Universidade Estadual de Londrina

LUIS HENRIQUE BARBOSA VIEIRA

APRENDIZADO NÃO SUPERVISIONADO PARA DETECÇÃO DE ATAQUES EM INTERNET DAS COISAS

LONDRINA
2022

LUIS HENRIQUE BARBOSA VIEIRA

APRENDIZADO NÃO SUPERVISIONADO PARA DETECÇÃO DE ATAQUES EM INTERNET DAS COISAS

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Bruno Bogaz Zarpelão

LONDRINA
2022

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UEL

Sobrenome, Nome.
Título do Trabalho : Subtitulo do Trabalho / Nome Sobrenome. - Londrina, 2017.
100 f. : il.

Orientador: Nome do Orientador Sobrenome do Orientador.
Coorientador: Nome Coorientador Sobrenome Coorientador.
Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Programa de Pós-Graduação em Ciência da Computação, 2017.
Inclui bibliografia.

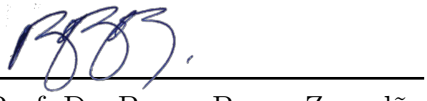
1. Assunto 1 - Tese. 2. Assunto 2 - Tese. 3. Assunto 3 - Tese. 4. Assunto 4 - Tese. I. Sobrenome do Orientador, Nome do Orientador. II. Sobrenome Coorientador, Nome Coorientador. III. Universidade Estadual de Londrina. Centro de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

LUIS HENRIQUE BARBOSA VIEIRA

APRENDIZADO NÃO SUPERVISIONADO PARA DETECÇÃO DE ATAQUES EM INTERNET DAS COISAS

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA



Orientador: Prof. Dr. Bruno Bogaz Zarpelão
Universidade Estadual de Londrina

Prof. Dr. Adilson Luiz Bonifácio
Universidade Estadual de Londrina – UEL

Gabriel Keith Tazima
Universidade Estadual de Londrina – UEL

Londrina, 02 de junho de 2022.

Este trabalho é dedicado ao Criador do universo que dá sentido a toda ciência.

AGRADECIMENTOS

Agradeço a todos os professores do Departamento de Computação da Universidade Estadual de Londrina pela atenção, seriedade e, pelo que considero uma das mais brilhantes características, a humildade. De forma especial, agradeço meu professor orientador Bruno Bogaz Zarpelão pela disponibilidade, paciência e compreensão.

Agradeço aos meus colegas de turma que, por muitas vezes, me ajudaram de forma excepcional no desenvolvimento das atividades da graduação através da troca de conhecimentos.

Agradeço aos meus pais por terem estado comigo em todos os momentos e a todos os meus familiares que sempre me apoiaram. Em especial, agradeço minha irmã Erika pela indicação do curso e pela parceria durante todos estes anos.

Por fim, agradeço principalmente a nosso Deus por, de forma especial ainda durante a graduação, ter me tirado do vazio existencial no qual me encontrava e me colocado novamente no caminho da verdade.

"Não vos amoldeis às estruturas deste mundo, mas transformai-vos pela renovação da mente, a fim de distinguir qual é a vontade de Deus: o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2))

VIEIRA, L. H. B.. **Aprendizado não supervisionado para detecção de ataques em Internet das Coisas**. 2022. 53f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2022.

RESUMO

A crescente quantidade de dispositivos de Internet das Coisas (IoT - *Internet of Things*) desafia a disponibilidade de condições de segurança satisfatórias em redes e sistemas computacionais. A presença destes dispositivos em ambiente domiciliar e urbano cria um cenário a ser explorado por usuários maliciosos que decidem por realizar ataques a sistemas de IoT. Com o aumento destes dispositivos no mercado, mais formas de contornar suas defesas são desenvolvidas, criando assim uma demanda por investigação e análise na área em questão, bem como uma solução que envolva reduzida intervenção humana na detecção destes ataques. Considerando esse cenário, o trabalho tem como objetivo estudar a aplicação de algoritmos não supervisionados como o DBSCAN (*Density-based spatial clustering of applications with noise*) na detecção de ataques realizada a partir da análise dos cabeçalhos dos pacotes coletados na rede, utilizando-se também do teste *Page-Hinkley* para a detecção de mudanças abruptas no comportamento do tráfego de rede. Para tal, será criado um sistema para a análise dos pacotes de rede obtidos de um ambiente IoT, extraindo os campos mais significativos dos cabeçalhos dos pacotes de cada protocolo. Experimentos foram realizados com um conjunto de dados público contendo dados sobre pacotes de rede coletados de equipamentos domésticos. Os resultados mostraram uma alta capacidade de detecção de ataques, mantendo uma baixa taxa de falsos positivos para a maioria dos casos.

Palavras-chave: Inteligência Computacional. Internet das Coisas. Aprendizado de Máquina. Cibersegurança. Detecção de Ataques.

VIEIRA, L. H. B.. **Unsupervised learning for attack detection in the Internet of Things**. 2022. 53p. Final Project (Bachelor of Science in Computer Science) – State University of Londrina, Londrina, 2022.

ABSTRACT

The increasing amount of IoT (*Internet of Things*) devices challenges the availability of an effective security in computer systems and networks. The presence of these devices in the urban and home environment creates a scenario to be exploited by malicious users who decide to perform attacks against IoT systems. More ways of getting around their defenses are developed with the increase of these devices on the market, thus creating a demand for research and analysis in their area, as well as a solution that involves reduced human intervention on the detection of these attacks. Considering this scenario, this work aims to study the use of unsupervised algorithms such as DBSCAN (*Density-based spatial clustering of applications with noise*) to detect attacks based on the analysis of network packet headers, along with the use of the Page-Hinkley test for detecting abrupt changes on the network traffic behavior. For this purpose, a system for analyzing network packets from an IoT environment will be developed, extracting the most significant fields from each protocol's headers. Experiments were carried out with the use of a public dataset containing network packets collected from domestic devices. The results presented a high attack detection capacity, maintaining a low rate of false positives for most cases.

Keywords: Computational Intelligence. Internet of Things. Machine Learning. Cybersecurity. Attack Detection.

LISTA DE ILUSTRAÇÕES

Figura 1 – Estágios da arquitetura de uma solução IoT. Fonte: [1] 26
Figura 2 – Exemplo sobre *directly-density reachable*. Fonte: [2] 31
Figura 3 – Exemplo sobre *density-reachable* e *density-connected*. Fonte: [2] 32
Figura 4 – *Cluster model* do DBSCAN. Fonte: [3] 33
Figura 5 – Exemplo ilustrativo do processo do DBSCAN. Fonte: [4] 33
Figura 6 – Detecção de mudança abrupta em um sinal. Fonte: [5] 34
Figura 7 – Processo de coleta 38
Figura 8 – Exemplo de funcionamento da janela deslizante 39
Figura 9 – Resultado da última execução do DBSCAN antes da ocorrência do primeiro ataque em tráfego TCP no dispositivo TP-Link NC200 42
Figura 10 – Resultado da execução do DBSCAN para a primeira ocorrência de ataque em TCP no dispositivo TP-Link NC200 43
Figura 11 – Exemplo visual do teste para o protocolo UDP no dispositivo Samsung Smart Things Hub 46
Figura 12 – Exemplo visual do teste para o protocolo UDP no dispositivo TP-Link NC200 47

LISTA DE TABELAS

Tabela 1 – Comparação entre *Partitioning* e *Density-based methods* 30
Tabela 2 – Conjunto dos campos selecionados para os protocolos TCP, UDP e ICMP em união aos do protocolo IP selecionados 38
Tabela 3 – Resumo dos melhores resultados para o dispositivo Hive Hub 45
Tabela 4 – Resumo dos melhores resultados para o dispositivo TP-Link NC200 . . 46
Tabela 5 – Resumo dos melhores resultados para o dispositivo Lifx Smart Lamp . 47
Tabela 6 – Resumo dos melhores resultados para o dispositivo TP-Link SmartPlug 48
Tabela 7 – Resumo dos melhores resultados para o dispositivo Samsung Smart Things Hub 48

LISTA DE ABREVIATURAS E SIGLAS

IoT	*Internet of Things*
DBSCAN	*Density-based spatial clustering of applications with noise*
RFID	*Radio Frequency Identification*
TI	Tecnologia da Informação
DDoS	*Distributed Denial of Service*
DoS	*Denial of Service*
OWASP	*Open Web Application Security Project*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hyper Text Transfer Protocol Secure*
API	*Application Programming Interface*
USB	*Universal Serial Bus*
CUSUM	*Cumulative Sum*
PCA	*Principal Component Analysis*
IP	*Internet Protocol*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
ICMP	*Internet Control Message Protocol*
MITM	*Man-in-the-Middle*

SUMÁRIO

1 INTRODUÇÃO 23

2 FUNDAMENTAÇÃO TEÓRICA 25
2.1 Internet of Things 25
2.2 Segurança em Redes IoT 26
2.3 Aprendizado de Máquina 28
2.4 *Clustering* 29
2.4.1 DBSCAN 30
2.5 Detecção de Mudanças 33
2.5.1 Page-Hinkley Test 34

3 DESENVOLVIMENTO DA PROPOSTA 37
3.1 Coleta de pacotes 37
3.2 Clusterização 38
3.3 Detecção 39

4 RESULTADOS 41
4.1 Conjunto de dados de teste 41
4.2 Análise visual do comportamento da execução do DBSCAN . 42
4.3 Testes e análise dos resultados 44

5 CONCLUSÃO 49

REFERÊNCIAS 51

1 INTRODUÇÃO

Dispositivos inteligentes de IoT estão se tornando cada vez mais difundidos nas diversas áreas do espaço urbano e rural. Seja no controle da agricultura e pecuária até a infraestrutura, logística e produção de uma grande corporação ou segurança de um espaço domiciliar, a ideia de integrar o acesso à Internet em dispositivos capazes de oferecer controle remoto ou enviar dados úteis para a observação e criação de relatórios tem se tornado cada vez mais atrativa [6]. Porém, ainda que apresentem um potencial de melhoria aos vários setores nos quais podem ser implementados, estes dispositivos ainda contam, até então, com uma segurança que necessita ser estudada e desenvolvida, o que certamente é um convite para usuários que queiram explorar brechas para atividades maliciosas [7].

