Universidade Estadual de Londrina

GABRIEL KEITH TAZIMA

APLICAÇÃO DO ALGORITMO DE CLUSTERIZAÇÃO DE FLUXOS CONTÍNUOS DE DADOS DENSTREAM NA DETECÇÃO DE ATAQUES EM INTERNET DAS COISAS

LONDRINA
2024

GABRIEL KEITH TAZIMA

APLICAÇÃO DO ALGORITMO DE CLUSTERIZAÇÃO DE FLUXOS CONTÍNUOS DE DADOS DENSTREAM NA DETECÇÃO DE ATAQUES EM INTERNET DAS COISAS

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Bruno Bogaz Zarpelão

LONDRINA

2024

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

T248a Tazima, Gabriel Keith.
Aplicação do Algoritmo de Clusterização de Fluxos Contínuos de Dados DenStream na Detecção de Ataques em Internet das Coisas / Gabriel Keith Tazima. - Londrina, 2024.
50 f. : il.

Orientador: Bruno Bogaz Zarpelão.
Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Programa de Pós-Graduação em Ciência da Computação, 2024.
Inclui bibliografia.

1. Aprendizado incremental - Tese. 2. Segurança de redes - Tese. 3. Internet das Coisas - Tese. 4. DenStream - Tese. I. Zarpelão, Bruno Bogaz. II. Universidade Estadual de Londrina. Centro de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU 519

GABRIEL KEITH TAZIMA

APLICAÇÃO DO ALGORITMO DE CLUSTERIZAÇÃO DE FLUXOS CONTÍNUOS DE DADOS DENSTREAM NA DETECÇÃO DE ATAQUES EM INTERNET DAS COISAS

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Mestre em Ciência da Computação.

BANCA EXAMINADORA

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

Prof. Dr. Rodolfo Miranda de Barros
Universidade Estadual de Londrina – UEL

Prof. Dr. Luiz Fernando Carvalho
Universidade Tecnológica Federal do Paraná – UTFPR

Londrina, 25 de Janeiro de 2024.

Dedico este trabalho a meu grande amigo Luiz.

AGRADECIMENTOS

Esta dissertação é fruto do apoio inestimável de diversas pessoas, cujas contribuições foram fundamentais para a sua concretização. Inicialmente, expresso minha sincera gratidão ao Professor Doutor Bruno Bogaz Zarpelão, meu dedicado orientador. Seu inabalável comprometimento, paciência, aconselhamentos e ajuda prática foram pilares essenciais para o desenvolvimento deste trabalho. Sem seu estímulo incansável, esta jornada não teria sido possível.

Aos colegas de Mestrado, em especial, agradeço ao Fernando Nakagawa, cujo apoio foi valioso em várias etapas desta trajetória acadêmica. Meus agradecimentos estendem-se a todos os amigos que sempre estiveram ao meu lado, com destaque para Gabriel Freitas e Tiago Pedrosa, que generosamente compartilharam seus conhecimentos, enriquecendo muito este estudo.

Por último, mas certamente não menos importante, expresso minha profunda gratidão pelo apoio incondicional da minha amada família: Antonio, Dirley, Deborah, Ednaldo, Eduardo, Karoline e Ivete. Vocês são a minha fonte inesgotável de inspiração. Sem a presença e o amor de vocês, eu não seria a pessoa que sou hoje. A cada um, meu mais sincero agradecimento por fazerem parte dessa jornada e por serem a força motriz por trás das minhas conquistas.

"Eu sou porque nós somos."
(Filosofia UBUNTU)

TAZIMA, G. K.. **Aplicação do Algoritmo de Clusterização de Fluxos Contínuos de Dados DenStream na Detecção de Ataques em Internet das Coisas**. 2024. 50f. Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual de Londrina – UEL, Londrina, 2024.

RESUMO

A *Internet* das Coisas (*Internet of Things - IoT*) pode ser definida como a convergência da Internet e objetos que podem se comunicar e interagir entre si. Para os próximos anos, aplicações de IoT devem se expandir e crescer exponencialmente. Diversas áreas como logística, indústria, saúde pública, automação residencial e monitoramento ambiental se beneficiarão deste novo paradigma. A IoT tem o potencial de prover facilidades e melhorias na vida cotidiana. Com a aproximação entre essas aplicações e os usuários finais, atividades que antes estavam imunes a ataques cibernéticos, podem estar ameaçadas por ações maliciosas. Dispositivos domésticos de IoT podem se tornar alvos de ataques e potencialmente ameaçar a segurança e a privacidade de indivíduos. Diferente de redes de Internet das Coisas industriais, por exemplo, em cenários residenciais, os dispositivos utilizados são mais heterogêneos e, normalmente, tem menor custo e capacidade computacional. Além disso, eles contam com aplicações que muitas vezes são descentralizadas e desenvolvidas de forma diferente pelos diversos fabricantes envolvidos. Uma das possibilidades de medidas de defesa é o provisionamento de um Sistema de Detecção de Intrusão (*Intrusion Detection System - IDS*), que seria capaz de detectar atividades maliciosas e gerar registros das mesmas. Há outros trabalhos que fazem detecção de ataques utilizando aprendizado de máquina, mas a maioria das soluções utiliza técnicas de aprendizado supervisionado por lotes e precisam de amostras normais e maliciosas para serem treinados, o que pode ser problemático em cenários de redes reais. A utilização de algoritmos de clusterização de fluxos de dados contínuos pode oferecer uma solução para os dois problemas mencionados. Esses algoritmos não exigem amostras de tráfego benigno e malicioso para serem treinados e também são capazes de aprender incrementalmente. Por outro lado, o grande desafio na utilização deles é compreender melhor o que devemos monitorar no comportamento dos clusters para identificar a ocorrência de ataques. Este trabalho analisou diferentes indicadores extraídos do comportamento dos clusters gerados pelo DenStream, um algoritmo de clusterização de fluxos contínuos de dados baseado em densidade, para compreender como eles podem contribuir na detecção da ocorrência de ataques no tráfego de rede. O conjunto de dados públicos utilizado neste trabalho conta com pacotes de tráfego normal e malicioso de diversos dispositivos com diferentes protocolos de rede. Os ataques contidos no conjunto de dados são: *Port Scanning*, Negação de serviço e *man-in-the-middle*. Os resultados dos experimentos mostraram que o monitoramento da distância máxima entre os centros dos clusters pode sinalizar a ocorrência de diferentes tipos de ataques.

Palavras-chave: Internet das Coisas, Aprendizado de fluxos contínuos de dados, Aprendizado Não-supervisionado, Detecção de Ataques, DenStream

TAZIMA, G. K.. **Application of the DenStream Data Stream Clustering Algorithm in Detecting Attacks in the Internet of Things**. 2024. 50p. Master's Thesis (Master in Science in Computer Science) – State University of Londrina, Londrina, 2024.

