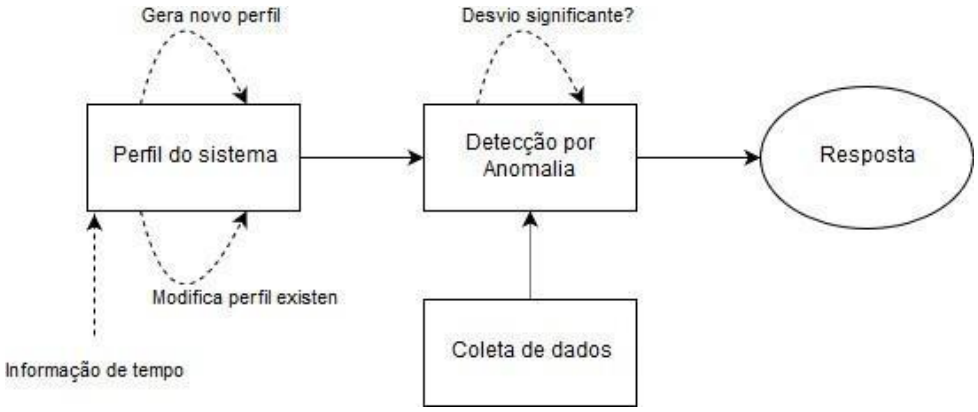
a detecção busca encontrar anomalias comportamentais para determinar se há ou não um incidente de segurança.

Para a detecção de ataques que ainda não sejam conhecidos, um modelo de detecção por anomalia deve ser utilizado por um IDS. Esse modo de detecção define um modelo padrão de comportamento de um sistema e detecta desvios. São utilizadas técnicas de data mining, clustering e estatística para a criação desses modelos (GHORBANI, LU e TAVALLAEE, 2010).

Uma forma de detecção por anomalia considera um conjunto de características que definem um comportamento e espera que esse comportamento atenda a um critério numérico. Quando a estatística computada do comportamento está fora da medida, o sistema então reporta uma anomalia e o caso é tratado como um incidente. O trabalho de Denning especifica três diferentes modelos estatísticos que são usados por sistemas baseados em detecção por anomalia. Um desses modelos, considera que eventos devem ocorrer *n* vezes e no máximo *m* vezes para um período. Se passado o período o valor computado não estiver no intervalo, tem-se uma anomalia. O segundo modelo utiliza de recursos estatísticos como média e desvio padrão, além de outras medidas, para categorizar um evento. O terceiro modelo, conhecido como modelo Markov, onde o sistema é analisado dado um período e o conjunto de observações modelam os estados possíveis. Ao passar do tempo, o modelo de treinamento conhece uma gama de estados possíveis para os processos de modo que é capaz de categorizar incidentes (BISHOP, 2004).

A Figura 3 ilustra a operação básica da operação da detecção por anomalia. Este sistema consiste de quatro componentes, a coleta de dados, o perfil de um sistema normal, a detecção de anomalia e uma resposta. Os dados de operação de um equipamento ou de uma rede são obtidos e salvos no recurso de coleta de dados. Técnicas de modelagem são empregados para a criação do perfil do sistema. O componente responsável pela análise de dados entrantes checa o perfil do dado e, caso esse desvie de forma significativa do padrão de sistema conhecido como normal então o incidente é reportado.

Figura 3 - Sistema de detecção por anomalia.



Fonte: Adaptado de (GHORBANI, LU e TAVALLAEE, 2010)

Um IDS baseado em um modelo de detecção por regras/assinatura utiliza um conjunto de regras de problemas conhecidos ou de possíveis ameaças ao sistema que podem ser exploradas. Esse conhecimento é armazenado em um conjunto de regras, as quais os dados analisados são submetidos para comparação. Essa etapa, geralmente é realizada por um sistema especialista. Esse modelo é incapaz de reconhecer novos tipos de ataque por conta própria, já que não há qualquer recurso de adaptabilidade. As regras são estáticas e cabe ao operador do IDS atualizar periodicamente o conjunto de regras (BISHOP, 2004).

Essa técnica de operação de IDS pode ser um fardo para a operar já que requer intervenção para a atualização e extensão da base de regras toda vez que surgem novas formas de intrusão. Além disso, a tarefa de desenvolver essas regras pode ser bastante complexa. Por consequência, na prática, os bancos de dados de regras para detecção por abuso logo tornam-se desatualizadas e obsoletas (GHORBANI, LU e TAVALLAEE, 2010). No entanto, essa forma de detecção em contraste com a detecção por anomalia, requer menos recursos computacionais para operar, podendo ser mais adequada para dispositivos restritos, como nodos sensores ou de nevoeiro.

2.4.2 Categorias de Detecção

Um IDS pode ser empregado para garantir a segurança de diferentes recursos em diferentes contextos. Um IDS pode ser implantado em um único computador ou então ser

estrategicamente posicionado em uma infraestrutura de rede para checar eventos em toda a rede.

Um Sistema de Detecção de Intrusão de Baseado em Host (Host Intrusion Detection System - HIDS), analisa várias fontes de dados de um único computador, tais como chamadas de sistema, acesso a arquivos, conteúdo da memória e registros. Como um HIDS está em operação em apenas um computador, para garantir a segurança de toda uma rede, seria necessário que todo computador tivesse instalado tal sistema (GHORBANI, LU e TAVALLAEE, 2010).

A estrutura de HIDS pode ser composta de um agente, que coleta registros do computador monitorado e submete para um diretor, que fica responsável pela análise. O diretor ao analisar os dados recebidos do agente e constatar que houve uma violação, passa a informação da violação para um terceiro, o notificador, que então realiza a notificação do evento conforme estiver configurado. O notificador pode diretamente acionar os agentes para ajustar os registros obtidos que são enviados ao diretor. Geralmente os agentes determinam o que é enviado para um diretor realizando uma rápida checagem do conteúdo coletado (BISHOP, 2004).

Através da coleta e análise de registros, um HIDS é capaz de detectar, por exemplo, usuários malfeitores que estejam tentando cometer ações não autorizadas no sistema ou que estejam tentando modificar ou deletar arquivos do sistema. As chamadas de sistema para modificação de arquivos também podem ser usadas como dados para detecção de um usuário mal-intencionado (GOODRICH e TAMASSIA, 2010).

Um Sistema de Detecção de Intrusão de Rede (Network Intrusion Detection System - NIDS) intercepta e analisa os dados que transitam em uma rede de computadores. Essa análise pode ocorrer em toda a rede, em uma parte da rede ou entre a rede e a Internet. Geralmente a parte dos pacotes que contém os dados, pertencentes a camada de Aplicação são ignorados por várias razões, entre elas questões de privacidade, por conta do grande volume de diferentes protocolos e também porque pode ocorrer de haver uma sobrecarga e um ponto de lentidão tendo origem no NIDS, entretanto, as demais camadas são analisadas (GHORBANI, LU e TAVALLAEE, 2010).

Um agente de rede de um IDS é capaz de detectar uma gama diferente de incidentes de segurança quando comparado com um agente que monitora um host. Os tipos de incidentes detectados por um NIDS são, por exemplo, de ataques de negação de serviço e análise de portas. Quando os pacotes de dados não são ignorados, esse tipo de análise recebe o nome de análise de conteúdo. Nesse modelo de análise, o agente é colocado estrategicamente em pontos na rede que dão acesso ao fluxo de dados de toda a rede.

Se, porém, a comunicação entre os nós for criptografada, a análise de conteúdo não será possível (BISHOP, 2004).

Um NIDS centralizado tende a causar latência na rede e perder pacotes em redes de alta velocidade que tenham alto volume de dados. Para endereçar esse problema, um sistema baseado em agentes distribuídos pode ser adotado. Quanto ao posicionamento de um NIDS, quando este está posicionado na borda da rede, para impedir ataques externos, este pode não ser plenamente capaz de detectar um incidente vindo da rede interna. Outro ponto de atenção quanto a implantação de um NIDS citado por (GHORBANI, LU e TAVALLAEE, 2010) aborda que se o NIDS possui uma frequência de análise contínua e comportamento ativo na resposta de um incidente, em um caso de falso positivo um ou mais recursos computacionais poderá ser afetado pelo bloqueio indevido das comunicações de rede.

2.4.3 Arquitetura de IDS

A implantação de um IDS pode ser centralizada ou distribuída. Na abordagem centralizada as atividades de captura e análise das informações são agrupadas em apenas um ponto. Por outro lado, na abordagem distribuída, as tarefas de um IDS são delegadas para diferentes pontos da rede.

Diferente da arquitetura centralizada, em uma arquitetura distribuída a coleta de dados ocorre em cada segmento de rede e então é processada em uma ou mais centrais. Essa abordagem pode ser inclusive expandida com o conceito de agentes, onde estes seriam responsáveis também pela análise dos dados coletados, apenas reportando para um notificador os incidentes (GHORBANI, LU e TAVALLAEE, 2010).

Quando implantados agentes, estes podem ser dispostos de forma hierárquica para a coleta e análise de dados. Nesse formato, geralmente existem quatro componentes: agentes, filtros, transceptores e monitores. Nessa estrutura, os filtros formam um sistema de

mensageria por subscrição no qual os agentes se inscrevem e solicitam os dados que desejam receber para análise e cabe aos filtros implementar todos os recursos necessários a nível de sistema para obtenção dos dados solicitados pelos agentes, inclusive com os critérios por eles especificados. Os transceptores recebem dos agentes as notificações de incidentes que são encaminhadas para o monitor, que tem acesso a todos os dados. Os transceptores também podem enviar para as agentes operações requisitadas pelo monitor, como por exemplo, solicitar a interrupção da análise (GHORBANI, LU e TAVALLAEE, 2010).

2.4.4 Frequência da análise

A frequência da análise de recursos monitorados por um IDS tem um importante papel na detecção em tempo oportuno e no possível impacto de sobrecarga que o IDS pode exercer, por isso a frequência deve ser avaliada para se adequar ao cenário e arquitetura do IDS.

A análise periódica de um IDS coleta uma série de dados do ambiente monitorado para um dado período e analisa o conjunto dessa coleta em busca de, por exemplo, configurações erradas, programas vulneráveis assim por diante. Essas verificações incluem as configurações do ambiente, verificação de senhas fracas, conteúdos de arquivos nas pastas dos usuários e configurações de serviços de internet (DEBAR, 2000).

A análise contínua de um IDS significa que o sistema monitora em tempo real os eventos de um sistema ou rede. Quando sob muita carga, a análise contínua pode fazer com que sejam perdidos pacotes de rede, por exemplo, fazendo com que possíveis incidentes ocorram sem serem detectados. A rede também pode sofrer problemas de conexão caso o IDS esteja capturando todos os pacotes da rede para repassá-los adiante (SCARFONE e MELL, 2007).

A análise contínua emprega sensores que são colocados online, isso é, monitorando todo o tráfego passante. O principal motivo de ter um sensor monitorando o tráfego em tempo real é de tornar possível a ação do IDS mais imediata quanto possível. Um IDS de análise contínua para ser efetivo, necessita de muito mais recursos computacionais que um de análise periódica (GHORBANI, LU e TAVALLAEE, 2010).

2.4.5 Comportamento pós detecção

Uma outra forma de categorização dos IDS é em relação ao seu mecanismo de resposta. Os passivos notificam quando um incidente ocorre enquanto os ativos tomam medidas de resposta automáticas (GHORBANI, LU e TAVALLAEE, 2010).

O comportamento ativo de um IDS diz respeito a forma como são tratados os incidentes de segurança. Quando um incidente é detectado, nesse modelo o IDS notifica a ocorrência e adota imediatamente medidas reativas e pró ativas como resposta a um ataque. Essa resposta ao incidente pode ser disparada totalmente autônoma, sem ação humana. A maior parte dos IDS que implementam esse comportamento a invasões segue um conjunto de regras que especifica quais devem ser as ações tomadas para cada caso. Entretanto, uma limitação nesse formato é que o sistema não leva em consideração efeitos colaterais que uma interrupção total ou parcial na comunicação de um computador, por exemplo, pode ter para os usuários desse sistema. Esse modelo de ação ainda pode ser categorizado de duas maneiras: reativo e proativo. Uma resposta reativa significa que o sistema age assim que uma intrusão é detectada, enquanto uma resposta proativa significa que o sistema executa uma série de ações com intuito de impedir que ocorra uma invasão (GHORBANI, LU e TAVALLAEE, 2010).

Um IDS com comportamento passivo apenas registra e notifica a identificação de um incidente de segurança. Os dados de um computador ou de uma rede podem ser monitorados e em uma ocorrência de ataque o IDS emite uma notificação. Uma resposta manual deve ser tomada para que o incidente seja tratado, o que pode ser um problema dependendo do tempo de resposta ao evento. Uma demora de horas, por exemplo, colocaria em risco os sistemas e poderia levar ao sucesso de uma invasão (GHORBANI, LU e TAVALLAEE, 2010).

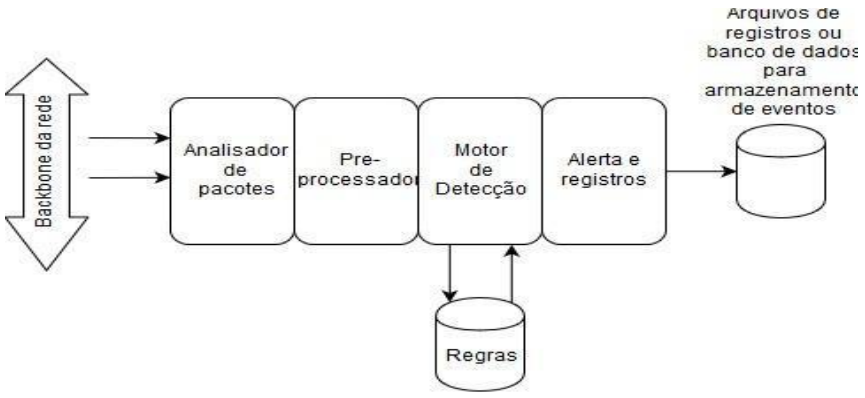
Um incidente de segurança pode ser confinado a um sistema de modo que o atacante não tenha acesso a outros recursos. Dessa forma, o IDS não interfere de forma alguma com o incidente em andamento, apenas monitora e registra todas as ações tomadas no ataque. Essa técnica pode auxiliar no detalhamento de um ataque, seus métodos e finalidades (BISHOP, 2004).