Analisando de forma mais cuidadosa a ideia de controlar remotamente ou enviar dados em algumas áreas como a de infraestrutura ou domiciliar, não é difícil encontrar um ponto no qual seja crítica a interrupção do serviço ou captura das informações por agentes não autorizados. Por exemplo, um dispositivo que controle a fechadura de uma casa e registre dados sobre a entrada dos moradores cadastrados, se tiver seu funcionamento interrompido ou o envio de dados interceptado, pode fornecer um acesso indevido à residência ou informações sobre os horários de entrada e saída dos moradores, respectivamente [7]. Tendo isso em vista, fica claro o fato de que a pesquisa por segurança e detecção de ataques em Internet das Coisas torna-se importante, observada a gradativa introdução desta tecnologia no mercado.

Existe a possibilidade de detectar ataques com o uso de aprendizado de máquina supervisionado. No aprendizado supervisionado, um conjunto de dados com amostras de exemplo é apresentado ao algoritmo para treinar um modelo que será utilizado na classificação de dados analisados posteriormente [8]. O aprendizado supervisionado foi utilizado, por exemplo, no trabalho de Belavagi e Muniyal [9], em que a técnica de *Random Forest* é usada obtendo-se uma precisão de 99% na detecção de intrusões. Esta técnica mostra-se eficaz também no trabalho de Stevanovic e Pedersen [10], que a classifica como promissora, precisa e oportuna. Porém, o processo de treinamento nos trabalhos citados foi realizado em computadores com *hardware* superior ao encontrado em dispositivos IoT, tornando muito difícil o aprendizado supervisionado neste caso, visto o grande volume de pacotes de rede. Além disso, temos o problema de que essas técnicas exigem um conjunto de dados de treinamento, no qual classificamos cada pacote como legítimo ou malicioso, tornando o processo de construção deste conjunto de dados trabalhoso e suscetível a erro, sendo difícil também colocar exemplos de todos os tipos de ataques nestes conjuntos de treinamento, o que pode levar o sistema a não detectar alguns deles depois.

Logo, o aprendizado de máquina não supervisionado foi escolhido como uma alter-

nativa para contornar estes problemas, já que esses algoritmos não exigem um treinamento prévio. Nesses algoritmos, os dados são agrupados de acordo com suas similaridades, o que permite identificar diferentes padrões de comportamento e detectar situações de interesse com mais facilidade.

Considerando este contexto, este trabalho apresenta um sistema de detecção de ataques utilizando o algoritmo de aprendizado de máquina não supervisionado DBSCAN (*Density-based spatial clustering of applications with noise*) para a clusterização dos dados juntamente a um algoritmo de detecção de anomalias Page-Hinkley para detectar ataques em redes IoT a partir da análise dos cabeçalhos de pacotes capturados na rede. Preocupa-se também em trabalhar o DBSCAN com reexecuções a cada intervalo de tempo fixo, de forma que a clusterização seja renovada conforme haja a entrada de novos dados, visto que o comportamento dos elementos monitorados muda ao longo do tempo. Ao final, com os resultados obtidos através dos testes, espera-se verificar a eficácia do aprendizado de máquina não supervisionado para a detecção de ataques a redes de IoT.

Os testes serão realizados com a utilização de um conjunto de dados público, contendo pacotes provenientes de uma rede IoT com diversos dispositivos domésticos. O objetivo será simular um ambiente com recebimento de pacotes em tempo real, visando observar o comportamento da solução proposta no trabalho.

Quanto a estrutura do trabalho, o Capítulo 2 é constituído pela fundamentação teórica, que apresenta os conceitos de *Internet of Things*, Segurança em redes IoT, Aprendizado de Máquina, *Clustering* e, por fim, *Change Detection*. No Capítulo 3, a solução proposta pelo trabalho é apresentada por meio dos tópicos de Coleta de Pacotes, *Clustering* e Detecção, representando as três etapas pelas quais o sistema pode ser compreendido. O Capítulo 4 apresenta uma introdução sobre os dados utilizados e a estruturação dos testes, bem como apresenta os resultados obtidos e as análises do comportamento da solução proposta durante e pós teste. Por fim, o Capítulo 5 fornece uma conclusão do autor sobre o trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Internet of Things

O termo Internet of Things foi primeiramente usado por Kevin Ashton em 1999 enquanto trabalhava na Procter & Gamble (P&G) em uma tentativa de atrair a atenção da gerência para uma nova tecnologia de identificação automática por radiofrequência conhecida como RFID (*Radio Frequency Identification*). Segundo o autor em [11], como o tópico Internet estava em alta naquele período, criou o termo em questão junto à palavra "coisas" e, visto que trabalhava no processo de optimização do *supply chain* da empresa, acreditou que isso certamente geraria uma boa impressão nos espectadores de sua apresentação.

Por mais que o termo tenha sido criado em 1999, foi apenas recentemente que a popularidade no seu uso aumentou de forma massiva. O progresso na aplicação e na tentativa de estabelecer uma definição sólida para o IoT acompanhou de perto os avanços na tecnologia de transmissão sem fio e o surgimento cada vez maior dos dispositivos inteligentes no mercado, o que é uma possível explicação para a diferença de tempo entre o surgimento do termo IoT e sua disseminação em trabalhos e discussões [12].

Ainda que o termo seja recente, a ideia de interligar aparelhos é algo que acompanha a área de tecnologia e inovação há muito tempo [13]. Mais especificamente, podemos dizer que a raiz para este desígnio é a própria disseminação da Internet ao público em geral, bem como quando começaram a surgir as primeiras redes de acesso local que interligavam os computadores de uma empresa ou universidade, por exemplo [12].

Assim que o termo se popularizou, várias definições foram surgindo através da interpretação de cada autor. Em meio a sensores que reportam continuamente estatísticas, possibilidade de controle remoto e interconectividade entre aparelhos, torna-se necessária uma definição que compreenda todos estes tipos de dispositivo [13]. Mesmo que o problema com a definição seja real, o que se pode afirmar é que se deseja integrar o mundo real ao virtual [14] [6], explorando cada vez mais a capacidade de criar uma rede de objetos que antes nem mesmo imaginávamos que futuramente poderiam fornecer tal tipo de interação [12].

Assumindo isso, acabamos elevando o espectro do IoT a uma ampla gama de aparelhos e dispositivos que poderiam ser contemplados com esta solução. Para termos uma melhor compreensão de onde tudo isso culminaria, a Figura 1 oferece uma perspectiva sobre a arquitetura separada em estágios de uma solução IoT. Nela, é possível perceber um processo que inicia no que classificamos como as "coisas", que podem ser todos os

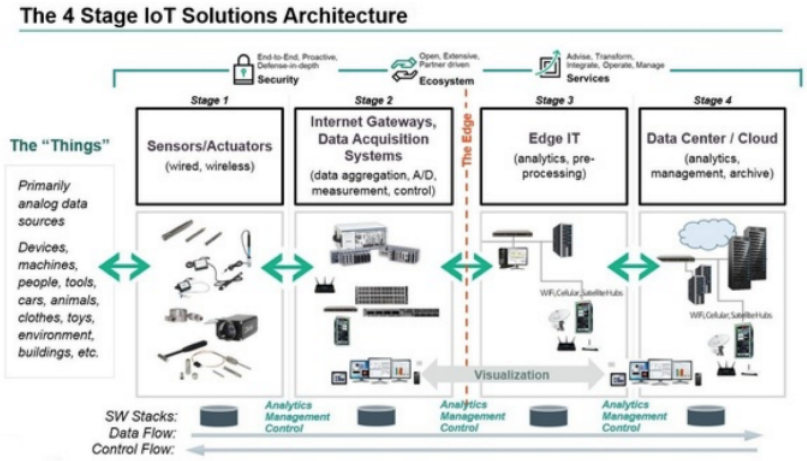


Figura 1 – Estágios da arquitetura de uma solução IoT. Fonte: [1]

dispositivos, máquinas, animais, construções entre outras fontes das quais podemos extrair alguma informação. Para tal, seria necessária a fixação de sensores ou atuadores que transmitam as informações com ou sem fio, dependendo do que se deseja conseguir como dados. Seguindo a sequência, percebe-se que a medida em que percorremos de forma crescente os estágios, desde sensores e atuadores até o estágio final de *Data Center* ou *Cloud*, temos um fluxo de dados que se inicia com a captação da informação do mundo real e termina com a análise, administração e arquivamento destes dados. Em contraste, quando percorremos de forma decrescente, temos um fluxo de controle, pois inicia já no estágio de *Data Center* ou *Cloud* e se encerra nos sensores e atuadores. Isso nos oferece uma visão de que o resultado de uma aplicação IoT, no geral, é dispor de estatísticas sobre as informações coletadas ou poder controlar os dispositivos observados.

2.2 Segurança em Redes IoT

A segurança em dispositivos IoT tem se tornado uma preocupação cada vez mais recorrente. Quando olhamos este cenário de uma forma mais ampla, podemos constatar que o desenvolvimento e evolução das tecnologias utilizadas nos dispositivos IoT não cresceu de forma unificada. Pelo contrário, o processo de crescimento na pesquisa por IoT aconteceu através de diversas contribuições em subáreas diferentes [15].

Porém, a forma na qual a pesquisa e desenvolvimento em IoT evoluiu não é a única adversidade encontrada para a criação de medidas de segurança. A limitação na capacidade do *hardware* encontrado em um dispositivo IoT, quando comparado ao de outro dispositivo de TI convencional, certamente estabelece também uma grande dificuldade, ou até mesmo impossibilidade, em criar uma forma de segurança complexa [16] [17]. Sendo assim, torna-se necessário empregar uma abordagem diferente para solucionar este problema, visto que defesas como antivírus e *firewalls*, que são usadas em redes de TI

comuns, não são uma saída que se encaixa nestas limitações [16].

Consideradas estas razões, Anthi et al. [16] citam que, junto à falta de mecanismos de autenticação e criptografia no transporte, algumas vulnerabilidades surgem e são exploradas através de ataques incluídos nas seguintes classificações:

- ***Denial of Service* (DoS):** Consiste em tentar sobrecarregar o dispositivo IoT com requisições irrelevantes visando inativar os serviços prestados por ele aos usuários temporariamente;
- ***Distributed Denial of Service* (DDoS):** O objetivo é comprometer uma grande quantidade de dispositivos IoT para que sirvam como auxiliadores em um processo DoS, tornando o impacto ainda maior;
- ***Man-In-The-Middle*:** Visa-se neste ataque posicionar-se entre o dispositivo IoT que envia os dados e qualquer outro dispositivo que os receba, atuando assim como uma ponte pela qual os dados passam e servindo-se da capacidade de leitura e modificação do conteúdo sendo transportado;
- ***Spoofing*:** Utilizando-se de uma vulnerabilidade na autenticação da fonte dos dados, o objetivo do ataque é a manipulação de identidades para falsificar o conteúdo enviado e comprometer o funcionamento do dispositivo IoT;
- ***Data Leakage* e *Insecure Firmware*:** Exposição dos dados do usuário e comprometimento no controle do dispositivo IoT devido a interfaces inseguras e falta de criptografia no transporte.