ABSTRACT

The Internet of Things (IoT) can be defined as the convergence of the Internet and objects that can communicate and interact with each other. For the coming years, IoT applications are expected to expand and grow exponentially. Several areas such as logistics, industry, public health, home automation and environmental monitoring will benefit from this new paradigm. The IoT has the potential to ease any tasks. With the approximation between these applications and end users, activities that were previously immune to cyberattacks may be threatened by malicious actions. In a pulverized form, home IoT devices can become targets of attacks and potentially threaten the security and privacy of individuals. Unlike industrial Internet of Things networks, for example, in residential scenarios, the devices are more heterogeneous and usually have lower cost and computational capacity. In addition, they rely on applications that are often decentralized and developed differently by the different manufacturers involved. A possible security control for these devices is the provision of an Intrusion Detection System (IDS), which would be able to detect malicious activities and generate records of them. There are other works that detect attacks using machine learning, but most solutions use batch supervised learning techniques and need benign and malicious samples to be trained, which can be challenging in real network scenarios. The use of clustering algorithms for data streams can offer a solution to the two problems mentioned. These algorithms do not require samples of benign and malicious traffic to be trained and are also capable of learning incrementally. On the other hand, the biggest challenge in using them is to better understand what we should monitor in the behavior of clusters to identify the occurrence of attacks. This work analyzed different indicators extracted from the behavior of clusters generated by DenStream, a desinity-based clustering algorithm for continuous data flows, to understand how they can contribute to detecting the occurrence of attacks in network traffic. The public data set used in this work includes normal and malicious traffic packets from different devices with different network protocols. The attacks contained in the dataset are: Port Scanning, Denial of Service and man-in-the-middle. The proposed experiment results, shows that monitoring the medium distance between clusters centers can indicate the occurrence of different types of attacks.

Keywords: Internet of Things, Stream Learning, Unsupervised Algorithm, Attack Detection, DenStream

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AD	Anomaly-based Detection
BLE	Bluetooth Low Energy
COBIT	Control Objectives for Information and related Technology
DDR4	Double Data Rate 4
DoS	Denial of Service
GSM	Global System for Mobile Communications
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
ITIL	Information Technology Infrastructure Library
IP	Internet Protocol
MHz	Megahertz
MITM	Man-in-the-middle Attack
RAM	Random Access Memory
R2L	Remote to Local
SD	Signature-based Detection
SDN	Software Defined Network
SLFN	Single Hidden Layer Feed-Forward Neural Network
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TI	Tecnologia da Informação
TTL	Time to Live

UDP	User Datagram Protocol
U2R	User to Root
WiFi	Wireless Fidelity
5G	Quinta geração de padrão de redes móveis

SUMÁRIO

1 INTRODUÇÃO

As aplicações de Internet da Coisas (*Internet of Things - IoT*) estão cada vez mais acessíveis e é inegável o impacto positivo que possuem na vida das pessoas permitindo, por exemplo, automação de tarefas, aumento de segurança física, monitoramento de saúde, etc. Porém, apesar das vantagens e benefícios que o uso desses dispositivos pode proporcionar, falhas graves de segurança e privacidade também podem surgir. O usuário final de soluções de IoT, normalmente, não compreende o funcionamento e que tipo de informações sensíveis podem estar sendo transmitidas por esses dispositivos. Isso piora quando equipamentos de procedência duvidosa são integrados à solução. Além do baixo nível de cibersegurança embarcado em alguns dos equipamentos, esses dispositivos costumam ter uma alta disponibilidade, o que os torna bons alvos de ataques como *botnets*, *malwares* e invasões de privacidade. Com isso em mente, fica cada vez mais evidente a necessidade de pesquisas e implementações de sistemas de cibersegurança para proteger essas novas redes tão vulneráveis e numerosas [1, 2, 3, 4].

Técnicas de cibersegurança para redes de computadores já existem há algum tempo. Porém, no contexto de IoT, algumas considerações precisam ser feitas. Redes IoT são compostas, em geral, por dispositivos com reduzido poder computacional. Com isso, controles de cibersegurança voltados a redes convencionais podem não ser adequados para dispositivos IoT. Existe ainda uma heterogeneidade grande entre os dispositivos que são utilizados em redes IoT (diferentes fabricantes, protocolos e implementações de cibersegurança, quando existem).

Além disso, o usuário comum raramente vai possuir qualquer tipo de preparo para lidar com configurações e análises que ajudam a aumentar a segurança em cenários de redes convencionais [2, 5, 6].

Uma das medidas de cibersegurança que podem ser aplicadas para lidar com a alta quantidade de ameaças são os Sistemas de Detecção de Intrusão (*Intrusion Detection Systems - IDS*) . Os IDSs trabalham de duas formas. Primeiramente, monitoram a rede em busca de anomalias, onde o IDS prepara perfis de comportamento normal e gera alertas quando encontra desvios deste comportamento. Na segunda forma, os IDSs são orientados a assinaturas. Características de ataques conhecidos são cadastradas no IDS, que monitora a rede procurando por atividades que possuam estas características, formalmente denominadas assinaturas [7, 8, 9, 10].

Muitas das soluções de detecção de ataques propostas na literatura são baseadas em algoritmos de aprendizado de máquina em lote. De forma geral, são técnicas de aprendizado supervisionado que requerem amostras rotuladas como tráfego normal ou malicioso

para treinamento. Em cenários reais, porém, não seria razoável criar a expectativa que usuários não especialistas teriam o conhecimento necessário para supervisionar e retreinar o modelo sempre que houver uma mudança no comportamento normal ou novos ataques forem desenvolvidos. Por isso, é importante pensar em uma forma de detectar ataques em um cenário de fluxos contínuo de dados que está suscetível, eventualmente, a mudanças de comportamento [11, 5, 6, 2, 12].

Uma alternativa a algoritmos de aprendizado em lote são os algoritmos de mineração de fluxos contínuo de dados. Uma vantagem desses algoritmos é que eles aprendem e alteram seu modelo de forma incremental, atualizando o modelo sempre que novas observações são processadas, sendo mais robusto em relação a mudanças de comportamento ou surgimento de ataques desconhecidos. Algoritmos de mineração de fluxos contínuos de dados, assim como ocorre com os algoritmos de aprendizado em lote, também podem ser não supervisionados. Nesta categoria, se encaixam os algoritmos de clusterização de fluxos contínuos de dados. Eles podem representar uma boa alternativa para a detecção de ataques em redes IoT, pois não requerem amostras rotuladas e se adaptam às modificações de comportamento que ocorrem nos fluxos analisados [13, 14].

O objetivo deste trabalho é investigar a possibilidade de aplicação do algoritmo de clusterização de fluxos contínuos de dados *DenStream* para detectar ataques no tráfego de redes IoT. O *DenStream* utiliza a aprendizagem incremental para criar e manter o seu modelo de clusterização. Por esta razão, pode fornecer respostas mais rápidas a mudanças de comportamento, mostrando uma adaptabilidade melhorada à medida que refina continuamente o seu modelo. Além disso, o algoritmo não requer o armazenamento de observações históricas, o que é importante, uma vez que este estudo considera cenários com restrições de armazenamento. Algoritmos como esse são focados em agrupar as observações de acordo com a semelhança entre elas, mas não incluem em seu escopo a classificação delas, apontando quais seriam normais e maliciosas. Portanto, um desafio na utilização do *DenStream* para detectar ataques é identificar quais características dos clusters produzidos por ele podem indicar que está ocorrendo um ataque. A principal hipótese do trabalho é que monitorar a distância entre os *micro-clusters* e os seus eventos de criação, remoção e clusterização pode oferecer indicadores de que ataques estão ocorrendo. Para tanto, será realizado um experimento em que, primeiramente, o tráfego de rede de cada equipamento IoT é dividido em três fluxos: TCP, UDP e ICMP. A partir daí, uma instância do *DenStream* será aplicada a cada um destes três tipos de fluxos, processando o tráfego pacote a pacote. Iremos repetir este experimento sobre dois conjuntos de dados. O primeiro é um conjunto de dados público que contém tráfego de rede de dispositivos IoT domésticos, extraído e disponibilizado originariamente por Anthi et al. [5]. O segundo é uma variação do primeiro gerada por Nakagawa et al. [15] onde os pesquisadores buscaram manipular o conjunto de dados de Anthi et al. para que apresentasse mais situações de transições entre estados normais e maliciosos.