2.4.6 Snort

O Snort é um IDS livre de código aberto que vem sendo desenvolvido desde 1998. O projeto inicial foi iniciado por Martin Roesch como uma ferramenta de análise de pacotes de rede. Em 1999, Roesch publicou um trabalho onde detalha a criação de um mecanismo de análise de padrões pra o Snort, já o classificando-o como ferramenta de detecção de intrusão (WHITE, FITZSIMMONS e MATTHEWS, 2013).

Sua arquitetura é composta analisador de pacotes, pré-processador, motor de detecção e alertas e registros. O analisador de pacotes captura os pacotes da rede na qual o IDS está implantado e os envia para o pré-processador que avalia o comportamento das comunicações. Os pacotes em seguida são enviados para o motor de detecção, que compara os pacotes que chegam com um conjunto de regras. Na eventual ocorrência de um incidente de segurança, mecanismo de alerta e registros notifica do incidente e registra as informações em um arquivo de registro ou banco de dados (PARK e AHN, 2016). A Figura 4 ilustra a arquitetura da aplicação.

Figura 4 - Arquitetura do Snort.



Fonte: Adaptado de (PARK e AHN, 2016).

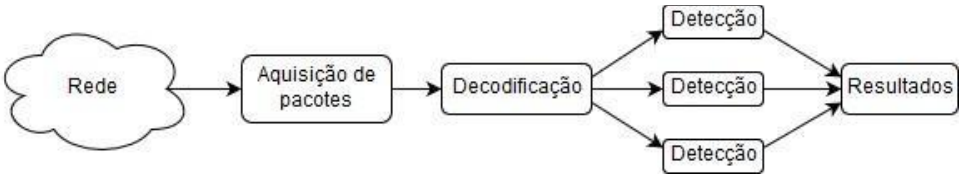
O Snort em sua operação utiliza somente um thread de processamento e por essa razão é capaz de processar entre 100 e 200 megabits por segundo antes que atinja o limite de processamento de um único processador e comece a despejar pacotes para compensar. Para

que o Snort seja capaz de fazer uso de múltiplos processadores, é necessário que várias instâncias dele sejam iniciadas. (ALBIN, 2011).

2.4.7 Suricata

O Suricata é um IDS livre de código aberto lançado em 2009 pela Open Information Security Foundation (OISF). É um NIDS que opera com método de detecção por abuso e que, diferentemente do Snort, possui uma arquitetura multi-thread. O projeto recebeu significativo investimento do Departamento de Segurança Interna dos Estados Unidos e de outros programas e departamentos de segurança americanos (ALBIN, 2011). O desenvolvimento do Suricata apesar de possuir código fonte desenvolvido para o projeto, baseia-se bastante nos conceitos do Snort a ponto de ser possível utilizar regras do Snort no Suricata (ALBIN, 2011). Dada a arquitetura multi-thread do Suricata e a sua capacidade de utilizar regras do Snort, ele pode ser um substituto para redes em que haja um grande fluxo de dados onde um processo single-thread poderia não ser suficiente. A Figura 5 ilustra a arquitetura desse IDS.

Figura 5 - Arquitetura do Suricata.



Fonte: Adaptado de (PARK e AHN, 2016)

Um motor de detecção multi-thread pode realizar decisões inteligentes em como dividir o processamento e coordenar a detecção nos threads (ALBIN, 2011).

3 REVISÃO DA LITERATURA

Uma revisão sistemática da literatura foi realizada para identificar o estado da arte na detecção de intrusão em Fog para IoT. Foram utilizadas as bibliotecas da Association for Computing Machinery (ACM), Elsevier, Institute of Electrical and Electronics Engineers (IEEE) e Springer. Os trabalhos selecionados estão em inglês, idioma predominante nas publicações científicas nessas plataformas. As palavras chave para a busca foram “Intrusion Detection System”, “Fog Computing”, “Internet of Things”, “Suricata” e “Snort”. No processo de seleção preliminar, para filtrar os resultados de modo a serem mais específicos ao escopo do trabalho foram utilizadas combinações lógicas das palavras chaves. Em seguida foi feita a leitura do resumo de cada trabalho. A Tabela 2 ilustra os resultados obtidos na revisão.

Tabela 2 - Revisão sistemática da bibliografia com os termos chave.

Palavras chave	Ocorrências
"Intrusion Detection"OR "IDS"	798155
"Internet of Things"OR "IoT"	107415
"Fog Computing"OR "Fog"	111435
"Snort"	11583
"Suricata"	1023
("Intrusion Detection"OR "IDS") AND ("Internet of Things"OR "IoT")	11792
("Intrusion Detection"OR "IDS") AND ("Fog Computing"OR "Fog")	6513
("Intrusion Detection"OR "IDS") AND ("Internet of Things"OR "IoT") AND ("Fog Computing"OR "Fog")	704
("Intrusion Detection"OR "IDS") AND ("Internet of Things"OR "IoT") AND ("Fog Computing"OR "Fog") AND ("Snort"OR "Suricata")	30

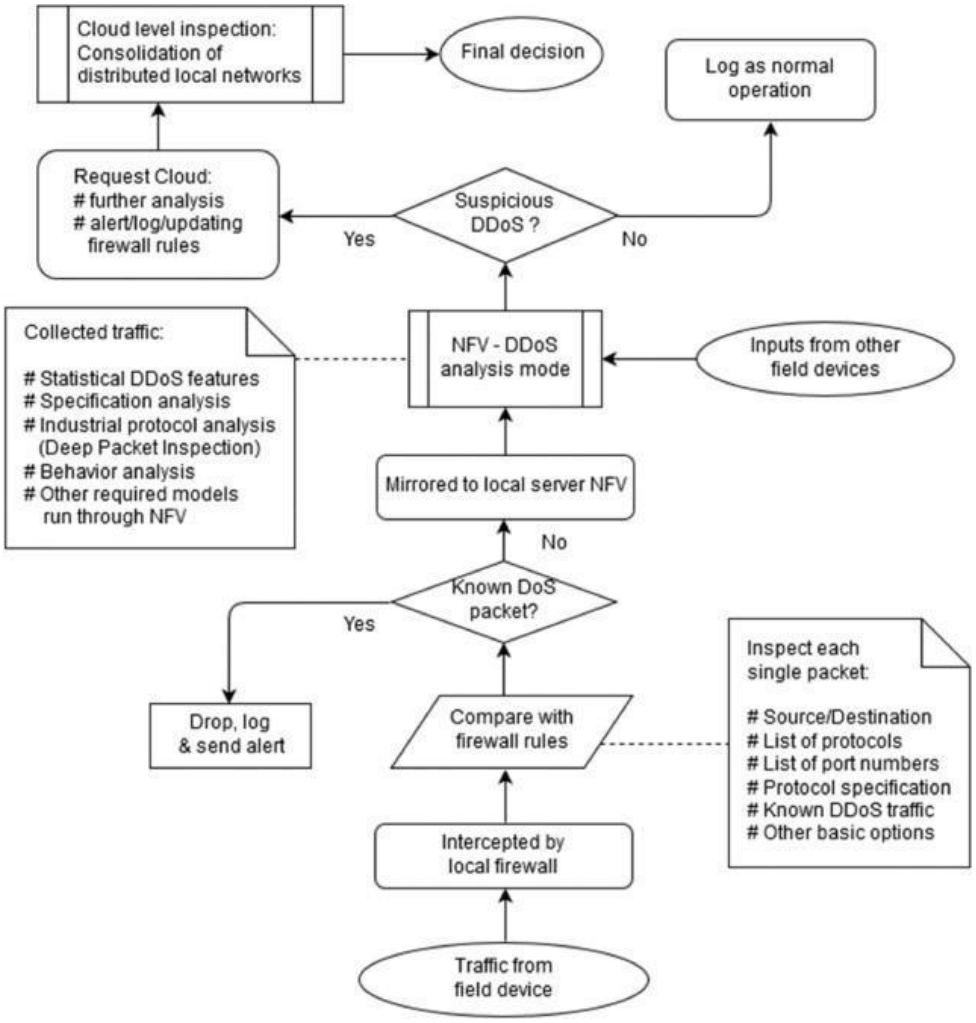
Fonte: autor (2019).

Ao usar a composição dos termos como parâmetro de filtro para a busca e aplicar critérios de inclusão e exclusão dos trabalhos, foram selecionados os três trabalhos a seguir.

O trabalho (ZHOU, GUO e DENG, 2019) propõe uma abordagem baseada em Fog Computing para mitigação de ataques de negação de serviço a equipamentos industriais de IoT. Para essa finalidade, foi utilizada uma abordagem que analisa os dados em três camadas. A primeira análise ocorre em tempo real na comunicação, consistindo de um filtro de firewall. A segunda ocorre offline, e analisa as especificações dos dados e a terceira ocorre

na nuvem, através da consolidação das informações de segurança. A Figura 6 contém o diagrama de operações do esquema proposto neste trabalho.

Figura 6 - Diagrama dos procedimentos do esquema proposto.



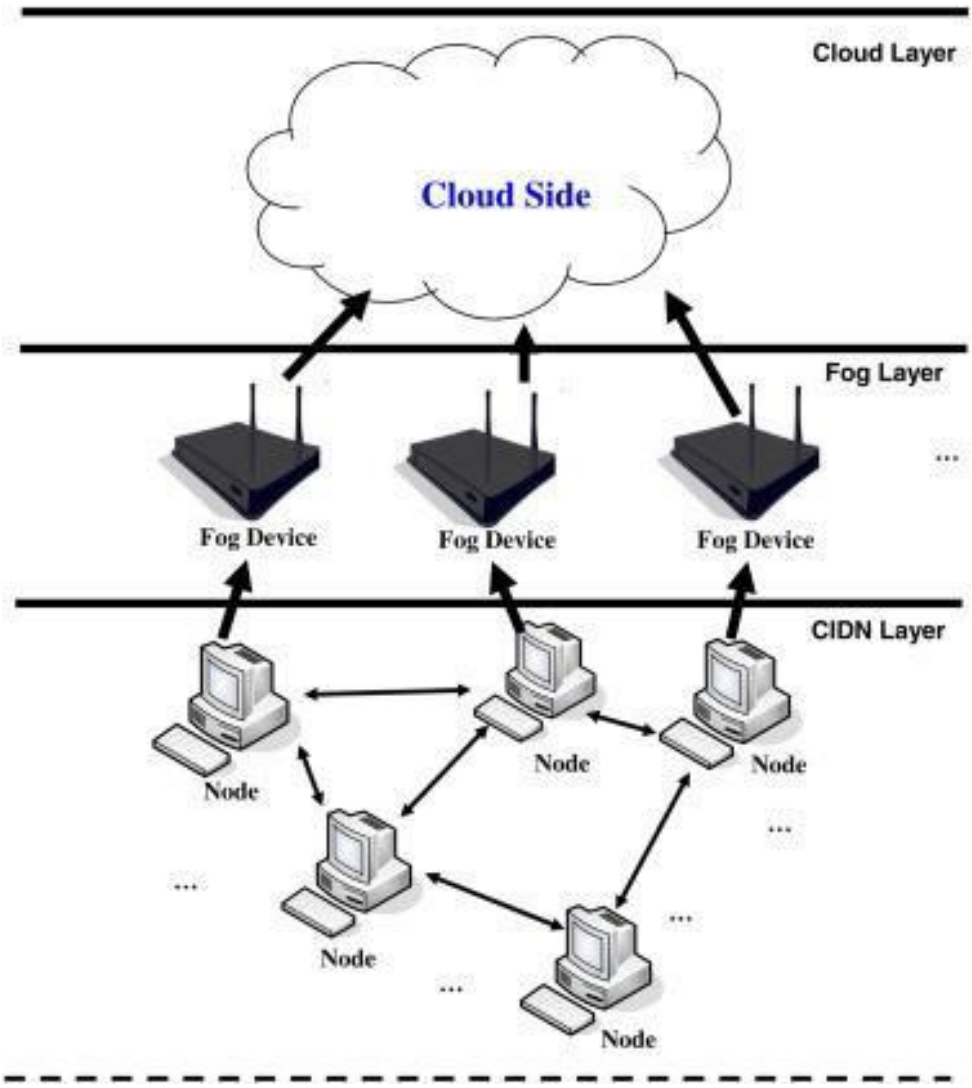
Fonte: (ZHOU, GUO e DENG, 2019).

Concluiu-se desse trabalho que a falta de estudos semelhantes, isto é, que avaliam a segurança de equipamentos da IoT na área industrial dificulta a análise comparativa. Apesar disso, sob a ótica de acurácia e tempo de resposta, o modelo distribuído proposto foi eficiente e auxiliou a aliviar a carga de rede e de processamento na nuvem.

A pesquisa conduzida por (WANG, MENG, *et al.*, 2018) descreve um IDS colaborativo com detecção por abuso baseado em Fog com a característica de preservar a privacidade dos dados analisados. Utilizou-se uma arquitetura distribuída porque um nodo poderia ser sobrecarregado com o tráfego para analisar. Para aprimorar a acurácia da

detecção, essa proposta emprega uma rede colaborativa de IDS que coletam e trocam informações entre si. A privacidade dos dados analisados é alcançada com o uso de técnicas de criptografia que avaliam os dados criptografados.

Figura 7 - Esquema geral do framework.



Fonte: (WANG, MENG, *et al.*, 2018).