Sobre os tipos de vulnerabilidades, não é uma tarefa fácil apontar de imediato todas as existentes tanto para sistemas IoT como para os convencionais de TI. Visto isso, a OWASP (*Open Web Application Security Project*), uma fundação sem fins lucrativos que visa trabalhar para aprimorar a segurança de *software*, dedicou um de seus projetos inteiramente para a pesquisa sobre as vulnerabilidades presentes em ecossistemas IoT [18]. Tal projeto começou em 2014 visando ajudar desenvolvedores, fabricantes, empresas e consumidores na criação e uso de sistemas IoT. Em 2018 a fundação anunciou a sua versão mais recente do que tem chamado "OWASP IoT Top 10", uma lista com os dez principais casos a serem evitados no desenvolvimento e gerenciamento deste tipo de sistema [18]. São eles:

- ***Weak, Guessable, or Hardcoded Passwords*:** Uso de uma senha padrão no processo de desenvolvimento que possa ser divulgada publicamente e que seja fraca e fácil de ser descoberta através de força bruta. O uso deste tipo de senha padrão em dispositivos pode comprometer uma linha inteira de um produto e seus usuários;

- ***Insecure Network Services*:** Serviços expostos à Internet rodando no dispositivo que sejam desnecessários e inseguros, comunicando-se, por exemplo, através de HTTP ao invés de HTTPS, comprometendo a confidencialidade da informação pela falta de criptografia;

- ***Insecure Ecosystem Interfaces*:** Uso inseguro de interfaces Web, *mobile* e APIs, sendo causas recorrentes disso a falta de autenticação, criptografia e filtro na entrada e saída de dados;

- ***Lack of Secure Update Mechanism*:** Realização das atualizações no dispositivo de forma insegura. Alguns exemplos para este caso são a falta de validação do *firmware*, de criptografia nos dados da atualização, de mecanismo de recuperação e de notificações em caso de mudança no sistema de segurança;

- ***Use of Insecure or Outdated Components*:** Uso de módulos e bibliotecas depreciados ou inseguros, encontrados em customizações do sistema operacional utilizado e componentes de terceiros com histórico de insegurança;

- ***Insufficient Privacy Protection*:** Uso sem permissão ou armazenamento incorreto das informações pessoais do usuário no dispositivo ou ecossistema IoT;

- ***Insecure Data Transfer and Storage*:** Armazenamento ou manipulação em texto (sem criptografia) de dados sensíveis em qualquer lugar ou etapa do ecossistema IoT;

- ***Lack of Device Management*:** Falta de suporte ou monitoramento de um dispositivo já implantado. Pode acarretar a presença de configurações erradas, o que facilitaria uma exploração de privilégios, por exemplo;

- ***Insecure Default Settings*:** Envio do produto na compra ou fabricação já configurado inseguramente ou sem permitir a alteração das configurações pelos operadores;

- ***Lack of Physical Hardening*:** Falta de segurança na parte física do produto, seja na facilidade em obter o componente de armazenamento interno dos dados ou na presença de portas externas USB usadas para acessar o dispositivo em modo de configuração ou manutenção.

2.3 Aprendizado de Máquina

O Aprendizado de Máquina é uma área do estudo sobre Inteligência Artificial que trata do desenvolvimento de algoritmos e sistemas que possibilitem a aquisição de conhecimento automaticamente sobre uma entrada de dados sendo analisada de forma computacional [19] [20]. Ainda que programar uma aplicação para tratar uma entrada

de dados seja algo bem conhecido, existem certos problemas que apresentam elevada dificuldade quanto a tentar resolvê-los com programação manual para todos os tipos de situações que possam surgir [21].

Se levarmos em consideração um cenário no qual os dados são obtidos de um ambiente que possa sofrer mudanças repentinas ou apresentar determinados comportamentos, torna-se árdua, ou até mesmo impossível, a tarefa de preparar um tratamento para todos os casos usando verificações ou testes fixos em código [20]. A solução para tal cenário, portanto, seria um sistema capaz de aprender com as situações passadas e adaptar-se automaticamente às mudanças que possam surgir. Desta forma, seria possível reduzir significativamente a necessidade de intervenção humana, visto que o desenvolvedor não precisaria predizer todos os casos, mas sim trabalharia com os dados coletados do ambiente e não apenas com intuição [20] [21].

Deve-se notar que não existe um algoritmo único que apresente resultado ótimo para todos os tipos de problemas [19]. Logo, é necessário alterar o tipo de abordagem dependendo da situação encontrada. O Aprendizado de Máquina, por utilizar-se de testes estatísticos, requer que uma análise profunda sobre os dados e suas características seja realizada para que, tendo sido escolhido o algoritmo, estes dados alinhem-se corretamente à proposta de solução do teste estatístico utilizado [20]. Uma abordagem correta neste quesito pode beneficiar tanto o desempenho geral de um sistema como no resultado encontrado pela aplicação. Consequentemente, o Aprendizado de Máquina pode ser dividido entre duas principais áreas que delimitam aproximações diferentes quanto a estratégia utilizada na construção de um sistema, sendo elas: aprendizado supervisionado e não supervisionado [19].

No Aprendizado de Máquina supervisionado, como o próprio nome sugere, o processo de aprendizado em si possui um "supervisor" que oferece os valores corretos para que, a partir deles como base, sejam feitas as comparações e inferências futuras. A supervisão, neste caso, geralmente consiste em um conjunto de dados de treino que possui rótulos indicando a resposta correta para exemplos ali incluídos. Já no aprendizado não supervisionado, o objetivo consiste em realizar o aprendizado tendo apenas os dados de entrada, sem qualquer tipo de supervisão por parte de um conjunto de treino, sendo necessário um outro tipo de abordagem ao problema [20].

2.4 *Clustering*

Sabendo que o Aprendizado de Máquina não supervisionado trabalha apenas com os dados de entrada, sem qualquer tipo de rótulo, o método usado neste caso não pode ser o mesmo do aprendizado supervisionado, que necessita de um conjunto base que lhe informe uma rotulação correta. A estratégia do aprendizado não supervisionado, portanto,

consiste em encontrar padrões de similaridade dentre estes dados de entrada [20] [22]. Isso é possível através de uma técnica denominada *clustering* a qual, em sua forma mais básica, consiste em particionar os dados coletados em grupos (*clusters*), procurando ao máximo aumentar a similaridade entre os elementos de um mesmo *cluster* [23] [24] [22].

O modo no qual a clusterização é realizada, porém, não é único. A ausência de uma descrição precisa sobre *cluster*, segundo Rokach e Maimon [25], levou ao desenvolvimento de vários métodos com aproximações diferentes para a resolução do problema de clusterização. Han et al. [26] organiza os métodos de clusterização em quatro categorias principais: *partitioning*, *hierarchical*, *density-based* e *grid-based methods*. Na Tabela 1 é possível observar uma comparação entre as caraterísticas principais do *partitioning* e *density-based methods*, segundo a própria descrição de Han et al [26]. Tendo em vista as características dos dados e do problema a ser descrito no decorrer do trabalho, escolheu-se por detalhar estes dois métodos que são, no geral, os utilizados para clusterização neste tipo de situação, bem como ressaltar suas diferenças.

Tabela 1 – Comparação entre *Partitioning* e *Density-based methods*. (Fonte: [26])

Método	**Características**
Partitiong methods	• Encontra *clusters* mutuamente exclusivos de forma esférica; • *Distance-based*; • Pode usar média ou medóide (etc.) para representar o centro do *cluster*; • Eficaz para conjuntos de dados pequenos e médios.
Density-based methods	• Pode encontrar *clusters* de formato arbitrário; • Clusters são regiões densas de objetos no espaço que são separados por regiões de baixa densidade; • Densidade do cluster: Cada ponto deve ter um número mínimo de pontos dentro de sua “vizinhança”; • Pode filtrar outliers.

2.4.1 DBSCAN

O DBSCAN (*Density-based spatial clustering of applications with noise*), que emprega um método *density-based*, foi inicialmente proposto no trabalho de Ester et al. [2] como um algoritmo de clusterização desenvolvido para encontrar agrupamentos e ruídos em um conjunto de dados espaciais.

Em síntese, os algoritmos de clusterização tem por objetivo primário detectar a similaridade entre elementos de um conjunto de dados e então agrupá-los de uma forma na qual não seja necessário informar previamente quais são estes grupos ou as características que determinam se um elemento deve ser incluído ou não. Segundo Stein e Busch [4] e

tendo em vista este objetivo inicial dos algoritmos de clusterização, a diferença entre os diversos algoritmos desta categoria dá-se pela estratégia adotada para a detecção e formação dos *clusters* com os elementos do conjunto informado, aspirando sempre maximizar a similaridade entre membros de um mesmo *cluster* e a diferença com os de fora.

Dentre as classificações de algoritmos de clusterização, temos os conhecidos por *Density-Based Algorithms*. O conceito principal destes algoritmos é, tendo recebido um valor k referente ao número mínimo de elementos por *cluster* e outro ϵ referente ao alcance máximo de um raio utilizado para determinar se certas instâncias de dados podem ser consideradas vizinhas, encontrar agrupamentos que possuam no mínimo k elementos alcançáveis diretamente, indiretamente ou por conexão no raio ϵ através da distância euclidiana, podendo estes elementos serem classificados entre *directly density-recheable*, *density-recheable* ou *density-connected* entre si.

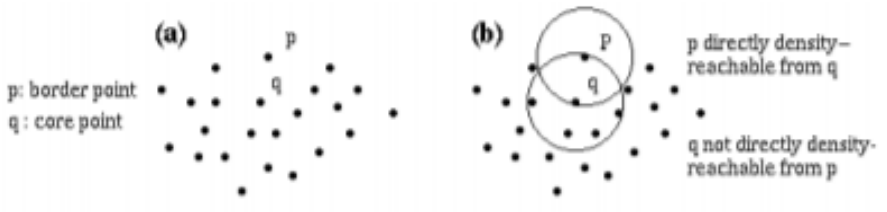


Figura 2 – Exemplo sobre *directly-density reachable*. Fonte: [2]

É possível observar na Figura 2 um exemplo para a definição de um elemento como diretamente alcançável. Tendo em vista que a circunferência maior ao redor dos pontos representa o raio ϵ informado e que os pontos representam os dados de um conjunto de dados, percebe-se que o ponto p é classificado como diretamente alcançável devido ao fato de estar no interior da vizinhança do ponto *core* q, porém o inverso não é verdadeiro, visto que isso causaria assimetria em decorrência da forma na qual os pontos foram marcados (*border* e *core*).