A implementação de algoritmos de aprendizado incremental, como o *DenStream*, no âmbito da detecção de ataques em redes IoT, assume uma relevância estratégica na gestão de Tecnologia da Informação (TI) corporativa. A eficácia dessas ferramentas não se restringe apenas à segurança cibernética, mas se estende para a garantia da continuidade dos serviços, alinhando-se aos objetivos recomendado por normas como a Biblioteca de Infraestrutura de Tecnologia da informação ITIL (*Information Technology Infrastructure Library*) [16] e COBIT (*Control Objectives for Information and related Technology*) [17]. Ao adotar abordagens proativas na identificação e mitigação de ameaças, as organizações conseguem proteger a integridade de seus sistemas, prevenindo interrupções inesperadas nos serviços. Esse enfoque é particularmente crítico em um ambiente cada vez mais conectado, no qual a indisponibilidade ou comprometimento da qualidade dos serviços pode resultar em prejuízos financeiros e danos à reputação. Ao integrar algoritmos de aprendizado incremental, as empresas fortalecem suas práticas de gestão de TI, assegurando a robustez de seus serviços e aprimorando a resiliência operacional diante de ameaças emergentes, contribuindo assim para o alcance de padrões elevados de excelência e conformidade normativa.

Este trabalho é organizado da seguinte forma: o capítulo 2 contextualiza os conceitos teóricos, descrevendo Internet das Coisas, os fundamentos de aprendizado e clusterização de fluxos contínuos de dados, sistemas de detecção de intrusão e trabalhos relacionados. O capítulo 3 apresenta em detalhes o experimento proposto. O capítulo 4 mostra os resultados obtidos com o estudo, enquanto o capítulo 5 conclui o trabalho com algumas considerações finais.

2 REFERENCIAL TEÓRICO

2.1 Internet das Coisas

As aplicações de Internet das Coisas têm potencial de transformar positivamente a vida das pessoas. Soluções de IoT focadas em idosos, pessoas vulneráveis e pessoas com deficiência viabilizam/facilitam algumas tarefas diárias aumentando a autonomia e a auto estima desses indivíduos. Um exemplo é o uso de acelerômetros de relógios inteligentes para detectar quedas [18]. No contexto de IoT focada em segurança, é possível diminuir o risco de exposição a algumas atividades. Equipamentos de mineração operados a distância podem mitigar o risco de acidentes envolvendo mineradores [19]. Outro exemplo de segurança do trabalho é o uso de sensores conectados e espalhados em uma fábrica que podem medir a concentração de gases inflamáveis ou tóxicos no ar e alertar quando valores limítrofes forem detectados [20].

Os benefícios do paradigma da IoT são muitos e a demanda por soluções cresce a cada ano. Essa pressão move os diversos atores da indústria, que tentam reivindicar sua fatia desse mercado e que, às vezes, acabam negligenciando a cibersegurança no desenvolvimento de seus dispositivos e soluções. Brinquedos conectados à Internet registrando e enviando áudio capturado sem autorização [21], babás digitais que transmitem vídeos sem criptografia [22], entre outros casos, demonstram o risco ao qual a sociedade pode estar exposta.

Na literatura, encontramos diversas propostas para melhorar as condições de cibersegurança desses dispositivos, porém, ainda existem muitos problemas não resolvidos. É necessário amadurecer a cibersegurança em IoT, além disso, questões de legislação e controle de qualidade de produção que definam diretrizes de implementação de cibersegurança são muito recentes. A ordem executiva EO14028 [23] de maio de 2021 é o exemplo de uma iniciativa do governo americano para definir padrões de cibersegurança no desenvolvimento de *software* e *hardware* para IoT. Além disso, como uma nova tecnologia, ainda existem diversas arquiteturas de implantação de IoT com protocolos diferentes para a comunicação entre dispositivos.

Em sua revisão, Al-Fuqaha et al. [24] discutem as arquiteturas mais comuns de IoT, desde a arquitetura básica de três camadas que consiste da camada de aplicação, camada de rede e camada de percepção, até proposições de arquitetura de maior abstração com até cinco camadas. A seguir iremos discutir brevemente cada uma delas.

- Camada de Objetos/Percepção: representa os sensores e atuadores físicos da rede que coletam, processam e reagem às informações da rede. Nesta camada, informações

do mundo físico são digitalizadas e encaminhadas para a camada de abstração dos objetos. O paradigma do "*Big Data*" e o alto volume de dados gerados pela aplicação em massa de soluções IoT se inicia aqui.

- Camada de abstração de objetos: nesta camada, os dados digitalizados gerados pela camada de objetos são transferidos para a camada de gestão de serviços. Hoje, para fazer essa transferência existem diversas tecnologias como: 5G, GSM, WiFi, BLE, ZigBee, etc.

- Camada de gestão de serviços: responsável por gerenciar e controlar os serviços oferecidos pelos dispositivos IoT, garantindo a disponibilidade, confiabilidade, segurança e desempenho dos serviços. Isso inclui funcionalidades como descoberta e registro de dispositivos, gerenciamento de identidade e acesso, monitoramento e análise de desempenho, gerenciamento de atualizações de software e políticas de segurança.

- Camada de aplicação: provê os serviços requisitados pelo usuário final. Por exemplo, o usuário requisita a leitura da quantidade de energia gerada por sua instalação fotovoltaica.

- Camada de negócios: gestão geral das atividades e serviços do sistema IoT. Nessa camada, análises de *Big Data* podem ser feitas baseadas nos dados providos pela camada de aplicação.

Neshenko et al. [25] definem três categorias de vulnerabilidades que estão ligadas às camadas propostas para a arquitetura de IoT.

- Vulnerabilidades de dispositivos: com a grande quantidade de dispositivos operando em alta disponibilidade e sem monitoramento, atacantes podem explorar as vulnerabilidades e ganhar acesso físico à rede, alterar comportamento de serviços ou acessar os dados armazenados neles. Trappe et al. [26] destacam o problema de como o baixo consumo energético e poder computacional dos dispositivos IoT são grandes limitantes na implantação de estratégias de cibersegurança na camada de percepção.

- Vulnerabilidades de rede: envolvem fraquezas específicas dos protocolo e redes IoT. Como exemplo, o protocolo ZigBee que foi desenvolvido para redes sem fio de baixo consumo energético faz a conexão entre dispositivos através do uso de chaves simétricas. Com base nisso, Olawmi et al. [27] demonstram alguns ataques relativamente simples a dispositivos em uma rede ZigBee que causam desde negação de serviço até tomada de controle de um dispositivo.

- Vulnerabilidades de software: ao explorar as vulnerabilidades de software, atacantes podem ganhar acesso a nós e dispositivos IoT. Na literatura encontramos diversos

estudos que apontam para problemas na gestão de dispositivos IoT [28, 29]. Alterar a senha padrão de dispositivos é uma prática de segurança negligenciada por muitos usuários e operadores de dispositivos. Com a IoT e a escala de dispositivos em utilização, o impacto de uma falha simples como essa pode ser enorme.

Também é importante considerar que soluções robustas de cibersegurança e validadas de redes de computadores convencionais foram desenvolvidas com outro paradigma tecnológico e outra escala de aplicação em mente. Redes IoT são compostas, em sua maioria, por dispositivos com pequeno poder computacional, normalmente com conexão sem fio e com baixo consumo energético, portanto, é inviável pensar nos dispositivos IoT como equipamentos que possam embarcar as tecnologias de cibersegurança complexas e que exijam uma robustez do dispositivo. Além disso, dispositivos IoT costumam ter alta disponibilidade e passam pouco tempo sob vigilância [30].