A Figura 7 ilustra a camada CIDN, que é composta de nodos, cada um com o IDS Snort instalado. Nessa camada eles trocam informações de segurança para aprimorar a acurácia do sistema. A camada de fog é utilizada para a tomada de decisão de segurança e a de nuvem para que sejam executados procedimentos mais exigentes de recursos

computacionais. O estudo conclui que o modelo proposto foi mais eficiente que propostas semelhantes. Para trabalhos futuros os autores sugerem o estudo desse modelo utilizando-se de um método de detecção baseado em anomalia.

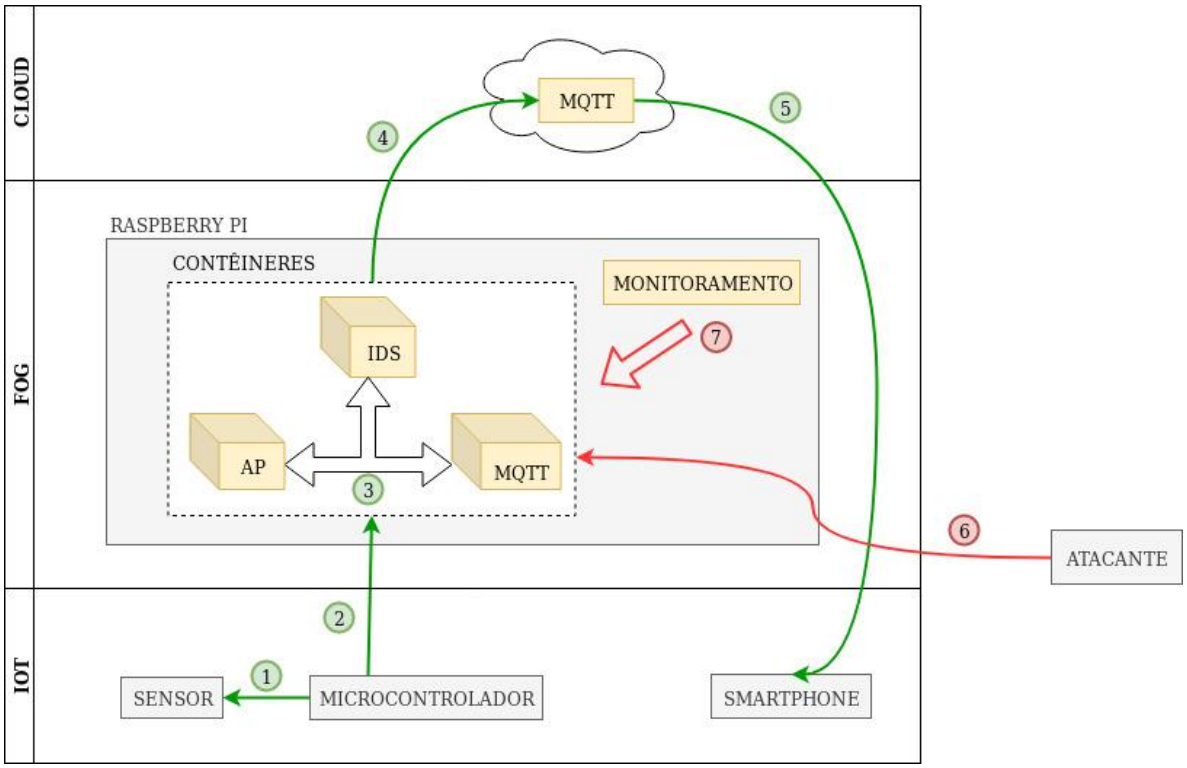
A proposta de (SOHAL, SANDHU, *et al.*, 2018) é a de identificar nodos maliciosos na borda da rede, através da Fog e da Cloud. O estudo utiliza de um framework que usam um modelo de Markov para identificar nodos na borda que são maliciosos, assim como aqueles que não o são. Os experimentos conduzidos pelos pesquisadores apontaram que o framework é eficiente e que, diferentemente de outras propostas, têm a capacidade de reverter a categorização de um nodo que tenha sido incorretamente marcado como malicioso. Futuramente os pesquisadores pretendem integrar um framework de testes de penetração em alta escala.

4 AMBIENTE EXPERIMENTAL

A área de segurança para IoT ainda possui muitos desafios e não há ainda métodos consolidados de segurança e principalmente de detecção de intrusão. Nesse trabalho, para analisar a aplicabilidade do Snort ou Suricata como recurso de IDS para a IoT através da Fog, foi realizado um experimento em que ambas as soluções foram instaladas em um equipamento Raspberry Pi.

A Figura 8 descreve o fluxo de operação que será executado no ambiente experimental. As etapas numeradas de 1 a 5 correspondem ao fluxo normal de operação para um ambiente IoT.

Figura 8 - Fluxo de operação



Fonte: O autor (2019).

Os passos são:

1. microcontrolador recupera medições de calor e umidade do sensor;
2. microcontrolador envia essas informações para o nodo fog;

3. o microcontrolador se conectou ao Fog através do contêiner AP e enviou a mensagem com as medições para o serviço MQTT. Toda comunicação de entrada e saída de rede dos contêineres passam pelo contêiner IDS;
4. a mensagem recebida pelo serviço de MQTT é encaminhada para a nuvem;
5. o serviço na nuvem notifica de uma nova mensagem para a aplicação operando no smartphone;

Os passos 6 e 7 são referentes ao processo de ataque e monitoramento de recursos do nodo da Fog. Conforme listado:

6. atacante tenta sobrecarregar serviço MQTT;
7. ferramenta de monitoramento registra uso de CPU e RAM durante ataques;

Na estrutura o Raspberry Pi faz o papel de recurso computacional da camada Fog, fornecendo segurança, serviço de mensageria para recebimento de dados dos nós da camada IoT e conectividade de rede.

A conexão entre a camada da IoT e a camada Fog ocorre através de uma rede Wireless. Para criar essa rede, foram utilizadas as aplicações wpa_supplicant, que fornece a rede ao qual o nodo IoT se conecta e a aplicação DHCPD, que fornece o serviço de DHCP para disponibilização de um endereço IP.

Para o serviço de mensageria, foi utilizada a aplicação Mosquitto, solução open source fornecida pela fundação Eclipse que implementa o protocolo *Message Queuing Telemetry Transport* (MQTT). Esse serviço foi configurado para encaminhar para a nuvem todas as mensagens recebidas. O serviço em nuvem que recebia as mensagens também é da Eclipse. Toda mensagem recebida era acessada pela aplicação MQTT Dash, instalada em um Smartphone.

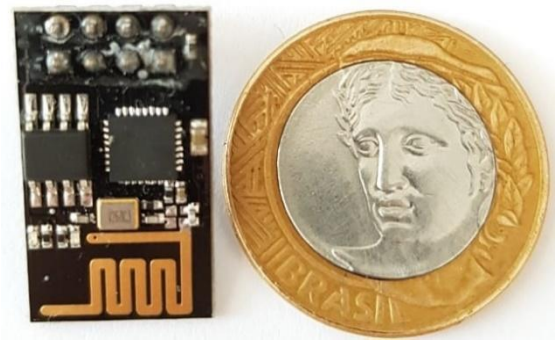
Toda entrada e saída de rede da camada Fog passa pelo IDS, que no cenário proposto pode ser utilizado o Snort ou Suricata. Em suma, o microcontrolador captura as informações do sensor, conecta-se na rede wireless da Fog e envia para o serviço de mensageria as medições. Esse serviço envia para a nuvem, que então tem os dados visualizados em uma aplicação em um Smartphone. A comunicação entre o nodo IoT e a Fog, e do serviço da Fog com a nuvem é monitorada pela aplicação IDS.

O cenário de ataque descrito nos passos 6 e 7 descreve um atacante que obteve acesso não autorizado ao nodo Fog está buscando sobrecarregar o serviço de mensageria. Um computador notebook foi utilizado para esse propósito.

4.1 PROJETO DA IOT

O microcontrolador ESP8266 modelo ESP-01 foi utilizado como recurso de IoT responsável por coletar e transmitir os dados de sensores. Esse microcontrolador se mostrou adequado para o experimento por possuir possibilidade de conexão Wireless 802.11, também já disponível no Raspberry Pi e por ter à disposição uma gama de bibliotecas que atendem todas as demais necessidades do experimento. A Figura 9 retrata o microcontrolador utilizado.

Figura 9 - ESP8266 modelo ESP-01 ao lado de moeda para referência de tamanho.



Fonte: O autor (2019).

Esse microcontrolador não possui nativamente a possibilidade de ser diretamente programado, geralmente ele é utilizado em conjunto com outro microcontrolador, fornecendo somente a possibilidade de acesso à rede Wi-Fi através de comandos AT. Por isso, para que ele pudesse ser programado, foi necessário o carregamento de um *firmware* de outro microcontrolador, do *NodeMCU*. Para esse procedimento, foi necessário um adaptador para conectar o ESP8266 à um computador através de uma interface USB, além de modificá-lo de modo a criar uma conexão entre os pinos *GROUND* e *GPIO-0* para que microcontrolador possa aceitar a escrita de um novo firmware. O adaptador modificado utilizado está ilustrado na Figura 10.

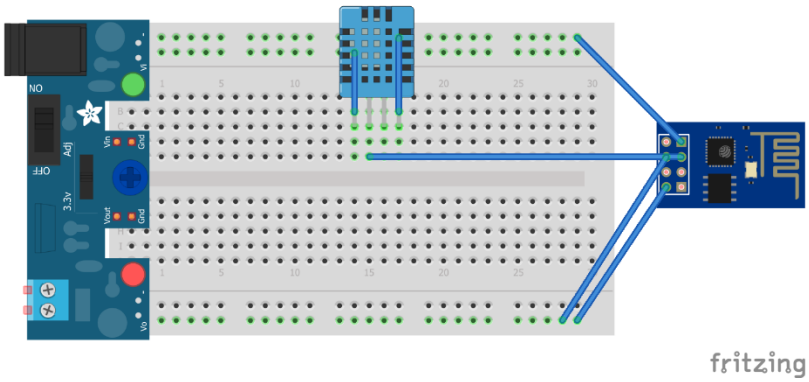
Figura 10 - Adaptador para ESP8266 ESP-01 modificado.

Fonte: O autor (2019).

A carga do firmware foi realizada com o auxílio do software *NodeMCU PyFlasher*, disponível na página do projeto *NodeMCU* junto do *firmware*.

O sensor utilizado em conjunto com o microcontrolador ESP8266 foi o DHT-11. Esse sensor tem acurácia de 5% para umidade e aproximadamente 2 graus centígrados para medições de temperatura de 0 até 50 graus centígrados, com capacidade de aferir as medições a cada segundo. A conexão entre o sensor e o microcontrolador foi facilitada através de uma placa para desenvolvimento de projetos de IoT, chamada Protoboard. O esquema de conexão do projeto é ilustrado na Figura 11.

Figura 11 - Projeto para conexão entre sensor e microcontrolador.

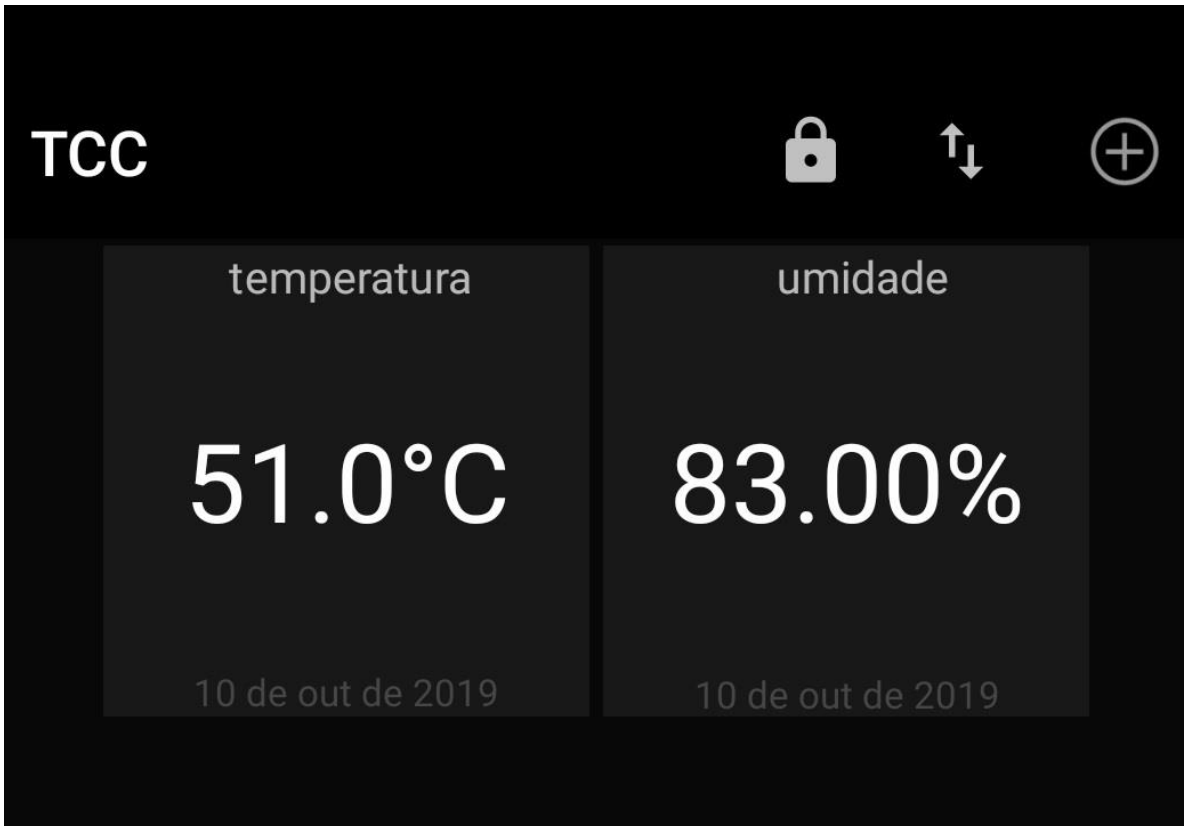


Fonte: O autor (2019).