Para que um ponto q seja classificado como *core*, é necessário que a condição k de tamanho mínimo da vizinhança seja satisfeita, com o próprio q fazendo parte desta contagem. Já para a classificação do ponto como *border*, esta condição não precisa ser seguida, visto que existe uma grande possibilidade de que os pontos da fronteira de um *cluster* não tenham uma vizinhança tão numerosa quantos os que estão mais ao interior. Por exemplo, se tivéssemos uma cadeia de pontos que fosse de um ponto *core* q até um ponto fronteiriço p poderíamos afirmar que todos os pontos que levassem até p seriam também *core* por satisfazerem a condição de vizinhança mínima, ou seja, conectariam o ponto q ao p por densidade. Mas o inverso não pode ser afirmado, visto que, como dito anteriormente, os pontos na fronteira são os primeiros a não conseguir satisfazer esta condição de vizinhança. Portanto se tentássemos começar uma cadeia que fosse de p até q, ou seja, o inverso, existe uma grande possibilidade de que já no começo dela em p não

conseguiríamos satisfazer a condição de densidade mínima, e assim nem mesmo teríamos uma cadeia. Esta, então, seria a noção de assimetria.

De acordo com Ester et al. [2], esta diferenciação entre *core* e *border points* como também a definição de assimetria são necessárias visto a menor quantidade de vizinhos encontrados na fronteira de um *cluster*. Logo, se todos os pontos fossem iguais para o algoritmo com apenas o número de vizinhos internos a um raio sendo considerados, haveria casos nos quais os pontos fronteiriços de um *cluster* sempre seriam excluídos.

Na Figura 3, observa-se dois exemplos para *density-reachable* e *density-connected*. No caso (a), nota-se novamente a existência de um ponto q *core* e outro p *border*, desta vez sem possuir qualquer alcance direto entre ambos. Porém, é possível identificar a presença de um alcance indireto entre os pontos citados se levarmos em consideração a direct density-reachability entre q e outro ponto intermediário. O caso (a) ainda apresenta uma situação de assimetria, alerta Ester et al. [2], o que é comparável ao exemplo dado anteriormente, no qual o inverso não é verdadeiro, isto é, de p até q.

Logo, no caso (b) é apresentado um exemplo no qual os pontos p e q são agora considerados *density-connected* em um outro ponto intermediário o pelo qual são ambos *density-reachable*. Percebe-se agora que a relação entre os pontos é simétrica, uma vez que não é vista como uma cadeia de pontos alcançáveis por densidade (o que levaria ao problema discutido anteriormente) mas pela presença de um ponto comum pelo qual são ambos alcançáveis, e isso garante que o inverso de uma relação seja verdadeiro (por exemplo, a relação p e q é *density-connected* e a q e p também).

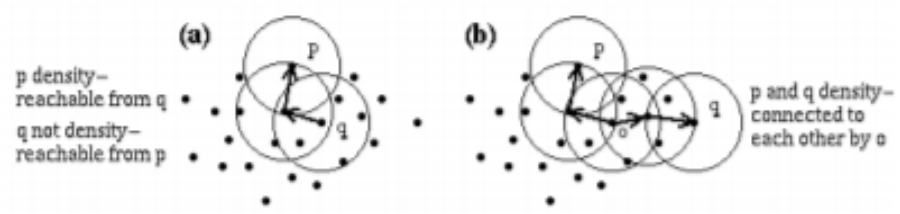


Figura 3 – Exemplo sobre *density-reachable* e *density-connected*. Fonte: [2]

Tendo em mente estas definições, na Figura 4 é possível verificar através do *cluster model* do DBSCAN como os exemplos anteriores se aplicam no caso deste algoritmo. Com as circunferências representando o raio ϵ e as setas representando a *density-reachability*, constata-se que os pontos B e C são *density-connected* devido ao fato de serem ambos *density-reachable* por A. Identifica-se também o ponto N que por não ser *density-reachable* foi classificado como *outlier*.

De uma forma mais ilustrativa, é possível compreender melhor como ocorreria o processo do DBSCAN e o seu resultado através da Figura 5, na qual dois *clusters* são gerados e dois dados do conjunto marcados como *outlier*.

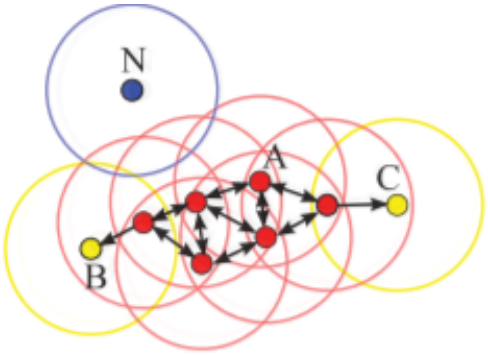


Figura 4 – *Cluster model* do DBSCAN. Fonte: [3]

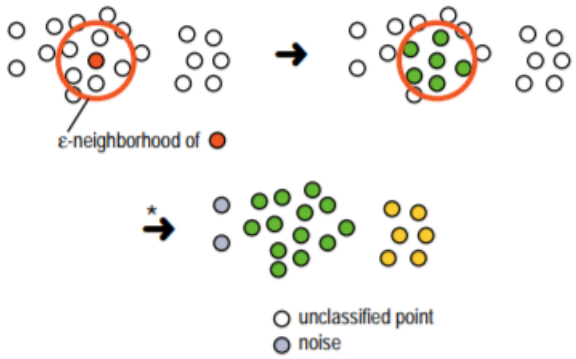


Figura 5 – Exemplo ilustrativo do processo do DBSCAN. Fonte: [4]

2.5 Detecção de Mudanças

Quando consideramos o estudo ou análise de uma série de dados que varia conforme o passar do tempo, logo chegamos a uma conclusão de que a observação dos valores assumidos por esta série pode ser útil para vários tipos de aplicações [27]. Dentre as características que podem ser observadas nestes dados, a que mais atrai o interesse para a construção de aplicações de monitoramento certamente é a presença de uma mudança abrupta nos valores que definem a série. Neste caso, podemos considerar como mudança abrupta a primeira ocorrência de um rápido distanciamento entre o valor atual sendo analisado em relação a um período no qual os valores mantiveram-se estáveis [27].

Considera-se como detecção de uma mudança abrupta, portanto, uma abordagem capaz de realizar a identificação correta e sem intervenção humana dos pontos de grande variação em uma série de dados temporal [27]. A Figura 6 exemplifica o que foi dito, na qual é possível observar os pontos de mudança que foram marcados em uma série que compreende o valor da média de um sinal ao longo do tempo. Nota-se que, assim como foi citado anteriormente, o conceito de mudança não se refere apenas ao aumento no valor quando comparado a um anterior imediato, mas sim na repentina variação quando comparado ao padrão que os valores estavam assumindo anteriormente.

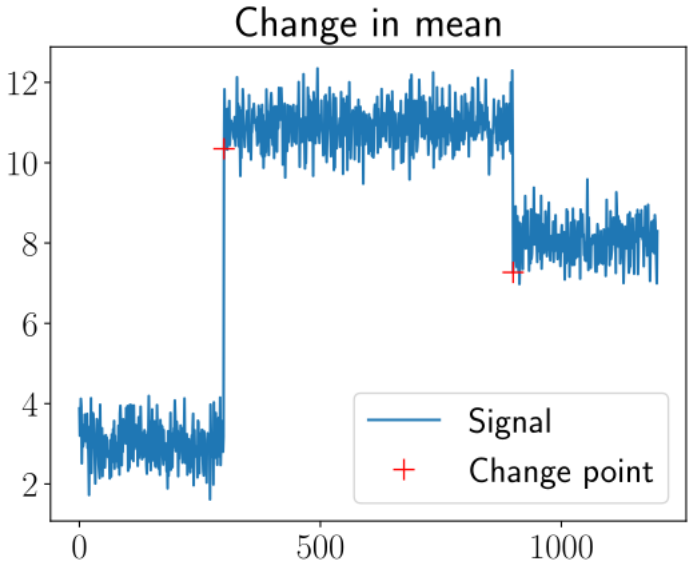


Figura 6 – Detecção de mudança abrupta em um sinal. Fonte: [5]

Não existe, porém, um único modelo para detecção de mudanças. Dependendo da aplicação desejada, um algoritmo de detecção com características diferentes pode ser utilizado. Gama [28] cita que um algoritmo para detecção de mudanças pode ser separado em quatro características principais, que se referem a como o algoritmo armazena a lembrança de exemplos anteriores, a forma na qual a detecção é realizada, o modo no qual se adapta aos exemplos mais recentes e o modelo de decisão utilizado. Cada estratégia tomada para o algoritmo impacta diretamente no resultado obtido, sendo muito provável encontrar abordagens diferentes para a resolução deste mesmo problema de detecção de mudanças dependendo do tipo de dado observado [27].

2.5.1 Page-Hinkley Test

O CUSUM, também citado como *Page-Hinkley Test* [29], é um teste estatístico utilizado para detectar uma mudança abrupta em uma série de dados. No geral, o problema de detectar um ponto de mudança já é um tópico muito discutido e aprofundado, visto sua utilidade em várias áreas [30], tendo o *Page-Hinkley Test* sido proposto como uma técnica para a análise sequencial, na qual o tamanho da amostra não é previamente fixada.

Para o caso do *Page-Hinkley*, vamos considerar uma sequência de valores que serão informados ao teste individualmente, iniciando pelo primeiro elemento. Consideremos também uma variável $\bar{m}$ que representará a média dos valores observados até momento, outra variável s que armazenará a diferença com a média anterior e um valor t que será o número de amostras que foram incluídas no teste até então. Ao início do teste, as variáveis

citadas iniciam com os seguintes valores:

$$s = 0, t = 1, \bar{m} = 0 \tag{2.1}$$

Considerando que um novo valor v seja inserido no teste, calcula-se uma nova média através da divisão entre $v - \bar{m}$ pela contagem de amostras t e adicionando o resultado ao valor de $\bar{m}$. A média é calculada desta forma para que não sejam necessárias as amostras antigas do teste, o que nos levaria a ter que realizar o cálculo através da divisão da soma de todos os valores anteriores.