Com esse cenário em mente, pesquisas que promovam a evolução dos mecanismos de cibersegurança das redes e dispositivos IoT são fundamentais.

2.2 Aprendizado de fluxos contínuos de dados

Em algoritmos de aprendizado em lote, o treinamento para indução de um modelo de aprendizado é realizado sobre um conjunto de dados pré-definido. Em outras palavras, uma porção dos dados é separada e o algoritmo é aplicado sobre ela para que seja criado um modelo que represente aquele conjunto de dados estático. Os dados podem ser reutilizados várias vezes no processo de aprendizagem e costumam gerar modelos de aprendizado com uma alta precisão. Para atualizar o modelo de aprendizado não é possível apresentar apenas as novas observações para o algoritmo, já que, em aprendizado em lote, a atualização só será válida quando o processamento do lote inteiro terminar. Ou seja, para cenários que exigem uma resposta rápida a mudanças, como o deste estudo, não seria interessante trabalhar com essa premissa. [31]

Cenários como cidades inteligentes, cuidados de saúde e de monitoramento de redes IoT, como é o caso deste trabalho, precisam de respostas mais rápidas para mudanças de comportamento [32]. Em aplicações de fluxos contínuo de dados, não podemos definir um ponto inicial e final do fluxo de dados. Se observarmos o tráfego de pacotes de um dispositivo de rede, por exemplo, o tamanho potencial dessa série de valores é infinito. Enquanto o dispositivo estiver ligado, novos pacotes serão gerados e isso só irá cessar quando o equipamento for desligado ou interrompido.

O comportamento de um fluxo contínuo de dados é imprevisível e tende a mudar com o tempo. Em problemas dessa natureza, é interessante utilizar modelos de aprendizado que possam ser atualizados e modificados incrementalmente. Este trabalho, por exemplo, estuda fluxos de tráfego de redes IoT. O comportamento de dispositivos de rede

pode mudar com o tempo, e não é razoável esperar que um operador atualize o modelo de aprendizado para incorporar este novo comportamento [31, 32]. Por esses motivos, este trabalho irá utilizar técnicas de aprendizado contínuo e não de aprendizado em lotes.

2.3 Clusterização de fluxos contínuos de dados

Em problemas de clusterização, o objetivo é agrupar os dados disponíveis em grupos/classes de acordo com a similaridade entre os dados. Por exemplo, no monitoramento de redes, é possível observar o tamanho do pacote, porta de origem/destino e TTL (*Time to Live*) e criar clusters. Esses clusters não devem ser rígidos, pois o comportamento do tráfego dos dispositivos monitorados é dinâmico e pode sofrer alterações. Se um dos dispositivos dessa rede for uma câmera e algum parâmetro de configuração como resolução, taxa de quadros ou compressão dessa for alterado, o comportamento do dispositivo na rede pode mudar juntamente com as características dos clusters. Idealmente, o algoritmo de clusterização selecionado precisa conseguir reconhecer que o comportamento mudou e alterar os clusters. Esta mudança do modelo base se chama evolução e é esperada em clusterização de fluxos contínuos de dados [31, 32].

O *DenStream* é um algoritmo de clusterização para fluxos de dados contínuos proposto por Cao et al. [33]. O *DenStream* consegue formar clusters de formato arbitrário e é robusto com relação a dados que sigam um comportamento anômalo, como um outlier. Em aplicações ligadas ao monitoramento de redes, essas anomalias podem ser causadas, por exemplo, por interferências ambientais, falhas temporárias de dispositivos e problemas de rede (principalmente em dispositivos Wi-Fi). Alguns destaques do *DenStream*:

- O *DenStream* armazena representações estatísticas dos clusters, ao invés de armazenar as instâncias de dados que formam o cluster. Esta decisão permite que se armazene dados de fluxos de dados infinitos sem necessitar de espaços de armazenamento ilimitados.
- Introduz uma função que altera o peso/relevância de cada cluster em cada iteração. Clusters que recebem uma nova observação ganham peso e os outros perdem. Essa função determina a relevância de um cluster na janela de tempo atual e baseado em um valor de tolerância, o *DenStream* determina se o cluster será removido ou não do modelo. Isto abre espaço para a formação de novos clusters enquanto mantém a forma e precisão do modelo. Em seu artigo, Cao et al. chama esse processo de poda.
- Pela adaptabilidade do modelo e a capacidade de lidar com os outliers, a qualidade da clusterização do *DenStream* costuma ser alta.

Para compreender o funcionamento do *DenStream*, primeiro é preciso entender os diferentes tipos de clusters que o *DenStream* define. Trabalhando com o conceito de

micro-clusters, que são pequenas representações das observações, o *DenStream* define três categorias de clusters: *core micro-cluster*, *potential core-micro-cluster* e *outlier core-micro-cluster*. *Core micro-clusters* são as principais representações do comportamento do fluxo de dados. Cada *core micro-cluster* representa um cluster de raio ϵ e cobre a área que seria ocupada por cada ponto p que chega do fluxo de dados. Essa redução de n pontos p para um *core micro-cluster* que faça a representação estatística dos pontos é importante, pois em fluxo de dados contínuos precisamos considerar que não existe limite para a quantidade de observações. *Potential core-micro-cluster* e *outlier core-micro-cluster* são as estruturas dinâmicas que são geradas e mantidas incrementalmente na fase *online* do algoritmo. *Potential core-micro-clusters* são os cluster que indicam qual a tendência das observações, já os *outlier core-micro-clusters* são os *clusters* que diferem da tendência das atuais observações. Vale ressaltar que esses rótulos não são estáticos e, conforme o modelo evolui, os *micro-clusters* podem mudar de papel dependendo das características das observações que chegam.

Adicionalmente é necessário entender como os hiperparâmetros do algoritmo afetam a clusterização das observações. Na Tabela 1 é possível verificar uma síntese da descrição da função de cada um dos hiperparâmetros.

Tabela 1 – Descrição dos hiperparâmetros do *DenStream*.

λ - *Decaying factor*	Delimita a influência de dados historicos. λ mais alto indica uma menor influência dos dados históricos na clusterização
β - *Outlier tolerance factor*	Delimita a divisão entre *outlier core-micro-clusters* em relação aos *core micro-clusters*. β deve assumir (0,1]
μ - *Core weight threshold*	Delimita a divisão entre *outlier core-micro-clusters* em relação aos *core micro-clusters*. $\beta * \mu > 1$
ϵ - *Maximum radius of a micro-cluster*	Determina o raio máximo de uma vizinhança de *micro-clusters*
InitN - n_samples_init	Número de iterações utilizadas na inicialização dos clusters antes da fase *online*.
v - *Stream speed*	Velocidade do fluxo de dados ou quantidade de dados por unidade de tempo. Cada pacote representa um dado.

O *DenStream* é dividido em duas fases: fase *online* e fase *offline*. A fase *online*, ou fase de manutenção dos *micro-clusters*, começa após um número de iterações determinado pelo parâmetro n_{init} que faz uma clusterização inicial destas observações e forma os *micro-clusters* que servem de ponto de partida para o *DenStream*. Em sua estratégia de clusterização das novas observações (Algoritmo 1), o *DenStream* primeiro verifica as condições para agrupar a nova observação a um dos *potential core-micro-clusters* (c_p) existentes. Se incluir este ponto ao c_p não alterar o valor do raio r_p para um valor superior ao parâmetro ϵ, a nova observação p é agrupado em c_p. Caso a nova observação não esteja dentro dos parâmetros para se agrupar a um dos c_p existentes, o algoritmo verifica se as condições de clusterização são cumpridas pelo *outlier core-micro-cluster* (c_o) mais próximo. Se o novo p for agrupado ao c_o, uma verificação do peso w é feita e, caso satisfaça a condição $w > \beta\mu$, o c_o é removido e um novo c_p é criado na mesma posição.