O desenvolvimento de código para o microcontrolador foi realizado no conjunto de ferramentas do Arduino. Foram necessárias as bibliotecas do sensor, de WiFi e do protocolo de mensagens.

A aplicação MQTT Dash foi instalada em um Smartphone para a visualização dos dados capturados e transmitidos pelo microcontrolador. A aplicação conecta-se com a nuvem, no serviço de mensageria da fundação Eclipse disponível no endereço *mqtt.eclipse.org*.

Figura 12 - Tela da aplicação MQTT Dash.



Fonte: O autor (2019).

4.2 PROJETO DA CAMADA FOG

Um Raspberry Pi modelo B+ foi utilizado para ser o nó da camada Fog. No experimento, esse nó foi responsável por prover conexão de rede e serviço de mensageria. É para esse serviço que o equipamento da IoT enviava as medições capturadas. As medições

então foram encaminhadas para o serviço de Nuvem através do qual um usuário do sistema poderia conectar-se pela internet e exibir as coletas de dados em um painel de monitoramento. Toda a comunicação de rede entre os nós da IoT e a Fog foram monitoradas pelo recurso de IDS.

Para preparo do Raspberry Pi, foi necessário inicialmente carregar um sistema operacional para um cartão micro SD. O sistema escolhido foi o Raspbian, da própria fundação por trás do projeto do Raspberry. Essa escolha foi realizada por conta do amplo suporte da comunidade ao sistema, com diversas aplicações e ferramentas de desenvolvimento disponíveis.

Para disponibilizar as aplicações e serviços nesse nó, um recurso de contêineres foi utilizado. Esse recurso, provido pela aplicação Docker, permite que sejam criados ambientes isolados para que aplicações e serviços sejam disponibilizados. Esses aplicativos têm somente acesso aos recursos os quais necessitam para operação. Essa forma de implantação além de privilegiar a segurança por haver um isolamento entre as aplicações, permite um desenvolvimento rápido e facilitado sem, no entanto, adicionar sobrecarga na operação das aplicações.

Foram criados três contêineres para o experimento: um contêiner para prover acesso à rede, através de um *access point* pela interface Wireless do Raspberry Pi, um contêiner com o serviço de mensageria pronto para receber mensagens no protocolo MQTT e as encaminha-las para a nuvem e um serviço de IDS que monitorava toda a comunicação. A comunicação do Raspberry Pi com a internet se deu por meio da interface de rede Ethernet, que foi conectada a uma rede com acesso à internet.

4.3 CENÁRIO DE TESTES

Um sensor semelhante ao DHT11 utilizado no estudo é capaz de capturar informações de temperatura e umidade a cada segundo. Assume-se que em um cenário residencial, até 25 sensores poderão estar operando e enviando os dados capturados para o nó da Fog. Assim, uma regra específica para esse cenário foi escrita para ambos os sistemas de IDS avaliados para que emitam um alerta quando detectada comunicação anômala na porta em que opera o serviço de mensageria, isso é, quando a quantidade de mensagens recebidas em um único segundo exceda 25.

No cenário proposto, um atacante obtém acesso não autorizado à rede Wireless em que opera o serviço de mensageria da IoT. Esse atacante efetuará o envio de sucessivas mensagens para o serviço de mensageria, de modo a congestioná-lo, como em um ataque de negação de serviço. Para tal empreendimento, será utilizada a aplicação MQTT Malaria, ferramenta utilizada para validação de serviços do protocolo MQTT, para aferir capacidade de recebimento de mensagens entre outros recursos.

Enquanto ocorrerem os ataques, o consumo de recursos do nó da Fog será medido com o uso da aplicação Sar. Serão medidos o uso de memória RAM e CPU.

Assim, será avaliado se ambas as soluções de IDS são capazes de operar em conjunto com um serviço de IoT no nó da Fog, sem causar exaustão de recursos computacionais e se serão capazes de identificar e notificar a sobrecarga causada pelo atacante.

5 RESULTADOS

Com o microcontrolador conectado ao nó Fog pela Wireless, enviado dados do sensor e estes sendo visualizados pelo Smartphone, iniciou-se o processo de ataque. O conjunto de regras de cada IDS foi obtido através das ferramentas de gerenciamento de regras próprias de cada um. Uma regra, nesse contexto, consiste de uma assinatura de ataque conhecido. A característica distintiva de cada ataque pode ser baseada em *offsets* fixos, *strings* ou informações de depuração presentes no pacote de rede. Além da regra criada especificamente para o cenário de teste, foram utilizadas outras regras capazes de detectar os mais variados tipos de eventos de segurança, como ataques de SQL Injection, exploits e vírus, entre outros.

Para o Snort, foi utilizada a ferramenta PulledPork, que consiste de um utilitário que obtém do site da Cisco um amplo conjunto de regras fornecidas através da colaboração de organizações e usuários, que ao identificarem novas ameaças as descrevem na sintaxe da aplicação IDS esses eventos para que sejam detectados. Nesse ponto já se observou a primeira limitação do Snort: a aplicação não iniciou com sucesso quando configurada para iniciar com todo o conjunto de regras disponível, apresentando uma falha e interrompendo abruptamente sua execução. O conjunto de regras que causou o travamento totaliza 12702 regras ativas.

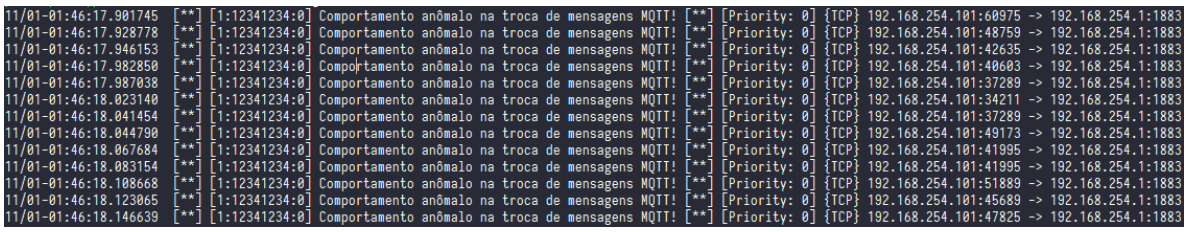
Em face dessa limitação, as regras foram separadas em arquivos distintos pela ferramenta PulledPork. Assim, foram criados três conjuntos de regras: um conjunto de uma regra, apenas para validar a capacidade das ferramentas em teste de detectar uma possível violação de segurança, especificamente um ataque de negação de serviço, um conjunto com 5 mil regras e outro conjunto com 10 mil regras.

Para a carga de regras no Suricata foi utilizada a ferramenta Suricata-update. Essa ferramenta assemelha-se no modo de operar da PulledPork. Ao total, foram transferidas 19.874 regras para o Suricata. Em relação a carga de regras, o Suricata não apresentou problemas de operação no Raspberry Pi 3 B+ para carregar todas as regras. Entretanto, para fins de comparação entre as duas soluções de IDS, a mesma abordagem de separação de regras foi adotada.

A partir de um computador que se conectou à mesma rede do nó da Fog, foi iniciada a transmissão de milhares mensagens para serviço de mensageria, com intuito de simular uma sobrecarga causada por uma tentativa de negação de serviços. Para o empreendimento da sobrecarga, a ferramenta MQTT-Malaria foi utilizada. Com ela, foram simulados o envio de mensagens de 25 clientes enviando 25 mensagens por segundo cada, 50 clientes enviando 50 mensagens por segundo cada e 100 clientes enviando 100 mensagens por segundo cada. Cada bateria de testes durou 1 minuto. Durante esse período os dados de uso de memória e processamento de ambas as soluções de IDS foram aferidos pela aplicação Sar. As três baterias de testes foram executadas em ambos os IDS, com 1 regra, 5 mil regras e 10 mil regras respectivamente. Também foram executados esses mesmos testes no nó da Fog sem que houvesse qualquer IDS ativo operando, para fins de determinar o impacto deles na operação.

Ambas as soluções detectaram com sucesso a sobrecarga que vinha sendo imposta ao sistema quando carregada a regra especificamente para esse propósito. O serviço de mensageria que coleta os dados do sensor de umidade e temperatura não cessou a operação em nenhum dos testes realizados, enviando com sucesso todos os dados coletados. A Figura 12 exibe as mensagens de alertas exibidas pelo Snort ao identificar a sobrecarga do serviço de mensageria.

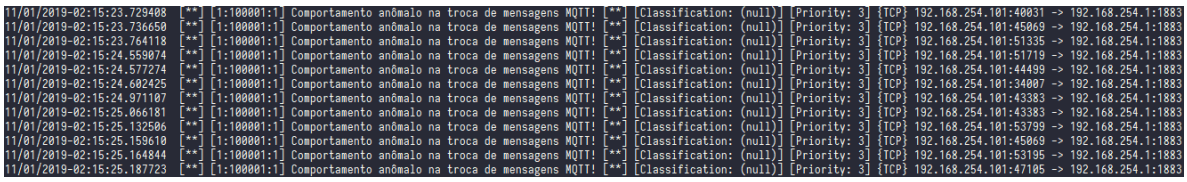
Figura 13 - Mensagens da regra de segurança de MQTT no Snort.



Fonte: O autor (2019).

A mesma regra que gerou os alertas no Snort quando iniciaram os envios das mensagens durantes os testes também foi aplicada para o Suricata, que também identificou com sucesso o ataque, conforme ilustra a Figura 13.

Figura 14 - Mensagens da regra de segurança de MQTT no Suricata.



Fonte: O autor (2019).

A Tabela 3 exibe o compilado de resultados obtidos no experimento, agrupando os testes realizados sem a operação de um IDS durante os ataques, com a operação do Snort com diferentes conjuntos de regras e também do Suricata com diferentes conjuntos de regras. Foram identificados a quantidade de clientes que foram simulados nos ataques, a quantidade de regras carregadas, o percentual da média de consumo de CPU de todos os núcleos de processamento do nó da Fog durante o minuto de coleta de dados do ataque e também o percentual da média de utilização da memória RAM.

Tabela 3 - Resultados do experimento.

IDS	CLIENTES	REGRAS	CPU (%)	RAM (%)
Nenhum	25	0	9,24	7,49
Nenhum	50	0	14,06	7,36
Nenhum	100	0	15,31	7,34
Snort	25	1	8,45	10,59
Snort	50	1	15,81	10,63
Snort	100	1	12,67	10,72
Snort	25	5000	6,8	39,91
Snort	50	5000	16,38	39,9
Snort	100	5000	10,82	39,91
Snort	25	10000	5,65	61,23
Snort	50	10000	8,17	61,22
Snort	100	10000	12,53	61,23
Suricata	25	1	8,79	12,33
Suricata	50	1	12,45	12,25
Suricata	100	1	12,99	12,21
Suricata	25	5000	14,95	18,13
Suricata	50	5000	19,77	18,24
Suricata	100	5000	13,68	18,15
Suricata	25	10000	10,39	24,17
Suricata	50	10000	13,49	24,13
Suricata	100	10000	23,44	24,14

Fonte: O autor (2019).

Analisando os resultados obtidos, observou-se que as soluções de IDS tiveram pouquíssimo impacto na operação do nó da Fog no experimento nos parâmetros propostos. A operação do serviço de mensageria não foi negativamente afetada pela presença de ambas as soluções, mantendo números aproximados na média de uso da CPU. Quanto a memória RAM, observou uma leve progressão no uso pela solução Suricata, variando de 6% a cada incremento de regras. A solução Suricata, entretanto, registrou um uso mais intenso de RAM, saltando de aproximadamente 10% com apenas 1 regras para 39% com 5 mil regras e 61% com 10 mil regras.

6 CONCLUSÃO E TRABALHOS FUTUROS

Esse trabalho buscou investigar a aplicabilidade das duas principais soluções de IDS disponíveis no mercado, chegando conclusão de que ambas são viáveis para o projeto de segurança de IoT através da Fog. Nos experimentos realizados, a ferramenta de melhor desempenho para os parâmetros estabelecidos de teste foi a Suricata, com substancial menor impacto no consumo de memória RAM quando comparada com a ferramenta Snort. O Suricata também não apresentou qualquer problema na carga de grandes volumes de regras, enquanto o Snort falhou na inicialização quando configurado para iniciar com todas as regras que estavam disponíveis. Nenhuma das ferramentas de IDS testadas causou impacto negativo no uso de recursos a ponto de causar qualquer indisponibilidade ou sobrecarga. Para trabalhos futuros, podem ser citados:

- análise comparativa de desempenho de ambas as soluções de IDS para diferentes tipos de ataques;
- comportamento de ambos os IDS para diferentes tipos de operação, como operação inline para IPS;
- efetividade na detecção de ataques para diferentes protocolos de comunicação de IoT.

REFERÊNCIAS

AAZAM, M.; ZEADALLY, S.; A. HARRAS, K. Fog Computing Architecture, Evaluation, and Future Research Directions. **IEEE Communications Magazine**, v. 56, p. 46-52, maio 2018.

ALBIN, E. **A Comparative Analysis of the Snort and Suricata Intrusion-Detection Systems**. [S.l.]: [s.n.]. 2011.

ALBIN, E.; ROWE, N. **A Realistic Experimental Comparison of the Suricata and Snort Intrusion-Detection Systems**. [S.l.]: [s.n.]. mar. 2012. p. 122-127.

AL-FUQAHA, A. et al. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. **IEEE Communications Surveys Tutorials**, v. 17, p. 2347-2376, 2015. ISSN: 1553-877X.

AL-FUQAHA, A. et al. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. **IEEE Communications Surveys Tutorials**, v. 17, p. 2347-2376, 2015. ISSN: 1553-877X.

ANDREEV, S. et al. Dense Moving Fog for Intelligent IoT: Key Challenges and Opportunities. **IEEE Communications Magazine**, v. 57, p. 34-41, maio 2019. ISSN: 0163-6804.