Tendo calculado a nova média, adquire-se o valor de s através de uma verificação pelo máximo entre 0 e a soma entre $v - \bar{m} - \delta$ e uma multiplicação de um fator de esquecimento α com o valor atual de s, como é possível observar em (2.2). Percebe-se que, ao multiplicar α com o valor anterior de s, α atua como um redutor do impacto das observações anteriores quando assume valores menores que 1, assim como o fator δ reduz o impacto do novo valor v no teste.

$$\begin{aligned} \bar{m} &= \bar{m} + \frac{v - \bar{m}}{t} \\ s &= max(0, \alpha * s + (v - \bar{m} - \delta)) \\ t &= t + 1 \end{aligned} \tag{2.2}$$

Após os cálculos, o teste chega a duas hipóteses, como pode ser visto em (2.3). Se o resultado s for maior que um limiar λ, uma mudança é detectada, como é o caso da hipótese H_1. Nesta situação, as variáveis têm seus valores restaurados iguais aos vistos em (2.1) e o teste é reiniciado para o primeiro valor que venha a ser inserido no teste.

$$\begin{aligned} H_0 &: s < \lambda \\ H_1 &: s \geq \lambda \end{aligned} \tag{2.3}$$

3 DESENVOLVIMENTO DA PROPOSTA

Este trabalho propõe o desenvolvimento de uma solução para detecção de ataques a dispositivos de Internet das Coisas (IoT) com a utilização do algoritmo de Aprendizado de Máquina não supervisionado DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) e do teste de Page-Hinkley.

De forma geral e por ordem de execução, a solução pode ser dividida entre as fases de coleta, *clustering* e detecção. A fase de coleta compreende tanto a captura dos pacotes recebidos pelo dispositivo como também a divisão dos pacotes de acordo com seus protocolos em grupos para a aplicação do DBSCAN. A fase de *clustering* compreende a normalização dos dados obtidos e a execução do DBSCAN para a definição dos *clusters* e *outliers*. Por fim, a fase de detecção compreende uma nova preparação sobre o que foi obtido como resultado do DBSCAN, utilizando-se do método PCA (Principal Component Analysis) para redução da dimensionalidade de cada instância de dados, seguido da utilização do teste de Page-Hinkley para verificar mudanças abruptas na distância entre clusters e outliers ao longo do tempo.

3.1 Coleta de pacotes

O processo, primeiramente, coleta pacotes derivados do tráfego de rede de um dispositivo de IoT durante um período fixo equivalente a 1 minuto. No decorrer deste tempo, todos os pacotes são coletados sem distinção de protocolo. Após o término desta janela de tempo, certos campos do cabeçalho dos pacotes coletados são extraídos, como é possível observar na Tabela 2. Estes campos foram selecionados com base no trabalho de Anthi et al. [16], cujos resultados indicaram que estes campos eram os que mostravam mais claramente a ocorrência de ataques.

Após selecionar estes campos, os pacotes são divididos em três grupos através de uma observação do protocolo posicionado sobre o cabeçalho IP, sendo divididos entre TCP, UDP e ICMP. Esta separação em grupos é realizada para que em cada grupo tenhamos apenas pacotes com os mesmos atributos, permitindo que a clusterização concentre-se apenas em mostrar o que é tráfego normal e anormal. Se misturássemos pacotes de protocolos diferentes, a clusterização também tentaria realizar o agrupamento por estes protocolos, o que poderia gerar um resultado indesejado. Foram selecionados apenas os protocolos TCP, UDP e ICMP pelo fato de serem os mais utilizados para transporte (TCP e UDP) e pela importância no controle da rede (ICMP). Os campos extraídos de um pacote compõe uma instância de dados que será processada na clusterização. É possível obter uma visão geral sobre este processo inicial de coleta através da Figura 7.

Tabela 2 – Conjunto dos campos selecionados para os protocolos TCP, UDP e ICMP em união aos escolhidos para o protocolo IP.

TCP + IP	**UDP + IP**	**ICMP + IP**
• tcp.flags.sys • tcp.flags.ack • tcp.flags.push • tcp.dstport • ip.flags.df • ip.flags • ip.flags.mf • ip.ttl • ip.frag.offset • ip.len • frame.len	• udp.dstport • udp.length • ip.flags.df • ip.flags • ip.flags.mf • ip.ttl • ip.frag.offset • ip.len • frame.len	• icmp.code • ip.flags.df • ip.flags • ip.flags.mf • ip.ttl • ip.frag.offset • ip.len • frame.len

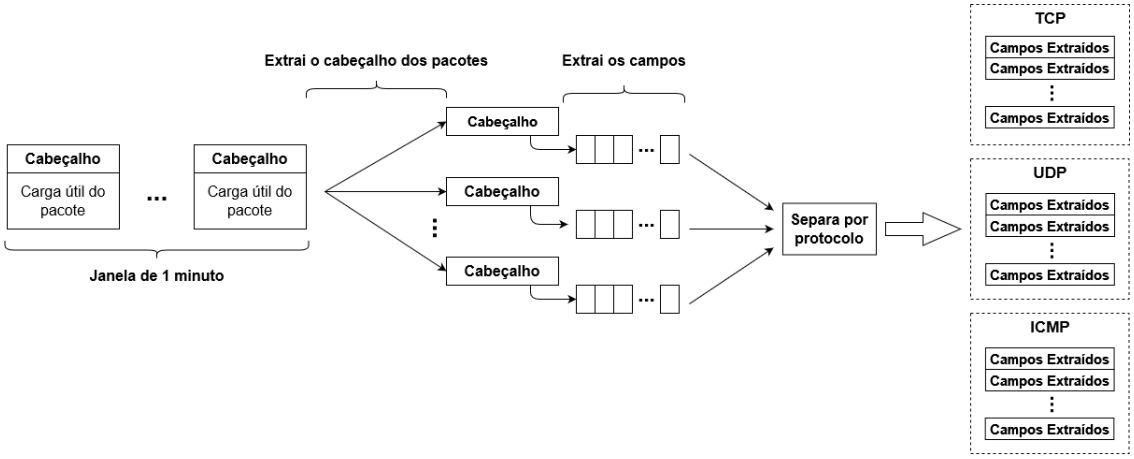


Figura 7 – Processo de coleta

3.2 Clusterização

Os intervalos de 1 minuto, na verdade, são acumulados para formar um conjunto maior. Este conjunto, que deve ser composto por até 30 intervalos de 1 minuto, age como uma janela deslizante, na qual o intervalo mais recente é adicionado e o mais antigo removido, como é possível observar no exemplo apresentado na Figura 8. Toda vez que a janela é atualizada com a remoção do intervalo mais antigo e a adição do mais novo, é executado um novo processo de clusterização sobre a janela.

A fase de clusterização é aquela na qual o algoritmo de aprendizado não supervisionado é aplicado. Para tal, antes realiza-se uma normalização dos valores presentes nas instâncias de dados, mantendo-as separadas de acordo com os seus protocolos (TCP, UDP ou ICMP). Esta normalização acontece por meio do cálculo do *z-score*, como é possível

observar na equação 3.1, na qual x é a amostra, u é a média das amostras, s o desvio padrão das amostras e z o novo valor após normalização [31].

$$z = \frac{x - u}{s} \tag{3.1}$$

Realizada a normalização, o DBSCAN é aplicado sobre as instâncias de dados inclusas na janela deslizante de 30 minutos (vide o exemplo da Figura 8) com distinção dos protocolos (TCP, UDP e ICMP). Ao fim da execução, obtém-se como resultado a classificação de cada instância de dados quanto ao *cluster* do qual faz parte ou se ele é um *outlier*.

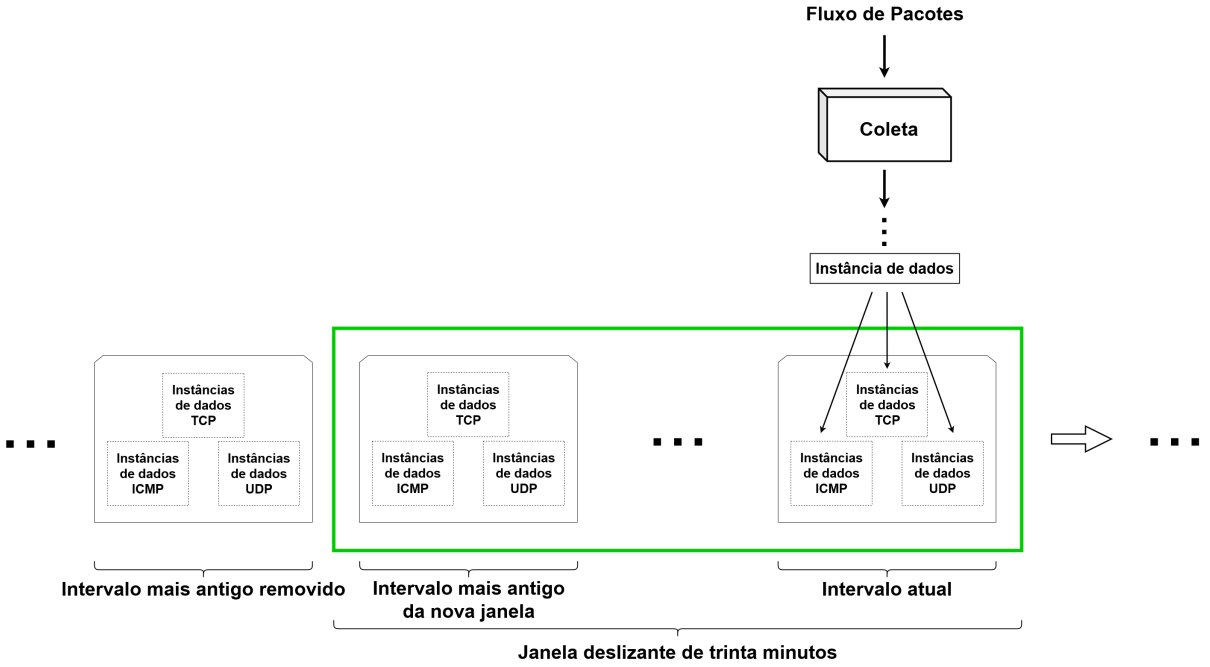


Figura 8 – Exemplo de funcionamento da janela deslizante

3.3 Detecção

Após o fim da fase de *clustering*, na qual foi realizada a aplicação do DBSCAN sobre as instâncias de dados contidas na janela deslizante (vide o exemplo da Figura 8), obtém-se como resultado a classificação como "elemento de *cluster*" ou "*outlier*" para cada instância de dados. Tendo este resultado, o processo encaminha-se para a fase de detecção.

Inicialmente, aplica-se o método PCA (Principal Component Analysis) sobre todos os dados inclusos na janela deslizante (similar ao que foi realizado na etapa do DBSCAN). Este procedimento é utilizado para redução da dimensionalidade linear usando decomposição de valor singular dos dados para projetá-los em um espaço dimensional inferior, com a implementação utilizada sendo baseada no método de Halko et al. [32]. Com isso,

podemos reduzir as dimensões de cada instância de dados (vários campos extraídos do cabeçalho) para apenas duas.