Algorithm 1 - Agrupar (p)

```
1: Tentar agrupar p no p-micro-cluster c_p mais próximo;
2: if r_p (novo raio de c_p) ≤ ε then
3:     Adicionar p no c_p;
4: else
5:     Tentar adicionar p no o-micro-cluster c_0 mais próximo;
6:     if novo raio de c_0 ≤ ε then
7:         Adicionar p em c_o;
8:         if novo peso de c_o > βμ then
9:             Remover c_o do outlier-buffer e criar um novo p-micro-cluster em c_o
10:        end if
11:    else
12:        Criar um novo o-micro-cluster em p e inserir no outlier-buffer;
13:    end if
14: end if
```

Por último, caso p não satisfaça nenhuma das condições anteriores, um novo c_o é criado e p se torna a primeira observação deste novo *outlier core-micro-cluster*. Já a fase *offline* ocorre sob demanda do usuário. Nesta etapa, o *DenStream* utiliza uma variação do algoritmo de clusterização por densidade DBScan que irá conectar as áreas de densidades dos *potential core-micro-clusters* da fase *online* e gerar o *core micro-cluster* de formato arbitrário "final". Na realidade, os *core micro-clusters* são formados pela coleção de *potential core-micro-clusters* não redundantes e em tese, essa coleção deve cobrir toda a área que conteria as observações pontuais caso todas fossem guardadas em memória.

2.4 Sistemas de detecção de intrusão

Metodologias de detecção de intrusão são usadas para identificar e prevenir ataques cibernéticos em redes de computadores. Essas metodologias podem ser classificadas em duas categorias principais: detecção baseada em assinaturas (*Signature-based Detection* - SD) e Detecção baseada em anomalias (*Anomaly-based Detection* - AD). No contexto de detecção de intrusão, alguns ataques têm um padrão de comportamento bem conhecido, o que chamamos de assinatura. Por exemplo, um ataque de vírus pode ter uma assinatura que é detectada por um software antivírus. Nesse caso, um IDS baseado em assinatura monitora a rede em busca dessas assinaturas específicas para detectar a presença de um ataque conhecido. Por outro lado, a detecção baseada em anomalias (anomalias são desvios em relação ao comportamento de rede considerado normal por um modelo) procura identificar comportamentos fora do padrão que possam indicar um ataque. Nesse caso, um IDS baseado em anomalias monitora constantemente o tráfego de rede e compara o comportamento atual com o comportamento histórico para identificar anomalias. Esse método pode ser útil para detectar novos ataques ou variações de ataques que não possuem uma assinatura conhecida. Em resumo, IDS baseados em assinatura procuram

Tabela 2 – Vantagens e desvantagens das metodologias de detecção

Detecção baseada em assinatura - SD	**Detecção baseada em anomalia - AD**
Vantagens	
Método simples e eficiente em detectar novos ataques	Eficiente em detectar ameaças desconhecidas
Análise contextual detalhada e menos dependente de SO	Facilita as detecções de abuso de privilégio
Desvantagens	
Ineficaz para detectar ataques desconhecidos, ataques de evasão e variantes de ataques conhecidos.	Precisão do modelo mais fraca devido a eventos observados sendo constantemente alterados
Atualização do modelo depende de preparo prévio.	Indisponível durante a reconstrução de perfis de comportamento.
Dificuldade em manter as assinaturas/modelos atualizados.	Mais suscetível a atrasos na detecção

detectar ataques conhecidos, enquanto IDS baseados em anomalias procuram identificar comportamentos fora do padrão que possam indicar a presença de um ataque [34, 35].

Na categoria SD, o algoritmo recebe padrões de ataques conhecidos e é treinado para reconhecer rapidamente quando qualquer ataque com aquele mesmo padrão (assinaturas) aconteça. Frequentemente, esses algoritmos procuram por padrões de ataque no tráfego e estão completamente alheios ao comportamento normal do dispositivo que está sendo monitorado [36]. Algumas das vantagens de se utilizar detecção baseada em assinaturas é que a velocidade e a precisão de detecção de ataques costuma ser alta, além da resistência a falsos positivos em caso de mudanças no comportamento normal do dispositivo. Em contrapartida, sistemas do tipo SD têm dificuldade em detectar ataques que não foram apresentados previamente e lidar com variações de ataques já conhecidos por ele. Essa forma de detecção é muito dependente de um especialista que não seja apenas capaz tecnicamente de alimentar a base de assinaturas do IDS, como também precisa que este especialista esteja sempre atualizado sobre possíveis alterações ou novas vulnerabilidades. Para o cenário desse estudo, apesar dos bons resultados em outros casos, uma forma de detecção que dependa menos da intervenção humana precisa ser utilizada.

A outra opção já mencionada são os sistemas do tipo AD. Diferente da categoria SD, nos sistemas do tipo AD, o algoritmo é treinado para reconhecer o comportamento normal do dispositivo. Para isso, o modelo normalmente é treinado observando o tráfego do dispositivo por um período de tempo. Neste algoritmo, ao invés de procurar por um ataque e acusar a sua incidência, o tráfego dos dispositivos é observado e desvios dos valores considerados normais são apontados. Este tipo de IDS possui uma vantagem em ser melhor para acusar ataques desconhecidos, ou versões modificadas de ataques conhecidos. Porém, possui uma velocidade e precisão de detecção relativamente menor.

A Tabela 2 [37] apresenta um resumo das vantagens e desvantagens de cada método de detecção. Apesar das vantagens da velocidade de detecção e precisão de resposta apresentada pelos sistemas da categoria SD, com as restrições existentes neste estudo - usuário não especialista, comportamento de equipamentos heterogêneos, novas formas de ataque sendo desenvolvidas - foi escolhido trabalhar com o conceito AD.

2.5 Trabalhos relacionados

Nos últimos anos, diversos estudos foram desenvolvidos propondo soluções de detecção de ataques para IoT. Como discutido anteriormente, neste estudo, avaliamos o potencial de usar o *DenStream* para este mesmo fim. Nesta seção, serão discutidos trabalhos recentes que tenham algum ponto de convergência com o objeto deste estudo.

Lohiya e Thakkar [38] apresentam um extenso estudo sobre as perspectivas dos IDSs em IoT com uso de aprendizado de máquina, ou mais especificamente, aprendizado profundo. No artigo, Lohiya e Thakkar discutem vantagens e desvantagens das diferentes estratégias de posicionamento (centralizada, distribuída e híbrida), estratégias de detecção (baseadas em anomalia e baseadas em assinatura), tipos de ataque e técnicas de detecção de ataque em IoT. Os autores concluem seu trabalho apontando algumas preocupações e sugerindo algumas direções para trabalhos que se propõem a contribuir com o desenvolvimento de IDSs para IoT.