ANTONAKAKIS, M. et al. **Understanding the Mirai Botnet**. Proceedings of the 26th USENIX Conference on Security Symposium. Berkeley: USENIX Association. 2017. p. 1093-1110.

ATLAM, H.; WALTERS, R.; WILLS, G. Fog Computing and the Internet of Things: A Review. **Big Data and Cognitive Computing**, v. 2, p. 10, abr. 2018. ISSN: 2504-2289.

ATZORI, L.; IERA, A.; MORABITO, G. The Internet of Things: A survey. **Computer Networks**, v. 54, p. 2787-2805, out. 2010. ISSN: 1389-1286.

BISHOP, M. **Introduction to Computer Security**. [S.l.]: Addison-Wesley Professional, 2004. ISBN: 0321247442.

BONOMI, F. et al. **Fog Computing and Its Role in the Internet of Things**. Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing. New York, NY, USA: ACM. 2012. p. 13-16.

DAFTARI, S. et al. **IoTGuard:** Scalable and agile safeguards for Internet of Things. MILCOM 2016 - 2016 IEEE Military Communications Conference. [S.l.]: [s.n.]. nov. 2016. p. 61-66.

DEBAR, H. An introduction to intrusion-detection systems. **Proceedings of Connect**, v. 2000, 2000.

DENNING, D. E. An Intrusion-Detection Model. **IEEE Transactions on Software Engineering**, Piscataway, v. SE-13, p. 222-232, fev. 1987. ISSN: 0098-5589.

EVANS, D. The Internet of Things: How the Next Evolution of the Internet is Changing Everything. **Cisco Internet Business Solutions Group (IBSG)**, v. 1, p. 1-11, jan. 2011.

GHORBANI, A. A.; LU, W.; TAVALLAEE, M. **Network Intrusion Detection and Prevention - Concepts and Techniques**. [S.l.]: Springer, v. 47, 2010. 1-198 p. ISBN: 978-0-387-88770-8.

GOODRICH, M.; TAMASSIA, R. **Introduction to Computer Security**. USA: Addison-Wesley Publishing Company, 2010. ISBN: 0321512944, 9780321512949.

GUBBI, J. et al. Internet of Things (IoT): A vision, architectural elements, and future directions. **Future Generation Computer Systems**, v. 29, p. 1645-1660, set. 2013. ISSN: 0167-739X.

HOCK, F.; KORTIš, P. **Commercial and open-source based Intrusion Detection System and Intrusion Prevention System (IDS/IPS) design for an IP networks**. [S.l.]: [s.n.]. nov. 2015. p. 1-4.

HOSSAIN, M. M.; FOTOUHI, M.; HASAN, R. **Towards an Analysis of Security Issues, Challenges, and Open Problems in the Internet of Things**. 2015 IEEE World Congress on Services. [S.l.]: IEEE. jun. 2015. p. 21-28.

IORGA, M. et al. **Fog Computing Conceptual Model**. [S.l.]: National Institute of Standards and Technology. mar. 2018.

LEE, K. et al. **On security and privacy issues of fog computing supported Internet of Things environment**. 2015 6th International Conference on the Network of the Future (NOF). [S.l.]: IEEE. set. 2015. p. 1-3.

MARÍN-TORDERA, E. et al. Do we all really know what a fog node is? Current trends towards an open definition. **Computer Communications**, v. 109, p. 117-130, 2017. ISSN: 0140-3664.

MIORANDI, D. et al. Internet of things: Vision, applications and research challenges. **Ad Hoc Networks**, v. 10, p. 1497-1516, set. 2012. ISSN: 1570-8705.

MOGHADDAM, F. F. et al. **Cloud computing:** Vision, architecture and Characteristics. 2015 IEEE 6th Control and System Graduate Research Colloquium (ICSGRC). [S.l.]: IEEE. ago. 2015. p. 1-6.

MUNIR, A.; KANSAKAR, P.; KHAN, S. U. IFCIoT: Integrated Fog Cloud IoT: A novel architectural paradigm for the future Internet of Things. **IEEE Consumer Electronics Magazine**, v. 6, p. 74-82, jul. 2017. ISSN: 2162-2248.

NI, J. et al. Securing Fog Computing for Internet of Things Applications: Challenges and Solutions. **IEEE Communications Surveys Tutorials**, v. 20, p. 601-628, 2018. ISSN: 1553-877X.

NI, J. et al. Securing Fog Computing for Internet of Things Applications: Challenges and Solutions. **IEEE Communications Surveys Tutorials**, v. 20, p. 601-628, 2018. ISSN: 1553-877X.

NIELES, M.; DEMPSEY, K.; PILLITTERI, V. Y. **An introduction to information security**. [S.l.]. 2017.

PARK, W.; AHN, S. Performance Comparison and Detection Analysis in Snort and Suricata Environment. **Wireless Personal Communications**, v. 94, p. 241-252, 01 fev. 2016. ISSN: 1572-834X.

PATIDAR, S.; RANE, D.; JAIN, P. **A Survey Paper on Cloud Computing**. 2012 Second International Conference on Advanced Computing Communication Technologies. [S.l.]: IEEE. jan. 2012. p. 394-398.

PRAJAPATI, A. G.; SHARMA, S. J.; BADGUJAR, V. S. **All About Cloud:** A Systematic Survey. 2018 International Conference on Smart City and Emerging Technology (ICSCET). [S.l.]: IEEE. jan. 2018. p. 1-6.

PULIAFITO, C. et al. Fog Computing for the Internet of Things: A Survey. **ACM Transactions on Internet Technology**, New York, NY, USA, v. 19, p. 18:1--18:41, abr. 2019. ISSN: 1533-5399.

ROMAN, R.; LOPEZ, J.; MAMBO, M. Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges. **Future Generation Computer Systems**, v. 78, p. 680-698, jan. 2018. ISSN: 0167-739X.

SCARFONE, K. A.; MELL, P. M. **SP 800-94. Guide to Intrusion Detection and Prevention Systems (IDPS)**. Gaithersburg, MD, United States. 2007.

SHAW, S. B.; SINGH, A. K. **A survey on cloud computing**. 2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE). [S.l.]: IEEE. mar. 2014. p. 1-6.

SOHAL, A. S. et al. A cybersecurity framework to identify malicious edge device in fog computing and cloud-of-things environments. **Computers & Security**, v. 74, p. 340-354, 2018. ISSN: 0167-4048.

SOUZA, C. A. D. **Método híbrido de detecção de intrusão aplicando inteligência artificial**. [S.l.]. 2018. Centro de Engenharias e Ciências Exatas.

TANENBAUM, A. S.; WETHERALL, D. J. **Computer Networks**. 5th. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2010. ISBN: 0132126958, 9780132126953.

WANG, Y. et al. A fog-based privacy-preserving approach for distributed signature-based intrusion detection. **Journal of Parallel and Distributed Computing**, v. 122, p. 26-35, 2018. ISSN: 0743-7315.

WHITE, J. S.; FITZSIMMONS, T.; MATTHEWS, J. N. **Quantitative analysis of intrusion detection systems:** Snort and Suricata. [S.l.]: SPIE, v. 8757, 2013.

YOUSEFPOUR, A.; ISHIGAKI, G.; JUE, J. P. **Fog Computing:** Towards Minimizing Delay in the Internet of Things. 2017 IEEE International Conference on Edge Computing (EDGE). [S.l.]: IEEE. jun. 2017. p. 17-24.

ZHOU, L.; GUO, H.; DENG, G. A fog computing based approach to DDoS mitigation in IIoT systems. **Computers & Security**, v. 85, p. 51-62, 2019. ISSN: 0167-4048.

ZITTA, T. et al. **Penetration Testing of Intrusion Detection and Prevention System in Low-Performance Embedded IoT Device**. 2018 18th International Conference on Mechatronics - Mechatronika (ME). [S.l.]: [s.n.]. dez. 2018. p. 1-5.

ZITTA, T.; NERUDA, M.; VOJTECH, L. **The security of RFID readers with IDS/IPS solution using Raspberry Pi**. 2017 18th International Carpathian Control Conference (ICCC). [S.l.]: [s.n.]. maio 2017. p. 316-320.

APÊNDICES

APÊNDICE A – ARTIGO NO FORMATO SBC

Abordagem de detecção de intrusão em Fog para IoT

Bruno Ribeiro da Silva

Centro Tecnológico – Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)

brunoribeiro@grad.ufsc.br

Abstract: The quantity of IoT equipment have considerably grown is the last years, to a point that the communication between machines in the internet surpass the communication man-machine. This new concept brings new challenges, primarily to the information security area, given its heterogenous nature and new architectures that support this paradigm. The fog computing has been utilized as an intermediate layer to give more strength to IoT applications, through supply of storage, processing and services. By having this centralizing characteristic, this work aims to evaluate the applicability of Intrusion Detection systems in IoT through the fog layer, utilizing the already traditional tools for computer networks security.

Resumo: A quantidade de equipamentos da IoT tem crescido consideravelmente nos últimos anos, a ponto de que a comunicação na internet entre máquinas supera a comunicação máquina homem. Esse novo conceito traz novos desafios, principalmente para área de segurança da informação, dada a sua natureza heterogênea e novas arquiteturas que que suportam esse paradigma. A computação em nevoeiro vem sendo utilizada como uma camada intermediária para dar mais robustez as aplicações da IoT, através do fornecimento de armazenamento, processamento e serviços. Por ter essa característica centralizadora, esse trabalho tem como por objetivo avaliar a aplicabilidade de sistemas de detecção de intrusão na IoT através da camada de nevoeiro, utilizando de ferramentas já tradicionais para a segurança de redes de computadores.

1 INTRODUÇÃO

A Internet of Things (IoT) é um novo paradigma que define equipamentos inteligentes, tecnologias e serviços que conectam objetos convencionais do nosso dia-a-dia, permitindo que estes possam se comunicar e cooperar. Os avanços tecnológicos dos microchips tornaram possível embutir uma antena em praticamente qualquer coisa, viabilizando a ideia de se ter um computador em qualquer lugar, a qualquer hora. A IoT tem tido um papel importante na proliferação de equipamentos conectados, espera-se que até 2020 tenham-se 212 bilhões de unidades implantadas pelo mundo (AL-FUQAHA, GUIZANI, *et al.*, 2015).

A IoT está presente em diversos segmentos de mercado, tendo impacto tanto em termos econômicos quanto financeiros. Através da IoT, equipamentos são capazes de cooperar para a tomada de decisão. Os equipamentos inteligentes podem interromper o fornecimento de calor, iluminação ou resfriamento em casas inteligentes, após a ocorrência de determinado evento, por exemplo. Em cidades inteligentes, a IoT pode alavancar o uso eficiente de energia, e contribuir para o melhor fluxo das vias através do controle de semáforos e locais de estacionamento. A segurança material e pessoal também pode ser aprimorada com o emprego de sistemas de detecção de fumaça inteligentes. Na saúde, sensores podem ser empregados para monitorar as condições físicas de pessoas, monitorando pressão sanguínea, temperatura do corpo e sinalizando uma equipe médica na necessidade de intervenção.

Essa presença pervasiva da IoT é garantida através do uso de equipamentos que utilizam baterias e que possuem CPU de baixo consumo. Esse baixo consumo, porém, reflete em baixo poder computacional e por esse motivo muitos serviços oferecidos pela IoT recorrem ao apoio da computação em nuvem. Na nuvem ocorrem a agregação e processamento dos dados gerados pela IoT. Apesar dessa arquitetura atender a necessidade de processamento, introduz o problema da latência, isso porque os datas centers dos provedores de serviços em nuvem são centralizados e em muitos casos distantes da origem dos dados.

Entre a camada de nuvem e de IoT existe a camada de nevoeiro, ou Fog computing. Essa camada intermediária traz pra perto dos equipamentos da IoT o processamento e armazenamento, diminuindo a latência e viabilizando novos serviços. Nesse contexto, a nuvem passa a ser utilizada como camada gerencial, que agrega os dados enquanto é no fog que parte ou todo o processamento dos dados ocorre.

A IoT por ser um conceito emergente, ainda não possui consolidados alguns sérios aspectos, como o de segurança. A IoT tem natureza pervasiva, possui dispositivos que capturam informações e atuam no meio físico que estão inseridos. Dados médicos e identificáveis de pessoas transitam pela IoT, assim como dados de sistemas de alarmes, de controles de estoque, de sensores magnéticos de vias púbicas e de outros serviços críticos. Portanto, dados sensíveis dos usuários são coletados, processados, transmitidos e armazenados através dos componentes de computação em névoa e IoT. Por isso, um ambiente IoT sem a devida segurança pode ser utilizado para causar sérios danos as vítimas.

Em setembro de 2016 uma série de ciberataques aos servidores de uma empresa chamada Dyn causaram a indisponibilidade de diversos grandes serviços da internet, como Amazon, Netflix e GitHub. Esses ataques ocorreram através de uma botnet chamada Mirai. A botnet Mirai, no ápice da sua atividade, chegou a produzir 600 Gbps de fluxo de dados que eram utilizados em ataques de negação de serviço. Durante o período de sua atividade, cerca de 65.000 novos equipamentos de IoT foram incluídos na botnet em 20 horas, chegando num número entre 200.000 e 300.000. Quase 50% dos equipamentos afetados estavam concentrados no Brasil, Colômbia e Vietnã. Entre os dispositivos da IoT afetados encontravam roteadores, impressoras, câmeras IP e dispositivos de gravação de vídeo (ANTONAKAKIS et al., 2017).

Em redes de computadores convencionais, Sistemas de Detecção de Intrusão (em inglês, Intrusion Detection Systems - IDS) são empregados como recurso de segurança capazes de detectar tentativas de invasões, ações suspeitas e atividades que possam prejudicar a integridade da rede e dos nodos. A pesquisa de segurança em IoT, porém, ainda é bastante recente e por essa razão ainda não existem sistemas de IDS consolidados.