Após a aplicação do PCA, calcula-se o elemento representativo médio de cada cluster. Para tanto, é feita uma média entre os pontos pertencentes em cada cluster, levando em conta apenas os dois componentes principais de cada um deles.

Com o cálculo dos elementos representativos médios, temos os diferentes *clusters* $c_1...c_n$, *outliers* $o_1...o_m$ (cada *outlier* também é uma instância de dados) e os elementos médios $e_1...e_n$ calculados, no qual e_i representa as instâncias presentes no *cluster* c_i para $i = 1...n$. Então, para cada *outlier* o_i com $i = 1...m$, calculamos a distância euclidiana deste ponto o_i para todos os elementos médios $e_1...e_n$ e guardamos a menor distância encontrada para cada *outlier* o_i. Com a obtenção das menores distâncias entre cada *outlier* e os pontos médios dos clusters, calcula-se uma média aritmética para todas estas distâncias. De maneira resumida, estamos calculando a distância entre cada outlier e o cluster mais próximo a ele.

Estas médias aritméticas, calculadas ao fim de cada intevalo de 1 minuto (vide subcapítulo 3.1), formam uma série que, quando assume um novo valor considerado anômalo com relação aos anteriores, isto é, uma mudança abrupta, caracteriza também uma possível mudança abrupta no comportamento do dispositivo. O surgimento de pacotes muito diferentes no tráfego de rede reflete na presença de *outliers* demasiadamente distantes dos *clusters* já existentes, afetando a distância média calculada. Portanto, com a análise do comportamento destas médias pretende-se também analisar o comportamento do dispositivo que, em casos de grande alteração repentina, pode nos apresentar uma situação de ataque.

Dada esta hipótese, aplica-se o teste de Page Hinkley sobre a série de distâncias médias para detectar mudanças abruptas em seu comportamento automaticamente. Caso o teste detecte uma mudança abrupta de comportamento na série de dados univariada, entende-se que o tráfego de pacotes adotou uma característica de ataque, retornando o estado do teste ao anterior à inserção da média das distâncias e excluindo o intervalo de 1 minuto mais recente das execuções do DBSCAN seguintes. Caso uma mudança não tenha sido detectada, o intervalo de pacotes mais recente permanece na janela deslizante de até trinta intervalos citada anteriormente (vide Figura 8), removendo o intervalo mais antigo (ou seja, na primeira posição)

4 RESULTADOS

Este capítulo apresenta os resultados obtidos através da aplicação da solução proposta no capítulo 3 sobre os dados provenientes de uma rede IoT simulada com diversos dispositivos e tráfego de pacotes TCP, UDP e ICMP.

4.1 Conjunto de dados de teste

Para a realização dos testes foram utilizados os dados dispostos em um conjunto com pacotes provenientes do tráfego de uma rede IoT simulada apresentada no trabalho de Anthi et. al. [16] com os seguintes dispositivos:

- Hive Hub: Permite a conexão de todos os outros dispositivos da fabricante Hive em ambiente domiciliar, oferecendo o controle destes através de um aplicativo de celular;
- TP-Link NC200: Câmera residencial com sensor de movimento que notifica o usuário e disponibiliza verificar suas gravações/áudio através de um aplicativo;
- Lifx Smart Lamp: Lâmpada inteligente que pode ter a cor ajustada ou ser controlada remotamente por aplicativo;
- TP-Link SmartPlug: Tomada residencial que permite controlar outros dispositivos como ventiladores ou lâmpadas conectados a ela por aplicativo;
- Samsung Smart Things Hub: Permite conectar fechaduras inteligentes, lâmpadas, entre outros dispositivos para receber notificações remotamente sobre o que acontece em um ambiente domiciliar.

O conjunto de dados utilizado também possui pacotes rotulados como ataques do tipo DoS (*Denial of Service*), MITM (*Man-in-the-Middle*), *Scanning*, *iot-toolkit* e *Deauth*. A presença desta rotulação posteriormente possibilitou a verificação da eficácia da solução atráves da comparação dos resultados, bem como o cálculo de falsos-positivos.

Inicialmente, os pacotes foram separados por dispositivo e protocolo (TCP, UDP e ICMP). Após separados, estes pacotes foram ordenados de forma crescente segundo um campo de *timestamp* referente ao momento real de recebimento do pacote em cada dispositivo na rede IoT. A presença deste campo de *timestamp* foi importante para possibilitar a simulação do momento de entrada de cada instância de dados no teste posteriormente, visto que a solução abordada no capítulo 3 envolve a aplicação constante do DBSCAN ao final de cada intervalo de pacotes de 1 minuto coletado em tempo real.

4.2 Análise visual do comportamento da execução do DBSCAN

Durante a execução dos testes, foram gerados gráficos bidimensionais com o objetivo de observar o comportamento da aplicação do DBSCAN sobre a janela deslizante de 30 minutos. Os pacotes pertencentes a cada intervalo da janela deslizante foram representados no gráfico como pontos no espaço de busca, no qual os dois componentes principais resultantes da execução do PCA para todos os atributos de cada instância de dados foram considerados como suas coordenadas x e y. Uma cor também foi atribuída para cada instância de dados (representada como ponto) de acordo com o *cluster* formado na execução do DBSCAN, recebendo a mesma cor todos os integrantes de um mesmo *cluster*. Caso um ponto fosse classificado como *outlier*, a cor atribuída foi o preto.

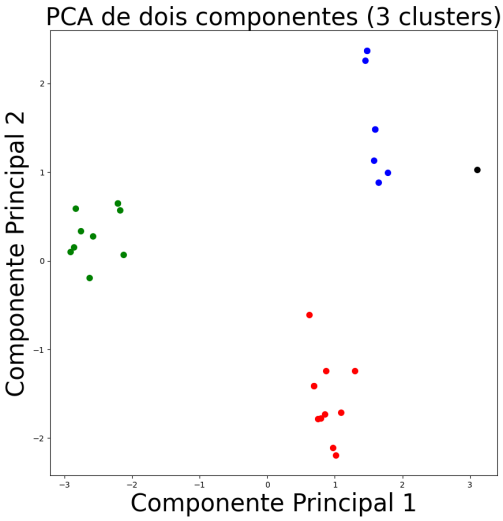


Figura 9 – Resultado da última execução do DBSCAN antes da ocorrência do primeiro ataque em tráfego TCP no dispositivo TP-Link NC200

Dentre os vários gráficos obtidos como resultado, é possível observar na Figura 9 e na Figura 10 dois exemplos. Os respectivos gráficos foram obtidos através da execução do DBSCAN sobre os pacotes do tráfego TCP do dispositivo TP-Link NC200. Na Figura 9, é possível observar o resultado da clusterização para uma situação de tráfego normal, na qual nenhum pacote rotulado como ataque estava presente na janela deslizante de 30 minutos. Já na Figura 10, é apresentado o primeiro resultado da execução do DBSCAN sobre uma janela deslizante envolvendo um ataque, isto é, um pacote malicioso estava presente no intervalo de 1 minuto mais recente enquanto no restante o tráfego era normal.

Percebe-se entre os dois exemplos uma diferença notável quanto à situação de tráfego normal e o surgimento de um ataque. No gráfico da Figura 9, por mais que os pacotes tenham sido agrupados em três *clusters* ou classificados como *outliers*, nota-se que não há uma grande distância entre os clusters e o outlier. Já no gráfico da Figura 10,

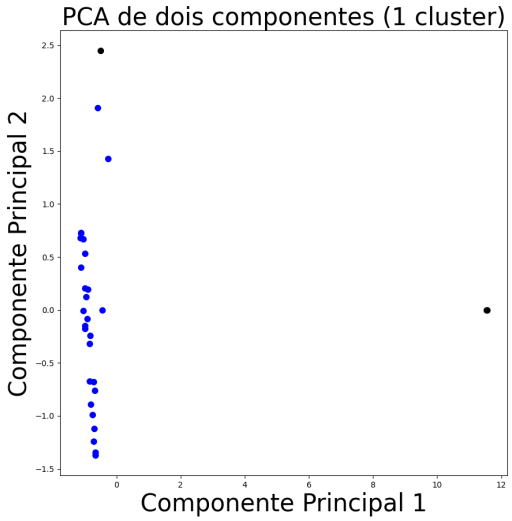


Figura 10 – Resultado da execução do DBSCAN para a primeira ocorrência de ataque em TCP no dispositivo TP-Link NC200

enquanto a variação entre os valores da componente principal 2 não foi tão significativa, é possível notar na componente principal 1 uma grande diferença quando em comparação com a Figura 10. Isto indica que há uma distância maior entre o outlier que surgiu (ataque) e os demais pacotes que compunham esta janela.

Os pacotes que antes haviam sido classificados em três *clusters* (Figura 9) na execução antes da primeira ocorrência de ataque, agora foram compreendidos pelo DBSCAN no caso da Figura 10 como membros de apenas 1 *cluster* (representado em azul), com exceção de alguns *outliers* logo acima que não são referentes a tráfego malicioso. Isto ocorreu devido a uma nova execução do PCA sobre os dados do caso da Figura 10, visto que, como a diferença na componente principal 1 diminui, o DBSCAN agora compreendeu que estas instâncias de dados estão densamente conectadas por estarem dentro do raio ϵ de cada uma (vide 2.4.1).

Com isso, compreende-se que a presença de um pacote malicioso realmente impacta no comportamento da execução do DBSCAN, seja na formação de *clusters* ou na rotulação como *outlier*. Percebe-se também que existe, de fato, uma grande diferença entre os dados do cabeçalho de pacotes originários de tráfego normal e os considerados como maliciosos. Estes dois fatores juntos certamente contribuem para a detecção de ataques realizados a um dispositivo, visto que a variação nos valores das componentes principais aliada ao comportamento na formação dos *clusters* e *outliers* resultariam em uma maior média de distância entre os clusters formados e *outliers*.

4.3 Testes e análise dos resultados

Tendo em mãos a solução desenvolvida durante o trabalho, optou-se por realizar execuções utilizando diversas configurações de hiperparâmetros para o DBSCAN (ϵ) e o teste Page-Hinkley (α e λ), visando verificar quais são as melhores combinações e entender como o sistema proposto reage às mudanças nestes valores. Mais precisamente, foram realizados testes variando o valor de ϵ entre $0,5 \ldots 3,0$ com passos de $0,5$, o limiar λ entre 10...100 com passos de 10 e o valor de α entre $0,1 \ldots 1,0$ com passos de $0,1$. Para o mínimo de elementos em um *cluster*, foi utilizado o valor 5, referente ao valor padrão para a implementação do DBSCAN em *Python* disponibilizada na biblioteca *scikit-learn*.