Em sua revisão, Silva et al. [39] criam uma taxonomia para facilitar o entendimento de alguns dos algoritmos de clusterização de fluxo contínuo de dados, classificando-os com base em aspectos importantes do problema de clusterização de fluxo contínuo de dados. Os sete aspectos são: estrutura de dados, modelo de janela temporal, mecanismo de detecção de outlier, número de parâmetros definidos pelo usuário, algoritmo de clusterização *offline*, formato dos clusters e tipo de problema de clusterização. Os autores apresentam uma extensa revisão dos 13 algoritmos considerados de maior relevância, segundo a taxonomia criada por eles. Entre eles, encontramos *BIRCH*, *CluStream*, *DenStream*, que é o algoritmo que utilizaremos neste estudo, entre outros. Na avaliação dos autores, boa parte dos algoritmos de clusterização de fluxos contínuos de dados ignora a detecção de mudança ou a evolução dos clusters. O *DenStream* lida bem com a evolução do fluxo de dados e apesar de alguns comentários dos autores sobre o custo computacional do algoritmo, pode ser uma boa escolha para a natureza do problema que estamos trabalhando neste estudo.

Anthi et al. [1] propuseram um IDS para dispositivos IoT residenciais baseado em aprendizado supervisionado em lote. O IDS proposto analisa o tráfego de rede pacote a pacote em três etapas. Na primeira etapa, um algoritmo de aprendizado de máquina é utilizado para identificar o dispositivo que está gerando ou recebendo o tráfego. Nesta etapa, cada dispositivo IoT é avaliado e um perfil de comportamento deles é traçado. Na próxima etapa, outro algoritmo supervisionado identifica se o tráfego que está chegando do dispositivo é benigno ou malicioso. Caso o pacote seja identificado como malicioso, um terceiro algoritmo entra em ação e tenta classificar o tipo de ataque.

Otoum et al. [40] sugerem uma solução de IDS híbrida (*Signature-based* e *anomaly-based*). Uma fraqueza desta estratégia apontada pelos autores é que a qualidade da classificação pode cair caso o volume de tráfego fique muito alto. Para mitigar este problema,

os autores sugerem complementar o AD IDS com o uso do algoritmo Deep Q. A solução proposta possui três fases. Na primeira fase, um filtro de tráfego verifica endereço IP (origem e destino), portas, protocolos e contagem de pacotes. Nesta etapa, pacotes anômalos são descartados e ataques básicos são bloqueados. Pacotes que passarem pelo filtro de tráfego seguem para a segunda etapa de pré-processamento, com sua dimensionalidade reduzida e normalização efetuada. A terceira fase aplica o IDS híbrido proposto pelos autores. Primeiro o SD-IDS procura por assinaturas de ataques conhecidos. Pacotes que passarem pela detecção de assinatura de ataques seguem para o AD-IDS que classifica o pacote em normal ou anômalo. Por último, pacotes considerados maliciosos são registrados e atualizam as assinaturas de ataques.

Em seu estudo, Otoum e Nayak [41] sugerem uma arquitetura de IDS baseada em Deep Learning. A abordagem conta com uma fase de pré-processamento para normalizar os valores e eliminar redundância dos dados. Em seguida as observações são encaminhadas para um algoritmo de extração de atributos (ao selecionar atributos mais relevantes, o tempo de aprendizado do classificador diminui), que são analisados para classificar as observações como normais ou anômalas. Neste estudo, quatro categorias de anomalias foram definidas: DoS, Probing, Remote to local (R2L) e User to root (U2R). O algoritmo proposto foi aplicado sobre o conjunto de dados públicos NSL-KDD, que é uma atualização do KDD'99. Os autores demonstram como a limpeza do conjunto de dados e a seleção de atributos mais relevantes aumentam a precisão e diminuem o tempo de treinamento se comparado com outros métodos.

A literatura também inclui trabalhos que propõem IDSs baseados em algoritmos de mineração de fluxos contínuos de dados. Scaranti et al. [42] propõem um IDS para redes definidas por software (*Software Defined Network - SDN*) baseado no *DenStream*. O modelo proposto, primeiramente, organiza o tráfego em fluxos IP. Na sequência, atributos dos fluxos são minerados pelo *DenStream*. Para detectar ataques, o modelo proposto define uma área no espaço de clusterização denominada área potencial. Clusters posicionados dentro dessa área são classificados como ataque. Somente ataques do tipo *scanning* e *Denial of Service* (DoS) foram avaliados.

Yin et al. [43] propuseram um novo algoritmo de clusterização de fluxos contínuos de dados para ser aplicado no problema de detecção de intrusões. O algoritmo proposto é uma variação de algoritmos existentes na literatura, como o próprio *DenStream*, e se concentra em clusterizar as observações de acordo com a densidade delas no espaço de busca e em estabelecer peso para as observações analisadas, que vai diminuindo conforme elas ficam mais antigas. Para detectar os ataques, os autores dividiram o problema em duas fases. Na primeira, o tráfego de rede é clusterizado para definir um perfil de comportamento normal, ou seja, exige-se que este tráfego não contenha pacotes maliciosos. A partir daí, começa a segunda fase, onde o tráfego é clusterizado e o resultado deste

processo é comparado com o que foi encontrado na primeira fase. Em caso de discrepância, emite-se o alerta de detecção do ataque. O perfil de comportamento normal é atualizado frequentemente para que o sistema se adapte às mudanças das observações. Os experimentos mostraram bons resultados, mas utilizaram um conjunto de dados bastante ultrapassado, o KDDCup 1999.

Aplicando técnicas de clusterização de fluxos contínuos de dados em um cenário similar a este trabalho, Nakagawa et al. [15] propõem o uso do algoritmo CluStream para clusterização dos pacotes de tráfego de rede e aplicação do teste de Page-Hinkley para detectar mudanças abruptas na série de valores monitorada. Diferente do *DenStream*, o CluStream possui um parâmetro para definir o número máximo de *micro-clusters* que serão mantidos na fase de manutenção. Em seu trabalho, Nakagawa et al. sugerem monitorar a distância entre os centroides dos *micro-clusters* para detectar a incidência de ataques.

A análise dos trabalhos relacionados mostra que a utilização de aprendizado supervisionado em lote ainda está bastante presente, como podemos ver nos trabalhos de Anthi et al. [1] e Lohiya e Thakkar [38]. Estudos que apostam na mineração de fluxos contínuos de dados também são encontrados, como podemos ver no trabalho de Silva et al. [39], mas guardam algumas diferenças com relação a este trabalho. Scaranti et al. [42] também utilizaram *DenStream*, mas analisaram fluxos IP em um ambiente diferente, as SDNs. Yin et al. [43] concentraram esforços em construir um novo algoritmo de clusterização, mas não definiram claramente qual tipo de rede ou tráfego iriam abordar. Outoum et al. [40] também trabalham com o ambiente de IDS para redes IoT, porém, com uma abordagem híbrida de IDS e baseado em assinatura e anomalias. Otoum e Nayak [41] sugerem a aplicação de um IDS baseado em Deep Learning focada em redes IoT e criam categorias para os tipos de ataque. Nakagawa et al. [15] focaram em redes IoT, mas utilizaram um algoritmo de clusterização diferente do aplicado neste trabalho.

Em geral, a maioria dos trabalhos analisados propôs algoritmos diferentes ou centrou-se em cenários diferentes. Os algoritmos de aprendizagem em lote têm uma baixa frequência de atualização do modelo e requerem grandes volumes de dados para treino, o que não é ideal em ambientes de redes domésticas. Além disso, os estudos revistos que partilhavam mais semelhanças com este em termos de escolhas de algoritmos não incluíam análises pacote a pacote ou não eram aplicados em ambientes IoT residenciais. Neste trabalho, exploramos um modelo de detecção baseado no algoritmo de clusterização de fluxos contínuos de dados *DenStream*. Esta abordagem não requer o armazenamento de observações, uma vez que o algoritmo escolhido aprende de forma incremental, e centra-se na análise pacote a pacote de dispositivos de rede IoT domésticos.