Nesse trabalho, serão conduzidos estudos para a avaliação de ferramentas de IDS tradicionais no contexto da Fog, visto que esta está em posição estratégica e normalmente já age como um gateway para os equipamentos da IoT.

2 CONCEITOS FUNDAMENTAIS

2.1 INTERNET OF THINGS

A Internet of Things (IoT, Internet das Coisas em português) é um paradigma que tem a premissa de computação ubíqua através de coisas e objetos, dotados de sensores que através de esquemas de endereçamento único são capazes de interagir e cooperar para um propósito comum (ATZORI, IERA e MORABITO, 2010).

Não há ainda na literatura, entretanto, um consenso quanto a definição formal de IoT, uma vez que esse paradigma tem muitas visões diferentes, que variam principalmente do fato de que os órgãos reguladores, empresas e pesquisadores focam em diferentes aspectos tecnológicos (ATZORI, IERA e MORABITO, 2010).

A padronização da arquitetura da IoT é necessária para que se crie um ambiente competitivo para as organizações (AL-FUQAHA, GUIZANI, *et al.*, 2015).

A IoT oferece uma excelente oportunidade de investimento para fabricantes de equipamentos e desenvolvedores. Até o fim do ano de 2020 projeta-se que os equipamentos de IoT em operação no mundo chegarão a 212 bilhões de unidades, com potencial de mercado para gerar até 2025 um montante de $6.2 trilhões de dólares (AL-FUQAHA, GUIZANI, *et al.*, 2015).

O potencial oferecido por esse paradigma torna possível o desenvolvimento de um imenso número de aplicações. A IoT tem potencial para facilitar atividades relacionadas ao lar, automatizando rotinas simples, como, abrir um portão ou o acionamento automático do ar condicionado. Além disso, pode automatizar o controle de estoque na indústria, controle de tráfego de vias através de semáforos inteligentes, entre outras atividades. A IoT tem presença importante para as seguintes áreas: transporte e logística, saúde, ambientes inteligentes (casa, escritório, indústria), domínio pessoal e social (ATZORI, IERA e MORABITO, 2010).

2.2 CLOUD COMPUTING

A sugestão do que viria a ser a computação em nuvem foi dada nos anos 1960s por John McCarthy quando foi feita a proposta do conceito de "Utilitário de Computação". (SHAW e SINGH, 2014).

A definição de computação em nuvem, segundo o Instituto Nacional de Padrões e Tecnologia (National Institute of Standards and Technology - NIST) é: "Computação em Nuvem é um modelo que permite acesso pela rede ubíquo, conveniente e sob demanda a um conjunto compartilhado de recursos computacionais configuráveis (p. ex., redes, servidores, armazenamento, aplicações, e serviços) que podem ser rapidamente provisionados e liberados com o mínimo esforço de gerenciamento ou interação com o provedor do serviço". (MOGHADDAM, ROHANI, *et al.*, 2015).

A Computação em Nuvem pode ser implantada em três modelos: Pública, Privada e Híbrida (PRAJAPATI, SHARMA e BADGUJAR, 2018).

- Nuvem Pública: esse tipo de nuvem possibilita o acesso a recursos compartilhados, num formato conhecido como multi-inquilinos. O conjunto de recursos providos nesse modelo adotam o formato de pagamento por uso.
- Nuvem Privada: quando usuários ou organizações precisam de um controle mais rígido sobre seus dados e acessos aos mesmos, esse modelo costuma ser adotado. A nuvem privada é geralmente ofertada também por fornecedores de nuvem pública, mas num molde em que os recursos não são compartilhados.
- Nuvem Híbrida: junção dos dois modelos supracitados, nesse caso os recursos mais sensíveis são mantidos em um ambiente mais controlado, na nuvem privada, enquanto outros recursos de armazenamento e processamento são delegados para a nuvem pública, permitindo um cenário mais seguro e com capacidade de escalabilidade.

Além dos diferentes tipos de implantação possíveis para a computação em nuvem, há também os modelos de serviços ofertados que podem ser: Infraestrutura como Serviço (Infrastructure as a service - IaaS), Plataforma como Serviço (Platform as a service PaaS) e Software como Serviço (Software as a service - SaaS).

- IaaS: o provedor desse modelo fornece aos clientes infraestrutura computacional física ou virtual, de modo que os clientes possam instalar sistemas operacionais, middleware e ferramentas que necessitarem.
- PaaS: na plataforma como serviço, o cliente não possui controle sobre a infraestrutura, já que essa é abstraída, mas tem à disposição um ambiente para desenvolvimento e implantação de soluções de software.
- SaaS: são oferecidos softwares hospedados em nuvem, que não requerem instalação, pouca ou nenhuma configuração. O contratante desse modelo não tem controle sobre a infraestrutura ou plataforma.

2.3 FOG COMPUTING

O termo Fog Computing (em português Computação em Nevoeiro) é geralmente associado aos trabalhos de pesquisa da Cisco. Esse conceito recebe variadas definições da indústria e da academia (MUNIR, KANSAKAR e KHAN, 2017). Entretanto, em 2015 surgiu uma coalisão da indústria e academia que fundou a OpenFog Consortium e que define a Fog Computing como: "Fog Computing é uma arquitetura horizontal a nível de sistema que distribui recursos e serviços de computação, armazenamento, controle e rede em qualquer lugar em continuidade a "Nuvem das Coisas" (MUNIR, KANSAKAR e KHAN, 2017).

Na prática, a Fog computing é uma extensão da nuvem, mais próxima das coisas da IoT. A Fog Computing age como um intermediário entre a nuvem e os dispositivos da IoT, trazendo processamento, armazenamento e serviços de rede para perto dos dispositivos finais. Esses dispositivos da Fog são chamados de nodos do nevoeiro. Qualquer dispositivo com capacidade de processamento, armazenamento e conectividade de rede pode ser um nó na Fog. (ATLAM, WALTERS e WILLS, 2018).

Os dispositivos da IoT estão se tornando essenciais para a permitir a troca de dados entre serviços e eletrônicos, mas eles geralmente não são capazes de processar esses dados e fornece-los a serviços que usem esses dados, por consequência da sua natureza intrínseca de limitação de recursos. Nesse ponto, a Fog Computing pode auxiliar, coexistindo com os equipamentos da IoT e cooperando com a nuvem. (PULIAFITO, MINGOZZI, *et al.*, 2019).

A integração entre a Computação em Nuvem e a IoT permite que dispositivos de baixa capacidade da IoT deleguem atividades complexas para a nuvem, fazendo uso da sua capacidade computacional e de armazenamento. No entanto, a natureza centralizada da nuvem pode ser um problema para serviços que demandem baixa latência, visto que os equipamentos que geram os dados podem estar longe do data center da nuvem (PULIAFITO, MINGOZZI, *et al.*, 2019).

Gerenciar os dados gerados pela IoT é um dos maiores desafios quando são implantados tais sistemas. Sistemas de IoT baseados na nuvem possuem aspectos de serem sistemas geralmente de larga escala, heterogêneos e, dependendo da localização dos dispositivos da IoT e da nuvem, podem sofrer de alta latência. Uma solução para esses desafios é o emprego da Fog Computing para descentralizar as aplicações, o gerenciamento e a análise dos dados gerados pela IoT (IORGA, FELDMAN, *et al.*, 2018).

A Fog é um modelo em camadas que permite acesso ubíquo a recursos de nuvem escaláveis. Esse modelo facilita a implantação de aplicações e serviços distribuídos e de baixa latência. Os nós da Fog podem ser físicos ou virtuais, estar distribuídos em uma arquitetura horizontal (com suporte a isolamento) ou vertical (com suporte a federação) e geralmente estão entre os dispositivos da IoT e os serviços em nuvem. (IORGA, FELDMAN, *et al.*, 2018).

2.3 INTRUSION DETECTION SYSTEM

Segundo o (NIELES, DEMPSEY e PILLITTERI, 2017), a detecção de intrusão é o processo de monitoramento de um sistema computacional ou de rede afim de detectar sinais de possíveis ou iminentes incidentes de violação de políticas de segurança. Em sua grande maioria, os incidentes são atividades maliciosas, mas há casos em que estes eventos são gerados por uso incorreto de recursos. Para (GHORBANI, LU e TAVALLAEE, 2010), a detecção de intrusão é o processo de identificar (e possivelmente responder) a uma atividade maliciosa que tenha como alvo computadores e recursos de rede. Nesse contexto, qualquer software ou hardware que possa monitorar, identificar ou responder a esses eventos é relevante para um Sistema de Detecção de Intrusão (Intrusion Detection System - IDS).

Uma intrusão, de acordo com (GHORBANI, LU e TAVALLAEE, 2010), é definida como uma tentativa de uso indevido ou de tornar indisponível um recurso computacional. Intrusões geralmente ocorrem a partir de uma brecha na segurança e por esse motivo devem ser rapidamente detectadas.

Um IDS é um sistema responsável por detectar e responder a essas intrusões. Esses sistemas podem ser empregados em computadores ou em redes de computadores e analisam toda a atividade para encontrar possíveis ataques a segurança dos recursos monitorados. Quando monitoram um computador, analisam seus processos e o comportamento do usuário com o objetivo de encontrar desvios. Em rede, monitoram toda a atividade de comunicação e podem comparar os pacotes transeuntes com um conjunto de ameaças conhecidas ou aplicar inteligência artificial para categoriza-los.

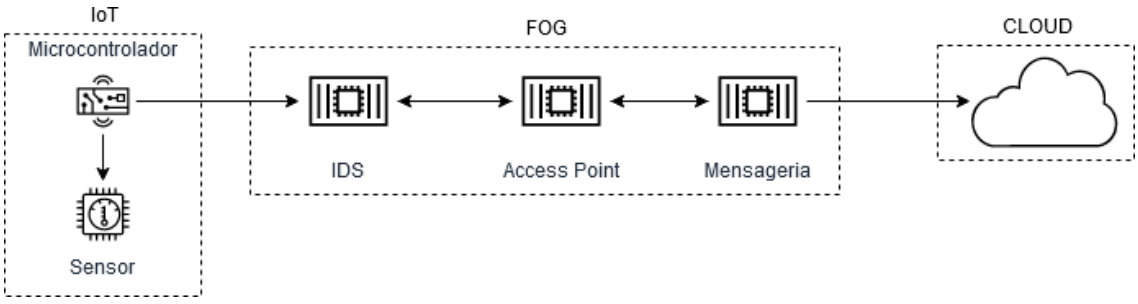
O trabalho de Denning (DENNING, 1987) descreve a automação do processo de detecção através da análise do comportamento de uso de programas. Em seu trabalho, ela sugere que as invasões ocorrem do uso incomum de programas comuns.

Segundo (BISHOP, 2004), existem 4 características importantes para um IDS: detectar uma ampla variedade de intrusões, isto é, ser capaz de detectar incidentes que ocorram dentro e fora do ambiente, sendo também capaz de detectar ataques conhecidos assim como novos ataques. A detecção do evento deve ser oportuna, isso é, nem sempre é possível que seja analisado todo o fluxo de uma rede em tempo real, visto que isso poderia causar um gargalo na rede. Dessa forma, a detecção deve ocorrer tão logo quanto possível. Quanto a informação de que um evento foi verificado, ela deve ser simples e de fácil leitura, já que um IDS pode ser responsável por um elevado número de sistemas, a interface com o usuário é um ponto importante. O último ponto ressaltado trata da acurácia do IDS. Um IDS precisa ser preciso, porque falsos positivos podem ser custosos e fazer com que o sistema fique desacreditado se esses casos forem recorrentes. Já um falso negativo pode ter impacto ainda maior, isso porque incidentes de segurança deixam de ser notificados.

3 AMBIENTE E RESULTADOS EXPERIMENTAIS

A área de segurança para IoT ainda possui muitos desafios e não há ainda métodos consolidados de segurança e principalmente de detecção de intrusão. Nesse trabalho, para analisar a aplicabilidade do Snort ou Suricata como recurso de IDS para a IoT através da Fog, foi realizado um experimento em que ambas as soluções foram instaladas em um equipamento Raspberry Pi. Na estrutura proposta faz o Raspberry Pi faz o papel de recurso computacional da camada Fog, fornecendo segurança, serviço de mensageria para recebimento de dados dos nós da camada IoT e conectividade de rede. Para o serviço de mensageria, foi utilizado o servidor Mosquitto, solução open source fornecida pela fundação Eclipse. Para disponibilizar conectividade aos equipamentos da camada de IoT, um conjunto de softwares foi utilizado para tornar o Raspberry Pi um gateway de acesso à internet, provendo uma conexão com senha, DHCP e regras de direcionamento de conexões para acesso à internet. Todas as mensagens geradas pelos sensores que foram recebidas pelo nó da camada Fog foram direcionadas para um serviço em nuvem da fundação Eclipse. Um software que permite conexão com esse serviço foi instalado em um smartphone Android, para recuperar as mensagens dos sensores através de um painel de visualização. Para testar a efetividade do Raspberry Pi como recurso de IDS, foi utilizado um computador que se conecta através da rede Wireless, simulando um atacante que obteve acesso não autorizado à rede. A Figura 8 ilustra o panorama geral do experimento, onde o microcontrolador obtém dados do sensor e os envia para a Fog, que encaminha para a nuvem. Na Fog, a comunicação passa pelo IDS até chegar no serviço de mensageria.

Figura 8 - Comunicação entre camadas.



Fonte: autor (2019).

3.1 PROJETO DA IOT

O microcontrolador ESP8266 modelo ESP-01 foi utilizado como recurso de IoT responsável por coletar e transmitir os dados de sensores. Esse microcontrolador se mostrou adequado para o experimento por possuir possibilidade de conexão Wireless 802.11, também já disponível no Raspberry Pi e por ter a disposição uma gama de bibliotecas que atendem todas as demais necessidades do experimento. A Figura 9 retrata o microcontrolador utilizado.

Figura 9 - ESP8266 modelo ESP-01 ao lado de moeda para referência de tamanho.