Para uma melhor análise dos resultados, foram escolhidos os cinco melhores casos para cada dispositivo e em cada protocolo (TCP, UDP e ICMP). Os resultados foram escolhidos de acordo com o cálculo das métricas *F1 score*, *precision* e *recall*. As três métricas variam de 0 a 100%, sendo 100% o melhor resultado possível. Para o cálculo destas métricas, são definidos os conceitos de verdadeiro positivo (VP), verdadeiro negativo (VN), falso positivo (FP) e falso negativo (FN). As métricas, por fim, podem ser definidas da seguinte forma:

- *precision*: no conjunto de pacotes classificados como ataques, esta métrica mostra quantos realmente eram ataques;

$$precision = \frac{VP}{VP + FP}$$

- *recall*: considerando o conjunto de pacotes que realmente eram ataques, esta métrica mostra quantos destes pacotes foram classificados corretamente;

$$recall = \frac{VP}{VP + FN}$$

- *F1 score*: avalia a cobertura e a precisão da classificação, considerando a média ponderada da entre *precision* e *recall*;

$$F1 = \frac{2VP}{2VP + FP + FN}$$

Também foram realizados testes sem a utilização do PCA, visando verificar os ganhos positivos da aplicação deste método. Nos casos sem PCA, a distância entre o centro representativo de um *cluster* e os *outliers* foi calculada através da distância euclidiana n-dimensional, na qual cada campo selecionado dos pacotes (vide Tabela 2) representou uma dimensão neste cálculo.

Os resultados foram dispostos em tabelas que apresentam a relação entre os valores dos hiperparâmetros no momento do teste e os valores obtidos para as métricas calculadas. Estes resultados podem ser observados nas Tabelas 3, 4, 5, 6 e 7, nas quais os cinco melhores casos para cada dispositivo (combinação entre *precision*, *recall* e *F1 score*) são apresentados para o teste com e sem a utilização do PCA.

Hive Hub							
Tipo do Teste	Protocolo	ϵ	λ	α	*precision*	*recall*	*F1 score*
Sem PCA	TCP	1.0	10	0.8	100%	100%	100%
		1.5	10	0.8	100%	100%	100%
		1.5	20	0.9	100%	100%	100%
		2.0	20	0.9	100%	100%	100%
		2.0	10	0.8	100%	98%	99%
	UDP	0.5	10	1.0	96%	100%	98%
		0.5	20	1.0	96%	100%	98%
		0.5	30	1.0	96%	100%	98%
		1.0	10	1.0	96%	100%	98%
		1.0	20	1.0	96%	100%	98%
	ICMP	0.5	10	0.7	99%	100%	100%
		0.5	10	0.8	99%	100%	100%
		0.5	10	0.9	99%	100%	100%
		0.5	10	1.0	99%	100%	100%
		0.5	20	0.8	99%	100%	100%
Com PCA	TCP	1.5	10	0.8	100%	100%	100%
		1.5	20	0.9	100%	100%	100%
		2.0	20	0.9	100%	100%	100%
		2.0	10	0.8	100%	99%	99%
		1.0	10	0.8	95%	100%	97%
	UDP	Não foi possível calcular as métricas (divisão por zero)					
	ICMP						

Tabela 3 – Resumo dos melhores resultados para o dispositivo Hive Hub

Ao observar os dados apresentados nas tabelas, é possível afirmar que todos os dispositivos tiveram ao menos um resultado excelente. Isso mostra que o algoritmo proposto conseguiu se adaptar às diferentes realidades de cada dispositivo quanto ao comportamento do tráfego e características dos dados do cabeçalho de cada pacote, com grande capacidade de detecção dos ataques e de evitar falsos-positivos, tendo em vista os altos valores obtidos através do cálculo do *recall* e *precision*, respectivamente. Quanto ao PCA, nota-se que sua utilização não trouxe ganhos positivos ao resultado, portanto conclui-se não compensar utilizá-lo no algoritmo proposto.

Mesmo tendo obtido resultados considerados como excelentes no geral, certos dispositivos não demonstraram um resultado tão bom para determinados protocolos quando comparados ao do TCP. Para analisar melhor o motivo desta diferença, foram gerados gráficos bidimensionais que mostram o valor da média de distância entre *outliers* e elementos representativos centrais dos *clusters*.

Com estes gráficos, tornou-se possível visualizar o comportamento desta série univariada através do tempo, isto é, do percorrer da janela deslizante. Foram também indicados nestes gráficos os pontos reais de início dos ataques, bem como o tipo de cada um. Os

TP-Link NC200							
Tipo do Teste	Protocolo	ϵ	λ	α	*precision*	*recall*	*F1 score*
Sem PCA	TCP	2.0	20	0.7	100%	100%	100%
		2.0	50	0.9	100%	100%	100%
		2.5	30	0.8	100%	100%	100%
		2.5	60	0.9	100%	100%	100%
		3.0	20	0.6	100%	100%	100%
	UDP	1.0	10	0.5	74%	68%	71%
		0.5	10	0.3	72%	68%	70%
		0.5	10	0.4	72%	68%	70%
		1.5	10	0.5	72%	68%	70%
		1.5	10	0.6	88%	59%	70%
	ICMP	Não foi possível calcular as métricas (divisão por zero)					
Com PCA	TCP	1.5	40	0.9	100%	100%	100%
		2.5	20	0.7	100%	100%	100%
		2.5	30	0.8	100%	100%	100%
		2.5	50	0.9	100%	100%	100%
		3.0	30	0.8	100%	100%	100%
	UDP	1.5	10	0.6	93%	70%	80%
		1.5	10	0.5	88%	72%	79%
		0.5	10	0.2	72%	75%	73%
		0.5	10	0.4	77%	69%	73%
		1.0	10	0.5	79%	69%	73%
	ICMP	Não foi possível calcular as métricas (divisão por zero)					

Tabela 4 – Resumo dos melhores resultados para o dispositivo TP-Link NC200

momentos em que o teste Page-Hinkley detectou mudanças abruptas na série univariada foram marcados em vermelho logo abaixo do valor da média de distância.

Na Figura 11 e na Figura 12, observam-se dois exemplos de gráficos que foram gerados utilizando a melhor configuração de hiperparâmetros para os dispositivos *Samsung Smart Things Hub* e *TP-Link NC200* no protocolo UDP, respectivamente. É possível perceber, através destes gráficos, que mesmo em situações com ausência de ataques houve grande variação na média de distância entre *outliers* e elementos representativos centrais.

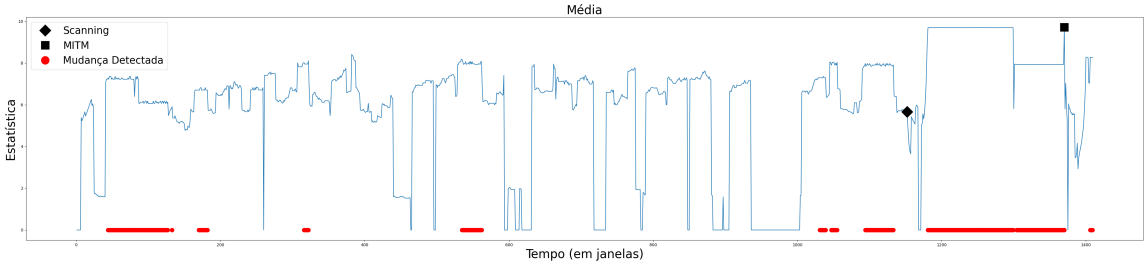


Figura 11 – Exemplo visual do teste para o protocolo UDP no dispositivo Samsung Smart Things Hub

Quanto aos hiperparâmetros utilizados na obtenção dos melhores resultados, não foi possível observar um padrão. Não há algo que indique a presença de uma faixa melhor de valores para estes hiperparâmetros, demonstrando que a aplicação dos algoritmos (DBSCAN e teste Page-Hinkley) em cenários diferentes dos que foram apresentados neste

Lifx Smart Lamp							
Tipo do Teste	Protocolo	ϵ	λ	α	*precision*	*recall*	*F1 score*
Sem PCA	TCP	0.5	10	0.5	100%	100%	100%
		0.5	10	0.7	100%	100%	100%
		0.5	20	0.8	100%	100%	100%
		0.5	20	0.9	100%	100%	100%
		0.5	30	0.9	100%	100%	100%
	UDP	0.5	10	0.3	100%	100%	100%
		0.5	10	0.4	100%	100%	100%
		0.5	10	0.5	100%	100%	100%
		0.5	10	0.6	100%	100%	100%
		0.5	10	0.7	100%	100%	100%
	ICMP	0.5	10	0.5	100%	3%	5%
		0.5	20	0.8	100%	3%	5%
		0.5	40	0.9	100%	2%	5%
		0.5	50	0.9	100%	3%	5%
		1.0	10	0.5	100%	3%	5%
Com PCA	TCP	0.5	10	0.7	100%	100%	100%
		0.5	20	0.8	100%	100%	100%
		0.5	20	0.9	100%	100%	100%
		0.5	30	0.9	100%	100%	100%
		0.5	40	0.9	100%	100%	100%
	UDP	Não foi possível calcular as métricas (divisão por zero)					
	ICMP	0.5	10	0.5	100%	3%	5%
		0.5	20	0.8	100%	3%	5%
		0.5	50	0.9	100%	3%	5%
		1.0	10	0.5	100%	3%	5%
		0.5	40	0.9	100%	2%	5%

Tabela 5 – Resumo dos melhores resultados para o dispositivo Lifx Smart Lamp

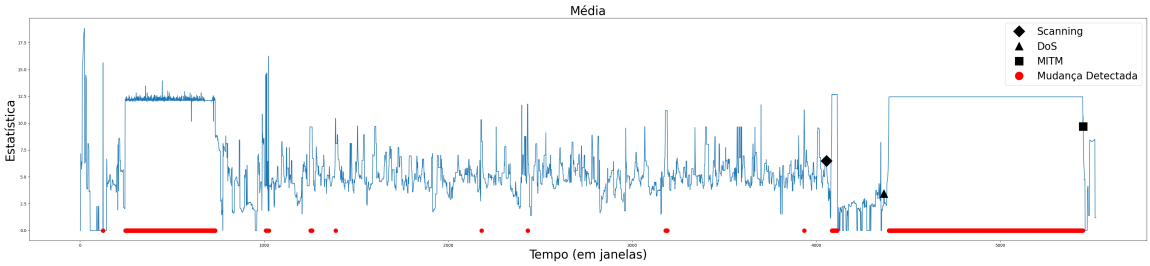


Figura 12 – Exemplo visual do teste para o protocolo UDP no dispositivo TP-Link NC200

trabalho exigiriam mais testes para a obtenção dos valores corretos para cada hiperparâmetro. Para um usuário comum, isto ainda é um obstáculo, o que nos leva a concluir que um trabalho no desenvolvimento de um método para a obtenção correta destes hiperparâmetros seria de grande importância.