3 MATERIAIS E MÉTODOS

O principal objetivo deste trabalho é investigar a possibilidade de se utilizar o algoritmo de clusterização de fluxos contínuos de dados *DenStream* para detectar tentativas de ataques cibernéticos em pacotes de uma rede IoT. A maioria dos estudos de detecção de ataques são baseados em algoritmos de aprendizado em lote, como *Random Forest* [44] e *Support Vector Machine - (SVM)* [45]. Algoritmos desse tipo abordam o aprendizado de máquina como um processo estático. Primeiro eles acumulam uma grande quantidade de observações e depois analisam as observações para induzir um modelo. A atualização desses modelos costuma ter um alto custo já que envolvem a repetição do processo mencionado anteriormente [46]. Para análise de tráfego de redes de computadores, é necessário desenvolver soluções mais dinâmicas. Como o comportamento de uma rede muda com o tempo, o algoritmo precisa ser capaz de aprender incrementalmente ao mesmo tempo que as observações chegam. Além disso, fluxos contínuos de dados são potencialmente infinitos e armazenar estes dados por um período extenso não é uma tarefa fácil. Algoritmos de mineração de fluxos contínuos de dados estão mais preparados para lidar com este tipo de problema, visto que são capazes de aprender de forma incremental e não exigem que observações sejam armazenadas e revisadas durante as análises.

É importante também comentar sobre o uso de aprendizado supervisionado em modelos de detecção de ataques. O aprendizado supervisionado necessita de observações rotuladas para o processo de treinamento do modelo. Sendo assim, em casos de detecção de ataques, é necessário apresentar para o algoritmo exemplos de comportamento normal e malicioso. Considerando a grande diversidade de ataques e o volume de dados que temos quando analisamos tráfego de redes, preparar conjuntos devidamente rotulados é um grande desafio. Uma possível alternativa é o uso de algoritmos de clusterização, categorizados como aprendizado não supervisionado. Neste caso, o algoritmo tenta criar clusters reunindo dados que possuem maior similaridade, como o caso do *DenStream* apresentado previamente.

Algoritmos de clusterização apenas organizam dados similares em *clusters*, mas não rotulam ou classificam estes dados. Portanto, para detectar ataques, é necessário criar mecanismos adicionais que sejam capazes de identificar quais clusters correspondem a observações normais ou maliciosas. Neste trabalho, nós procuramos por indicadores no comportamento dos *micro-clusters* criados em tempo real que demonstrem variações que possam ser detectadas no momento em que ataques comecem. Em boa parte, nós iremos analisar um indicador: a distância entre os centros dos diferentes tipos de *micro-clusters* produzidos pelo *DenStream*.

A Figura 1 ilustra os passos do experimento conduzido e os detalhes serão apresen-

tados nas seções seguintes. O experimento será centrado na análise dos pacotes coletados do tráfego de rede. No processo proposto na Figura 1, cada observação representa um pacote de rede. Esses pacotes são separados conforme o dispositivo de origem/destino e os protocolos que estão logo acima do protocolo IP na estrutura de camadas do TCP/IP: ICMP, TCP ou UDP. A ideia por trás dessa separação é organizar as observações antes de apresentá-las ao algoritmo de clusterização. Se os pacotes coletados em uma rede forem entregues indiscriminadamente para o algoritmo, a tendência é a criação de muitos clusters representando múltiplos dispositivos e protocolos. Isso provavelmente dificultaria a identificação de quais clusters representam comportamento normal e malicioso. Ao separar inicialmente os pacotes conforme o dispositivo e o protocolo, podemos executar uma instância do algoritmo de clusterização para cada sub-grupo. Desta forma, permitimos que as diferentes instâncias do algoritmo se moldem a comportamentos e eventos que são mais específicos para cada dispositivo e protocolo.

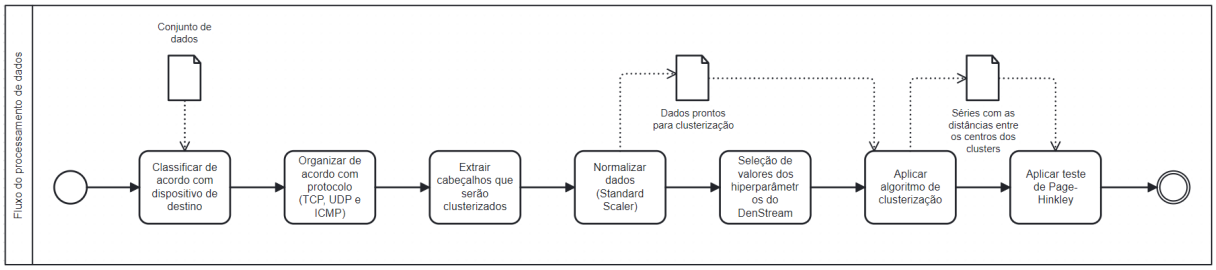

Figura 1 – Diagrama do experimento proposto.

3.1 Pré-processamento

O primeiro passo do pré-processamento é preparar o fluxo de dados que serão clusterizados pelas instâncias do *DenStream*. Consideramos um cenário onde pacotes de rede são coletados por um dispositivo ou software nos pontos de interconexão entre a rede local e a Internet. Em seguida, esses pacotes são classificados conforme o dispositivo de destino deles. Além da separação dos fluxos de dados por dispositivo, estes pacotes ainda são organizados em três grupos: um para pacotes TCP, outro para pacotes UDP e um terceiro para pacotes ICMP. Com os pacotes organizados por dispositivo e protocolo, são extraídos os cabeçalhos que melhor podem indicar a ocorrência de diferentes tipos de ataques, de acordo com Nakagawa et al.[15] e Anthi et al.[1]. Os campos selecionados são apresentados na Tabela 3. O campo *frame_len*, presente em todos os pacotes, e os campos *ip_flags*, *ip_flags_df*, *ip_flags_mf*, *ip_frag_offset* e *ip_ttl*, sendo parte do cabeçalho IP, foram selecionados para compor todas as observações, independente de serem derivadas de pacotes TCP, UDP ou ICMP. Os campos *tcp_dstport*, *tcp_flags_syn*, *tcp_flags_ack* e *tcp_flags_push* foram selecionados para compor as observações que representam os fluxos

Tabela 3 – Descrição dos atributos selecionados.

Atributo	**Descrição**
frame_len	Tamanho do pacote
ip_flags	*Flags* do controle de fragmentação do pacote
ip_flags_df	*Flag Don't Fragment* que indica para o intermediário não fragmentar este pacote
ip_flags_mf	*Flag More Fragments* que indica se este é o último fragmento de uma sequência
ip_frag_offset	Indica onde este fragmento deve ser inserido em relação ao fragmento original
ip_ttl	*Time to Live*
tcp_dstport	Porta de destino TCP
tcp_flags_syn	*Flag* de sincronização do *Three-way handshake*
tcp_flags_ack	*Flag* de reconhecimento do *Three-way handshake*
tcp_flags_push	*Flag* que indica se o pacote deve entrar em buffer antes de ser entregue a aplicação
udp_dstport	Porta de destino UDP
icmp_code	Tipo de pacote ICMP

TCP, enquanto o campo *udp_dstport* foi selecionado para compor os fluxos UDP. Para as observações ICMP, o campo *icmp_code* foi selecionado. Ao final dessa etapa, teremos conjuntos de dados compostos de três fluxos de dados (TCP, UDP e ICMP) para cada um dos dispositivos.