Fonte: O autor (2019).

Esse microcontrolador não possui nativamente a possibilidade de ser diretamente programado, geralmente ele é utilizado em conjunto com outro microcontrolador, fornecendo somente a possibilidade de acesso à rede Wi-Fi através de comandos AT. Por isso, para que ele pudesse ser programado, foi necessário o carregamento de um *firmware* de outro microcontrolador, do *NodeMCU*. Para esse procedimento, foi necessário um adaptador para conectar o ESP8266 à um computador através de uma interface USB, além de modificá-lo de modo a criar uma conexão entre os pinos *GROUND* e *GPIO-0* para que microcontrolador possa aceitar a escrita de um novo firmware. O adaptador modificado utilizado está ilustrado na Figura 10.

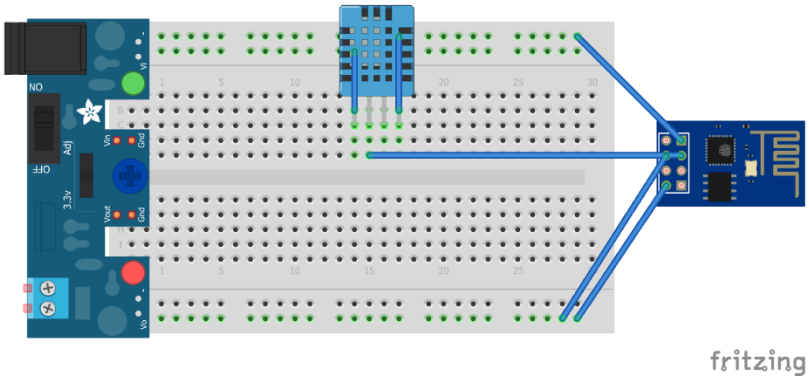
Figura 10 - adaptador para ESP8266 ESP-01 modificado.

Fonte: O autor (2019).

A carga do firmware foi realizada com o auxílio do software *NodeMCU PyFlasher*, disponível na página do projeto *NodeMCU* junto do *firmware*.

O sensor utilizado em conjunto com o microcontrolador ESP8266 foi o DHT-11. Esse sensor tem acurácia de 5% para umidade e aproximadamente 2 graus centígrados para medições de temperatura de 0 até 50 graus centígrados, com capacidade de aferir as medições a cada segundo. A conexão entre o sensor e o microcontrolador foi facilitada através de uma placa para desenvolvimento de projetos de IoT, chamada Protoboard. O esquema de conexão do projeto é ilustrado na Figura 11.

Figura 11 - Projeto para conexão entre sensor e microcontrolador.



Fonte: O autor (2019).

O desenvolvimento de código para o microcontrolador foi realizado no conjunto de ferramentas do Arduino. Foram necessárias as bibliotecas do sensor, de WiFi e do protocolo de mensagens.

A aplicação MQTT Dash foi instalada em um Smartphone para a visualização dos dados capturados e transmitidos pelo microcontrolador. A aplicação conecta-se com a

nuvem, no serviço de mensageria da fundação Eclipse disponível no endereço *mqtt.eclipse.org*.

3.2 PROJETO DA CAMADA FOG

Um Raspberry Pi modelo B+ foi utilizado para ser o nó da camada Fog. No experimento, esse nó foi responsável por prover conexão de rede e serviço de mensageria. É para esse serviço que o equipamento da IoT enviava as medições capturadas. As medições então foram encaminhadas para o serviço de Nuvem através do qual um usuário do sistema poderia conectar-se pela internet e exibir as coletas de dados em um painel de monitoramento. Toda a comunicação de rede entre os nós da IoT e a Fog foram monitoradas pelo recurso de IDS.

Para preparo do Raspberry Pi, foi necessário inicialmente carregar um sistema operacional para um cartão micro SD. O sistema escolhido foi o Raspbian, da própria fundação por trás do projeto do Raspberry. Essa escolha foi realizada por conta do amplo suporte da comunidade ao sistema, com diversas aplicações e ferramentas de desenvolvimento disponíveis.

Para disponibilizar as aplicações e serviços nesse nó, um recurso de contêineres foi utilizado. Esse recurso, provido pela aplicação Docker, permite que sejam criados ambientes isolados para que aplicações e serviços sejam disponibilizados. Esses aplicativos têm somente acesso aos recursos os quais necessitam para operação. Essa forma de implantação além de privilegiar a segurança, dado que o isolamento entre as aplicações, permite um desenvolvimento rápido e facilitado sem, no entanto, adicionar sobrecarga na operação das aplicações.

Foram criados três contêineres para o experimento: um contêiner para prover acesso à rede, através de um *access point* pela interface Wireless do Raspberry Pi, um contêiner com o serviço de mensageria pronto para receber mensagens no protocolo MQTT e as encaminha-las para a nuvem e um serviço de IDS que monitorava toda a comunicação. A comunicação do Raspberry Pi com a internet se deu por meio da interface de rede Ethernet, que foi conectada a uma rede com acesso à internet.

3.3 CENÁRIO DE TESTES

Um sensor semelhante ao DHT11 utilizado no estudo é capaz de capturar informações de temperatura e umidade a cada segundo. Assume-se que em um cenário residencial, até 25 sensores poderão estar operando e enviando os dados capturados para o nó da Fog. Assim, uma regra específica para esse cenário foi escrita para ambos os sistemas de IDS avaliados para que emitam um alerta quando detectada comunicação anômala na porta em que opera o serviço de mensageria, isso é, quando a quantidade de mensagens recebidas em um único segundo exceda 25.

No cenário proposto, um atacante obtém acesso não autorizado à rede Wireless em que opera o serviço de mensageria da IoT. Esse atacante efetuará o envio de sucessivas mensagens para o serviço de mensageria, de modo a congestiona-lo, como em um ataque de negação de serviço. Para tal empreendimento, será utilizada a aplicação MQTT Malaria, ferramenta utilizada para validação de serviços do protocolo MQTT, para aferir capacidade de recebimento de mensagens entre outros recursos.

Enquanto ocorriam os ataques, o consumo de recursos do nó da Fog foi medido com o uso da aplicação Sar. Serão medidos o uso de memória RAM e CPU.

Assim, será avaliado se ambas as soluções de IDS são capazes de operar em conjunto com um serviço de IoT no nó da Fog, sem causar exaustão de recursos computacionais e se serão capazes de identificar e notificar a sobrecarga causada pelo atacante.

3.4 RESULTADOS

Com o microcontrolador conectado ao nó Fog pela Wireless, enviado dados do sensor e estes sendo visualizados pelo Smartphone, iniciou-se o processo de ataque. O conjunto de regras de cada IDS foi obtido através das ferramentas de gerenciamento de regras próprias de cada um. Para o Snort, foi utilizada a ferramenta PulledPork, que consiste de um utilitário que obtém do site da Cisco um amplo conjunto de regras. Nesse ponto já se observou a primeira limitação do Snort: a aplicação não iniciou com sucesso quando configurada para iniciar com todo o conjunto de regras disponível, apresentando uma falha e interrompendo abruptamente sua execução. O conjunto de regras que causou o travamento totaliza 12702

regras ativas. Em face dessa limitação, as regras foram separadas em arquivos distintos pela ferramenta PulledPork. Assim, foram criados três conjuntos de regras: um conjunto de uma regra, apenas para validar a capacidade das ferramentas em teste de detectar uma possível violação de segurança, especificamente um ataque de negação de serviço, um conjunto com 5 mil regras e outro conjunto com 10 mil regras.

Para a carga de regras no Suricata foi utilizada a ferramenta Suricata-update. Essa ferramenta assemelha-se no modo de operar da PulledPork. Ao total, foram transferidas 19.874 regras para o Suricata. Em relação a carga de regras, o Suricata não apresentou problemas de operação no Raspberry Pi 3 B+ para carregar todas as regras. Entretanto, para fins de comparação entre as duas soluções de IDS, a mesma abordagem de separação de regras foi adotada.

A partir de um computador que se conectou à mesma rede do nó da Fog, foi iniciada a transmissão de milhares mensagens para serviço de mensageria, com intuito de simular uma sobrecarga causada por uma tentativa de negação de serviços. Para o empreendimento da sobrecarga, a ferramenta MQTT-Malaria foi utilizada. Com ela, foram simulados o envio de mensagens de 25 clientes enviando 25 mensagens por segundo cada, 50 clientes enviando 50 mensagens por segundo cada e 100 clientes enviando 100 mensagens por segundo cada. Cada bateria de testes durou 1 minuto. Durante esse período os dados de uso de memória e processamento de ambas as soluções de IDS foram aferidos pela aplicação Sar. As três baterias de testes foram executadas em ambos os IDS, com 1 regra, 5 mil regras e 10 mil regras respectivamente. Também foram executados esses mesmos testes no nó da Fog sem que houvesse qualquer IDS ativo operando, para fins de determinar o impacto deles na operação.

Ambas as soluções detectaram com sucesso a sobrecarga que vinha sendo imposta ao sistema quando carregada a regra especificamente para esse propósito. O serviço de mensageria que coleta os dados do sensor de umidade e temperatura não cessou a operação em nenhum dos testes realizados, enviando com sucesso todos os dados coletados.

A Tabela 3 exibe o compilado de resultados obtidos no experimento.

Tabela 3 - Resultados do experimento

IDS	CLIENTES	REGRAS	CPU (%)	RAM (%)
Nenhum	25	0	9,24	7,49
Nenhum	50	0	14,06	7,36
Nenhum	100	0	15,31	7,34
Snort	25	1	8,45	10,59
Snort	50	1	15,81	10,63
Snort	100	1	12,67	10,72
Snort	25	5000	6,8	39,91
Snort	50	5000	16,38	39,9
Snort	100	5000	10,82	39,91
Snort	25	10000	5,65	61,23
Snort	50	10000	8,17	61,22
Snort	100	10000	12,53	61,23
Suricata	25	1	8,79	12,33
Suricata	50	1	12,45	12,25
Suricata	100	1	12,99	12,21
Suricata	25	5000	14,95	18,13
Suricata	50	5000	19,77	18,24
Suricata	100	5000	13,68	18,15
Suricata	25	10000	10,39	24,17
Suricata	50	10000	13,49	24,13
Suricata	100	10000	23,44	24,14

Fonte: O autor (2019).

Analisando os resultados obtidos, observou-se que as soluções de IDS tiveram pouquíssimo impacto na operação do nó da Fog no experimento nos parâmetros propostos. A operação do serviço de mensageria não foi negativamente afetada pela presença de ambas as soluções, mantendo números aproximados na média de uso da CPU. Quanto a memória RAM, observou uma leve progressão no uso pela solução Suricata, variando de 6% a cada incremento de regras. A solução Suricata, entretanto, registrou um uso mais intenso de RAM, saltando de aproximadamente 10% com apenas 1 regras para 39% com 5 mil regras e 61% com 10 mil regras.

4 CONCLUSÃO E TRABALHOS FUTUROS

Esse trabalho buscou investigar a aplicabilidade das duas principais soluções de IDS disponíveis no mercado, chegando conclusão de que ambas são viáveis para o projeto de segurança de IoT através da Fog. Nos experimentos realizados, a ferramenta de melhor desempenho para os parâmetros estabelecidos de teste foi a Suricata, com substancial menor impacto no consumo de memória RAM quando comparada com a ferramenta Snort. O Suricata também não apresentou qualquer problema na carga de grandes volumes de regras, enquanto o Snort falhou na inicialização quando configurado para iniciar com todas as regras que estavam disponíveis. Nenhuma das ferramentas de IDS testadas causou impacto negativo no uso de recursos a ponto de causar qualquer indisponibilidade ou sobrecarga. Para trabalhos futuros, podem ser citados:

- análise comparativa de desempenho de ambas as soluções de IDS para diferentes tipos de ataques;
- comportamento de ambos os IDS para diferentes tipos de operação, como operação inline para IPS;
- efetividade na detecção de ataques para diferentes protocolos de comunicação de IoT.

REFERÊNCIAS

AAZAM, M.; ZEADALLY, S.; A. HARRAS, K. Fog Computing Architecture, Evaluation, and Future Research Directions. **IEEE Communications Magazine**, v. 56, p. 46-52, maio 2018.

ALBIN, E. **A Comparative Analysis of the Snort and Suricata Intrusion-Detection Systems**. [S.l.]: [s.n.]. 2011.

ALBIN, E.; ROWE, N. **A Realistic Experimental Comparison of the Suricata and Snort Intrusion-Detection Systems**. [S.l.]: [s.n.]. mar. 2012. p. 122-127.

AL-FUQAHA, A. et al. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. **IEEE Communications Surveys Tutorials**, v. 17, p. 2347-2376, 2015. ISSN: 1553-877X.

AL-FUQAHA, A. et al. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. **IEEE Communications Surveys Tutorials**, v. 17, p. 2347-2376, 2015. ISSN: 1553-877X.

ANDREEV, S. et al. Dense Moving Fog for Intelligent IoT: Key Challenges and Opportunities. **IEEE Communications Magazine**, v. 57, p. 34-41, maio 2019. ISSN: 0163-6804.

ANTONAKAKIS, M. et al. **Understanding the Mirai Botnet**. Proceedings of the 26th USENIX Conference on Security Symposium. Berkeley: USENIX Association. 2017. p. 1093-1110.

ATLAM, H.; WALTERS, R.; WILLS, G. Fog Computing and the Internet of Things: A Review. **Big Data and Cognitive Computing**, v. 2, p. 10, abr. 2018. ISSN: 2504-2289.

ATZORI, L.; IERA, A.; MORABITO, G. The Internet of Things: A survey. **Computer Networks**, v. 54, p. 2787-2805, out. 2010. ISSN: 1389-1286.

BISHOP, M. **Introduction to Computer Security**. [S.l.]: Addison-Wesley Professional, 2004. ISBN: 0321247442.