TP-Link SmartPlug							
Tipo do Teste	Protocolo	ϵ	λ	α	*precision*	*recall*	*F1 score*
Sem PCA	TCP	0.5	10	0.1	100%	100%	100%
		0.5	10	0.2	100%	100%	100%
		0.5	10	0.3	100%	100%	100%
		1.0	10	0.1	100%	100%	100%
		1.0	10	0.2	100%	100%	100%
	UDP	0.5	10	0.7	100%	100%	100%
		0.5	10	0.8	100%	100%	100%
		0.5	20	0.8	100%	100%	100%
		0.5	20	0.9	100%	100%	100%
		0.5	30	0.9	100%	100%	100%
	ICMP	Não há pacotes					
Com PCA	TCP	0.5	10	0.1	100%	100%	100%
		0.5	10	0.2	100%	100%	100%
		0.5	10	0.3	100%	100%	100%
		1.0	10	0.1	100%	100%	100%
		1.0	10	0.2	100%	100%	100%
	UDP	0.5	10	0.6	100%	100%	100%
		0.5	10	0.7	100%	100%	100%
		0.5	10	0.8	100%	100%	100%
		0.5	20	0.7	100%	100%	100%
		0.5	20	0.8	100%	100%	100%
	ICMP	Não há pacotes					

Tabela 6 – Resumo dos melhores resultados para o dispositivo TP-Link SmartPlug

Samsung Smart Things Hub							
Tipo do Teste	Protocolo	ϵ	λ	α	*precision*	*recall*	*F1 score*
Sem PCA	TCP	0.5	10	0.9	100%	99%	99%
		2.0	10	0.8	98%	93%	96%
		2.5	10	0.6	92%	100%	96%
		3.0	10	0.6	90%	94%	92%
		1.0	100	1.0	100%	84%	91%
	UDP	1.5	10	0.8	74%	50%	60%
		1.0	10	0.7	47%	78%	59%
		0.5	70	1.0	95%	20%	33%
		0.5	50	1.0	100%	19%	32%
		0.5	60	1.0	92%	19%	32%
	ICMP	Não foi possível calcular as métricas (divisão por zero)					
Com PCA	TCP	3.0	10	0.6	100%	100%	100%
		2.0	10	0.7	99%	100%	100%
		2.0	10	0.8	100%	99%	100%
		0.5	10	0.9	98%	99%	98%
		1.0	10	0.9	98%	98%	98%
	UDP	1.5	10	0.8	74%	50%	60%
		1.0	10	0.7	47%	78%	59%
		1.0	10	0.9	100%	19%	32%
		1.0	40	1.0	100%	19%	32%
		0.5	60	1.0	93%	19%	32%
	ICMP	Não foi possível calcular as métricas (divisão por zero)					

Tabela 7 – Resumo dos melhores resultados para o dispositivo Samsung Smart Things Hub

5 CONCLUSÃO

Visto o objetivo de detectar ataques realizados a redes IoT utilizando o Aprendizado de Máquina não supervisionado, este trabalho apresentou a elaboração de uma solução que utiliza o algoritmo DBSCAN juntamente ao teste Page-Hinkley para detectar a presença de tráfego malicioso de pacotes em dispositivos pertencentes a uma rede IoT para o os protocolos TCP, UDP e ICMP.

Foram realizados testes utilizando o conjunto de dados apresentado no trabalho de Anthi et. al. [16] e calculadas as métricas *precision*, *recall* e *F1 score* para cada dispositivo em várias configurações de hiperparâmetros do DBSCAN e teste Page-Hinkley.

Os resultados obtidos foram bastante satisfatórios, apresentando um valor de *F1 score* próximo a 100% em boa parte dos casos para o protocolo TCP, o que se deve a uma ótima combinação entre valores de *precision* e *recall*, significando uma baixa taxa de falsos positivos e alta eficácia na detecção de ataques reais, respectivamente. Em contraste, os protocolos UDP e ICMP apresentaram alguns casos de maior dificuldade. No geral, os resultados apontam para uma possível eficácia em casos mais amplos, demonstrando também a eficácia do uso do Aprendizado de Máquina não supervisionado para a detecção de ataques a redes IoT, apresentando uma elevada taxa de detecção e baixo número de falsos-positivos.

Entretanto, através da análise dos resultados não foi possível encontrar um padrão na configuração dos valores dos hiperparâmetros para o DBSCAN e teste Page-Hinkley. Isto levanta uma boa proposta para trabalhos futuros que objetivem o desenvolvimento de um método para a obtenção correta dos valores destes hiperparâmetros.

REFERÊNCIAS

[1] STOKES, P. *4 Stages of IoT architecture explained in simple words*. 2018. Acesso em: 11 de nov. de 2021. Disponível em: <https://medium.datadriveninvestor.com/4-stages-of-iot-architecture-explained-in-simple-words-b2ea8b4f777f>.

[2] ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *kdd*. [S.l.: s.n.], 1996. v. 96, n. 34, p. 226–231.

[3] SCHUBERT, E. et al. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, ACM New York, NY, USA, v. 42, n. 3, p. 1–21, 2017.

[4] STEIN, B.; BUSCH, M. Density-based cluster algorithms in low-dimensional and high-dimensional applications. In: CITESEER. *International Workshop on Text-Based Information Retrieval*. [S.l.], 2005. p. 45.

[5] SCHROTH, C.; SIEBERT, J.; GROß, J. *Time Traveling with Data Science: Focusing on Change Point Detection in Time Series Analysis (Part 2)*. 2021. Acesso em: 11 de nov. de 2021. Disponível em: <https://www.iese.fraunhofer.de/blog/change-point-detection/>.

[6] ZARPELÃO, B. B. et al. A survey of intrusion detection in internet of things. *Journal of Network and Computer Applications*, Elsevier, v. 84, p. 25–37, 2017.

[7] BEZERRA, V. H. et al. Iotds: A one-class classification approach to detect botnets in internet of things devices. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 19, n. 14, p. 3188, 2019.

[8] RUSSELL, S. J. *Artificial intelligence a modern approach*. [S.l.]: Pearson Education, Inc., 2010.

[9] BELAVAGI, M. C.; MUNIYAL, B. Performance evaluation of supervised machine learning algorithms for intrusion detection. *Procedia Computer Science*, Elsevier, v. 89, n. 2016, p. 117–123, 2016.

[10] STEVANOVIC, M.; PEDERSEN, J. M. An efficient flow-based botnet detection using supervised machine learning. In: IEEE. *2014 international conference on computing, networking and communications (ICNC)*. [S.l.], 2014. p. 797–801.

[11] ASHTON, K. et al. That 'internet of things' thing. *RFID journal*, v. 22, n. 7, p. 97–114, 2009.

[12] GREENGARD, S. *The internet of things*. [S.l.]: MIT press, 2021.

[13] ROSE, K.; ELDRIDGE, S.; CHAPIN, L. The internet of things: An overview. *The internet society (ISOC)*, v. 80, p. 1–50, 2015.

[14] HALLER, S. The things in the internet of things. *Poster at the (IoT 2010). Tokyo, Japan, November*, v. 5, n. 8, p. 26–30, 2010.

[15] VASHI, S. et al. Internet of things (iot): A vision, architectural elements, and security issues. In: IEEE. *2017 international conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*. [S.l.], 2017. p. 492–496.

[16] ANTHI, E. et al. A supervised intrusion detection system for smart home iot devices. *IEEE Internet of Things Journal*, IEEE, v. 6, n. 5, p. 9042–9053, 2019.

[17] XU, T.; WENDT, J. B.; POTKONJAK, M. Security of iot systems: Design challenges and opportunities. In: IEEE. *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. [S.l.], 2014. p. 417–423.

[18] FERRARA, P. et al. Static analysis for discovering iot vulnerabilities. *International Journal on Software Tools for Technology Transfer*, Springer, v. 23, n. 1, p. 71–88, 2021.

[19] MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações*, Manole Ltda, v. 1, n. 1, p. 32, 2003.

[20] ALPAYDIN, E. *Introduction to machine learning*. [S.l.]: MIT press, 2020.

[21] LISON, P. An introduction to machine learning. *Language Technology Group (LTG)*, v. 1, n. 35, p. 1–35, 2015.

[22] LIKAS, A.; VLASSIS, N.; VERBEEK, J. J. The global k-means clustering algorithm. *Pattern recognition*, Elsevier, v. 36, n. 2, p. 451–461, 2003.

[23] DY, J. G.; BRODLEY, C. E. Feature selection for unsupervised learning. *Journal of machine learning research*, v. 5, n. Aug, p. 845–889, 2004.

[24] DUBES, R.; JAIN, A. K. Clustering techniques: the user's dilemma. *Pattern Recognition*, Elsevier, v. 8, n. 4, p. 247–260, 1976.

[25] ROKACH, L.; MAIMON, O. Clustering methods. In: *Data mining and knowledge discovery handbook*. [S.l.]: Springer, 2005. p. 321–352.

[26] HAN, J.; PEI, J.; KAMBER, M. *Data mining: concepts and techniques*. [S.l.]: Elsevier, 2011.

[27] BASSEVILLE, M.; NIKIFOROV, I. V. et al. *Detection of abrupt changes: theory and application*. [S.l.: s.n.]. v. 104.

[28] GAMA, J. Change detection. *Knowledge Discovery from Data Streams*, Chapman and Hall/CRC, p. 53–68, 2010.

[29] PAGE, E. S. Continuous inspection schemes. *Biometrika*, JSTOR, v. 41, n. 1/2, p. 100–115, 1954.

[30] HARTLAND, C. et al. Change point detection and meta-bandits for online learning in dynamic environments. In: *CAp 2007: 9è Conférence francophone sur l'apprentissage automatique*. [S.l.: s.n.], 2007. p. 237–250.

[31] CHUBB, H.; SIMPSON, J. M. The use of z-scores in paediatric cardiology. *Annals of pediatric cardiology*, Wolters Kluwer–Medknow Publications, v. 5, n. 2, p. 179, 2012.

[32] HALKO, N.; MARTINSSON, P.-G.; TROPP, J. A. Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions. California Institute of Technology, 2009.