BONOMI, F. et al. **Fog Computing and Its Role in the Internet of Things**. Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing. New York, NY, USA: ACM. 2012. p. 13-16.

DAFTARI, S. et al. **IoTGuard:** Scalable and agile safeguards for Internet of Things. MILCOM 2016 - 2016 IEEE Military Communications Conference. [S.l.]: [s.n.]. nov. 2016. p. 61-66.

DEBAR, H. An introduction to intrusion-detection systems. **Proceedings of Connect**, v. 2000, 2000.

DENNING, D. E. An Intrusion-Detection Model. **IEEE Transactions on Software Engineering**, Piscataway, v. SE-13, p. 222-232, fev. 1987. ISSN: 0098-5589.

EVANS, D. The Internet of Things: How the Next Evolution of the Internet is Changing Everything. **Cisco Internet Business Solutions Group (IBSG)**, v. 1, p. 1-11, jan. 2011.

GHORBANI, A. A.; LU, W.; TAVALLAEE, M. **Network Intrusion Detection and Prevention - Concepts and Techniques**. [S.l.]: Springer, v. 47, 2010. 1-198 p. ISBN: 978-0-387-88770-8.

GOODRICH, M.; TAMASSIA, R. **Introduction to Computer Security**. USA: Addison-Wesley Publishing Company, 2010. ISBN: 0321512944, 9780321512949.

GUBBI, J. et al. Internet of Things (IoT): A vision, architectural elements, and future directions. **Future Generation Computer Systems**, v. 29, p. 1645-1660, set. 2013. ISSN: 0167-739X.

HOCK, F.; KORTIš, P. **Commercial and open-source based Intrusion Detection System and Intrusion Prevention System (IDS/IPS) design for an IP networks**. [S.l.]: [s.n.]. nov. 2015. p. 1-4.

HOSSAIN, M. M.; FOTOUHI, M.; HASAN, R. **Towards an Analysis of Security Issues, Challenges, and Open Problems in the Internet of Things**. 2015 IEEE World Congress on Services. [S.l.]: IEEE. jun. 2015. p. 21-28.

IORGA, M. et al. **Fog Computing Conceptual Model**. [S.l.]: National Institute of Standards and Technology. mar. 2018.

LEE, K. et al. **On security and privacy issues of fog computing supported Internet of Things environment**. 2015 6th International Conference on the Network of the Future (NOF). [S.l.]: IEEE. set. 2015. p. 1-3.

MARÍN-TORDERA, E. et al. Do we all really know what a fog node is? Current trends towards an open definition. **Computer Communications**, v. 109, p. 117-130, 2017. ISSN: 0140-3664.

MIORANDI, D. et al. Internet of things: Vision, applications and research challenges. **Ad Hoc Networks**, v. 10, p. 1497-1516, set. 2012. ISSN: 1570-8705.

MOGHADDAM, F. F. et al. **Cloud computing:** Vision, architecture and Characteristics. 2015 IEEE 6th Control and System Graduate Research Colloquium (ICSGRC). [S.l.]: IEEE. ago. 2015. p. 1-6.

MUNIR, A.; KANSAKAR, P.; KHAN, S. U. IFCIoT: Integrated Fog Cloud IoT: A novel architectural paradigm for the future Internet of Things. **IEEE Consumer Electronics Magazine**, v. 6, p. 74-82, jul. 2017. ISSN: 2162-2248.

NI, J. et al. Securing Fog Computing for Internet of Things Applications: Challenges and Solutions. **IEEE Communications Surveys Tutorials**, v. 20, p. 601-628, 2018. ISSN: 1553-877X.

NI, J. et al. Securing Fog Computing for Internet of Things Applications: Challenges and Solutions. **IEEE Communications Surveys Tutorials**, v. 20, p. 601-628, 2018. ISSN: 1553-877X.

NIELES, M.; DEMPSEY, K.; PILLITTERI, V. Y. **An introduction to information security**. [S.l.]. 2017.

PARK, W.; AHN, S. Performance Comparison and Detection Analysis in Snort and Suricata Environment. **Wireless Personal Communications**, v. 94, p. 241-252, 01 fev. 2016. ISSN: 1572-834X.

PATIDAR, S.; RANE, D.; JAIN, P. **A Survey Paper on Cloud Computing**. 2012 Second International Conference on Advanced Computing Communication Technologies. [S.l.]: IEEE. jan. 2012. p. 394-398.

PRAJAPATI, A. G.; SHARMA, S. J.; BADGUJAR, V. S. **All About Cloud:** A Systematic Survey. 2018 International Conference on Smart City and Emerging Technology (ICSCET). [S.l.]: IEEE. jan. 2018. p. 1-6.

PULIAFITO, C. et al. Fog Computing for the Internet of Things: A Survey. **ACM Transactions on Internet Technology**, New York, NY, USA, v. 19, p. 18:1--18:41, abr. 2019. ISSN: 1533-5399.

ROMAN, R.; LOPEZ, J.; MAMBO, M. Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges. **Future Generation Computer Systems**, v. 78, p. 680-698, jan. 2018. ISSN: 0167-739X.

SCARFONE, K. A.; MELL, P. M. **SP 800-94. Guide to Intrusion Detection and Prevention Systems (IDPS)**. Gaithersburg, MD, United States. 2007.

SHAW, S. B.; SINGH, A. K. **A survey on cloud computing**. 2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE). [S.l.]: IEEE. mar. 2014. p. 1-6.

SOHAL, A. S. et al. A cybersecurity framework to identify malicious edge device in fog computing and cloud-of-things environments. **Computers & Security**, v. 74, p. 340-354, 2018. ISSN: 0167-4048.

SOUZA, C. A. D. **Método híbrido de detecção de intrusão aplicando inteligência artificial**. [S.l.]. 2018. Centro de Engenharias e Ciências Exatas.

TANENBAUM, A. S.; WETHERALL, D. J. **Computer Networks**. 5th. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2010. ISBN: 0132126958, 9780132126953.

WANG, Y. et al. A fog-based privacy-preserving approach for distributed signature-based intrusion detection. **Journal of Parallel and Distributed Computing**, v. 122, p. 26-35, 2018. ISSN: 0743-7315.

WHITE, J. S.; FITZSIMMONS, T.; MATTHEWS, J. N. **Quantitative analysis of intrusion detection systems:** Snort and Suricata. [S.l.]: SPIE, v. 8757, 2013.

YOUSEFPOUR, A.; ISHIGAKI, G.; JUE, J. P. **Fog Computing:** Towards Minimizing Delay in the Internet of Things. 2017 IEEE International Conference on Edge Computing (EDGE). [S.l.]: IEEE. jun. 2017. p. 17-24.

ZHOU, L.; GUO, H.; DENG, G. A fog computing based approach to DDoS mitigation in IIoT systems. **Computers & Security**, v. 85, p. 51-62, 2019. ISSN: 0167-4048.

ZITTA, T. et al. **Penetration Testing of Intrusion Detection and Prevention System in Low-Performance Embedded IoT Device**. 2018 18th International Conference on Mechatronics - Mechatronika (ME). [S.l.]: [s.n.]. dez. 2018. p. 1-5.

ZITTA, T.; NERUDA, M.; VOJTECH, L. **The security of RFID readers with IDS/IPS solution using Raspberry Pi**. 2017 18th International Carpathian Control Conference (ICCC). [S.l.]: [s.n.]. maio 2017. p. 316-320.

APÊNDICE B – QUADRO DE SOFTWARES UTILIZADOS

Software	Versão
Alpine	3.10.2
Arduino IDE	1.8.10
DHCPD	4.4.1
Docker	19.03.4
Docker Compose	1.24.1
Fritzing	0.9.3b
Hostapd	2.8
kSar2	0.0.5
Make	4.2.1
Malaria	0.1
Mosquitto	1.6.7
NodeMCU-PyFlasher	4.0
PulledPork	0.7.3
Raspbian	10
Sar	12.0.3
Snort	2.9.13
Suricata	4.0.4
Suricata-Update	1.1.0
Visual Studio Code	1.39.2

APÊNDICE C – QUADRO DE HARDWARES UTILIZADOS

Equipamento	Função
Raspberry Pi 3 Model B+	Nodo Fog de IDS e serviço
ESP8266 ESP-01	Microcontrolador IoT
DHT11	Sensor de umidade e temperatura
Notebook Dell Vostro 5480	Desenvolvimento do trabalho e simulação de ataques

APÊNDICE D – CÓDIGO FONTE DE CONEXÃO DO DISPOSITIVO IOT EM LIGUAGEM ARDUINO

```
#define DHTPIN 2
#define DHTTYPE DHT11

#include <Adafruit_Sensor.h>
#include <ArduinoMqttClient.h>
#include <DHT.h>
#include <DHT_U.h>
#include <ESP8266WiFi.h>

DHT_Unified dht(DHTPIN, DHTTYPE);
uint32_t delayMS;

const char *SSID = "SSID";
const char *PASS = "PASSWORD";

WiFiClient wifiClient;
MqttClient mqttClient(wifiClient);

const char broker[] = "192.168.254.1";
int port = 1883;
const char temperature_topic[] = "tccsinufsc/temperature";
const char humidity_topic[] = "tccsinufsc/humidity";

const long interval = 1000;
unsigned long previousMillis = 0;

void setup() {

  Serial.begin(9600);
  dht.begin();
```

```
  sensor_t sensor;
  dht.temperature().getSensor(&sensor);
  dht.humidity().getSensor(&sensor);
  delayMS = sensor.min_delay / 1000;
  while (!Serial) {
    //
  }

  while (WiFi.begin(SSID, PASS) != WL_CONNECTED) {
    Serial.print(".");
    delay(5000);
  }
  mqttClient.setId("esp01");
  if (!mqttClient.connect(broker, port)) {
    while (1)
      ;
  }
}

void loop() {

  mqttClient.poll();
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    sensors_event_t event;
    dht.temperature().getEvent(&event);

    mqttClient.beginMessage(temperature_topic);
```

```
    if (isnan(event.temperature)) {
      Serial.print("Failed to read temperature.");
    } else {
      mqttClient.print(event.temperature);
      mqttClient.endMessage();
    }
    dht.humidity().getEvent(&event);
    mqttClient.beginMessage(humidity_topic);
    if (isnan(event.relative_humidity)) {
      Serial.print("Failed to read humidity");
    } else {
      mqttClient.print(event.relative_humidity);
      mqttClient.endMessage();
    }
  }
}
```

APÊNDICE E – CÓDIGO FONTE DE ORQUESTRAÇÃO DOS SERVIÇOS DE CONTÊINERES EM LINGUAGEM YAML

```
version: "3"
services:

  access-point:
    container_name: "access-point"
    build:
      context: ./access-point
    cap_add:
      - NET_ADMIN
    restart: unless-stopped
    network_mode: host
    environment:
      - "INTERFACE=wlan0"
      - "CHANNEL=11"
      - "SSID=SSID"
      - "AP_ADDR=192.168.254.1"
      - "SUBNET=192.168.254.0"
      - "WPA_PASSPHRASE=PASSWORD"
      - "OUTGOINGS=eth0"
      - "HW_MODE=g"

  mqtt-bridge:
    container_name: "mqtt-bridge"
    image: "eclipse-mosquitto:latest"
    depends_on:
      - access-point
    restart: unless-stopped
    ports:
      - "1883:1883"
      - "9001:9001"
    volumes:
      - "~/tcc/mqtt-bridge/mosquitto.conf:/mosquitto/config/mosquitto.conf"
      - "~/tcc/mqtt-bridge/log/:/mosquitto/log/"
      - "~/tcc/mqtt-bridge/data/:/mosquitto/data/"

  snort:
    container_name: "snort"
    build:
      context: ./snort
    depends_on:
```

```
    - access-point
  network_mode: host
  command: "-A console -q -N -c /etc/snort/snort.conf -i wlan0"
  volumes:
    - "${PWD}/snort/config/snort.conf:/etc/snort/snort.conf"
    - "${PWD}/snort/config/classification.config:/etc/snort/classification.config"
    - "${PWD}/snort/config/reference.config:/etc/snort/reference.config"
    - "${PWD}/snort/config/gen-msg.map:/etc/snort/gen-msg.map"
    - "${PWD}/snort/config/unicode.map:/etc/snort/unicode.map"
    - "${PWD}/snort/config/threshold.conf:/etc/snort/threshold.conf"
    - "${PWD}/pulledpork/rules/:/var/lib/snort/rules/"
    - "${PWD}/pulledpork/rules/sid-msg.map:/var/lib/snort/etc/sid-msg.map"

suricata:
  container_name: "suricata"
  privileged: true
  build:
    context: ./suricata
  depends_on:
    - access-point
  network_mode: host
  command: "-c /etc/suricata/suricata.yaml -i wlan0"
  volumes:
    - "${PWD}/suricata/config/suricata.yaml:/etc/suricata/suricata.yaml"
    - "${PWD}/suricata/rules:/etc/suricata/rules"

pulledpork:
  container_name: "pulledpork"
  build:
    context: ./pulledpork
```

APÊNDICE F – CÓDIGO FONTE DOS CONTÊINERES AP, SNORT E SURICATA EM LINGUAGEM DOCKERFILE

```
# Snort

FROM alpine:3.10.2
RUN apk --no-cache add bash snort
ADD entrypoint.sh entrypoint.sh
RUN chmod a+x entrypoint.sh
ENTRYPOINT [ "/entrypoint.sh" ]


# Suricata

FROM alpine:3.10.2
RUN apk --no-cache add bash jq python3 suricata && \
    pip3 install --upgrade suricata-update && \
    suricata-update
ADD entrypoint.sh entrypoint.sh
RUN chmod a+x entrypoint.sh
ENTRYPOINT [ "/entrypoint.sh" ]

# AP
FROM alpine:3.10.2
RUN apk --no-cache add dhcp hostapd
ADD entrypoint.sh entrypoint.sh
RUN chmod +x entrypoint.sh
ENTRYPOINT [ "/entrypoint.sh" ]
```