Após preparar os fluxos de dados para o experimento, será executada a normalização. Os atributos dos conjuntos de dados estão em diferentes ordens de grandeza. O campo *frame_len*, por exemplo, costuma assumir valores maiores do que 1000, enquanto os campos de *flags* assumem valores binários. Isso pode influenciar no cálculo da distância entre as observações e consequentemente na forma que são clusterizadas. Para normalizar os dados, optamos por utilizar o método *Standard Scaler*, que os transforma para terem média igual a zero e variância igual a 1. A implementação utilizada foi da biblioteca *River*[1] em linguagem *Python*.

Finalmente, com os fluxos de dados prontos, vamos para a etapa de seleção dos valores de hiperparâmetros do *DenStream* que serão testados. Na Tabela 1, a descrição de cada hiperparâmetro é apresentada. Mudanças nesses hiperparâmetros alteram drasticamente a forma que o algoritmo clusteriza os fluxos de dados que chegam. Dependendo dos valores dos hiperparâmetros, clusters de diferentes formatos e tamanhos são criados, além de influenciar nos eventos de criação e fusão de clusters. Adicionalmente, os hiperparâmetros também definem quando clusters mais antigos ou com menor peso devem ser removidos. Tudo isso influencia diretamente na métrica que desejamos observar: distância entre os clusters. Portanto, é importante selecionar uma faixa abrangente de valores que nos permita explorar o potencial do *DenStream* para os conjuntos de dados que iremos analisar. Por outro lado, uma faixa grande demais tende a dificultar o experimento, já que temos uma quantidade muito grande de combinações se forem considerados todos os cenários para todos os hiperparâmetros. Como ponto de partida para selecionar nossa faixa de valores, nos baseamos no estudo de Scaranti et al. [42]. Depois expandimos as

[1] https://riverml.xyz/0.15.0/api/preprocessing/StandardScaler/

faixas utilizadas e definimos uma faixa de valores que pode ser vista na Tabela 4.

Tabela 4 – Faixa de valores dos hiperparâmetros do *DenStream*.

λ - *Decaying factor*	[0.01, 0.05, 0.10, 0.25, 0.50, 0.75, 1.00]
β - *Outlier tolerance factor*	[0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 1.00]
μ - *Core weight threshold*	[10,100,1000,1500, 2000]
ϵ - *Maximum radius of a micro-cluster*	[0.02, 0.05, 0.10, 0.25, 0.5, 0.75, 1.00, 5.00]
InitN - n_samples_init	[1000]
v - *Stream speed*	[1]

3.2 Mineração de fluxos contínuos de dados

Com os conjuntos de dados prontos e a faixa de valores de hiperparâmetros selecionada, o experimento segue para sua etapa principal: execução de instâncias do algoritmo *DenStream* para que seja analisado o comportamento dos clusters durante a ocorrência de ataques.

Para estudar como a distância entre clusters pode se relacionar com a ocorrência de ataques, foram desenvolvidas duas métricas baseadas na distância Euclidiana. Para ambas as métricas, primeiro, precisamos construir uma matriz que calcula a distância Euclidiana entre os *micro-clusters* existentes em cada iteração. Precisamente, os clusters são analisados em pares e para cada par a distância Euclidiana entre os centros destes clusters é calculada. Suponha que em uma iteração o número de clusters seja n. Uma matriz D com dimensão $n \times n$ vai ser criada. Cada posição i, j da matriz irá armazenar a distância Euclidiana entre os centros dos clusters i e j. Desta matriz, vamos extrair duas métricas: a média das distâncias e a distância máxima. Considerando estas duas métricas e os diferentes tipos de *micro-clusters*, as seguintes séries foram avaliadas:

- Distância máxima entre *potential core-micro-clusters*: esta série contém a maior distância Euclidiana entre os centros dos *potential core-micro-clusters* em cada iteração.
- Distância média entre *potential core-micro-clusters*: esta série contém a distância Euclidiana média entre os centros dos *potential core-micro-clusters* em cada iteração.
- Distância máxima entre *outlier core-micro-clusters*: esta série contém a maior distância Euclidiana entre os centros dos *outlier core-micro-clusters* em cada iteração.
- Distância média entre *outlier core-micro-clusters*: esta série contém a distância Euclidiana média entre os centros dos *outlier core-micro-clusters* em cada iteração.
- Distância máxima considerando potential e *outlier core-micro-clusters*: esta série contém a maior distância Euclidiana entre os centros de ambos os tipos de *micro-cluster* em cada iteração.

- Distância média considerando potential e *outlier core-micro-clusters*: esta série contem a distância Euclidiana média entre os centros de ambos os tipos de *micro-cluster* em cada iteração.

Essas séries foram calculadas e armazenadas para cada combinação de valores de hiperparâmetros do *DenStream*.

3.3 Teste de Page-Hinkley

Proposto por Alan Page e Lawrence Hinkley em 1954 [47], o teste de Page-Hinkley é uma técnica estatística muito utilizada em análise sequencial para detectar mudanças súbitas e inesperadas em dados monitorados. É baseado no conceito de "soma cumulativa", que é uma forma de acompanhar o comportamento de dados sobre o tempo e calcular a soma dos desvios entre valores observados e esperados. A aplicação prática do teste de Page-Hinkley envolve calcular a soma cumulativa dos desvios entre os dados observados e os valores esperados. Essa soma é comparada a um limiar pré-definido e quando esse limiar é excedido, uma mudança é detectada e considerada estatisticamente significativa. O resultado é um alerta indicando a mudança de valores da série.

O teste de Page-Hinkley possui alguns parâmetros-chave que afetam seu desempenho e sensibilidade na detecção de mudanças em séries temporais. O parâmetro mais crítico é o *threshold* (λ), que determina a sensibilidade do teste, ou seja, quão grande uma mudança deve ser para ser considerada significativa. Outro parâmetro importante é o fator de esquecimento (α), que controla a velocidade com que a estatística acumulativa é atualizada ao longo do tempo, influenciando na sensibilidade do teste às mudanças recentes em relação as mudanças antigas na série. Além disso, o Teste de Page-Hinkley também possui um parâmetro de sensibilidade (δ), que ajuda a evitar falsos positivos ao ajustar a sensibilidade do teste conforme a série progride. Neste trabalho, exploramos apenas se a variação do parâmetro *threshold* seria o suficiente para causar alterações das detecções. Optamos por manter os outros parâmetros com os valores padrões para simplificar a análise final dos resultados.

No contexto deste trabalho, o teste de Page-Hinkley foi utilizado para detectar automaticamente mudanças nas séries formadas pelas métricas que estão sendo analisadas, visando determinar se elas podem ser consideradas indicadores de ataques. A ideia é comparar as mudanças detectadas pelo teste de Page-Hinkley com os pontos onde os ataques foram observados. Cenários em que a maioria das mudanças detectadas se alinhar com os ataques sugere que a métrica tem o potencial de ser um indicador de ataque. Para analisar estes cenários objetivamente, coletamos o número de alertas verdadeiros e alertas falsos para cada caso e o atraso para sinalizar o ataque. Na sequência, são selecionados os cenários melhores ranqueados sob o critério estabelecido anteriormente. A Tabela 5 mostra

Tabela 5 – Valores de hiperparâmetros para o teste de Page-Hinkley

Nome do parâmetro	Descrição	Valores
min_instances	O número inicial de instâncias antes de o teste começar a detectar mudanças.	[1000]
threshold	O limiar da detecção de mudança (λ)	[20, 50, 100, 150, 200]
α	O fator de esquecimento, utilizado para balancear o valor observado e a média.	[0.9999]
δ	Controla a sensibilidade do limiar de detecção do teste.	[0.005]

os valores de hiperparâmetros utilizados pelo teste de Page-Hinkey nos experimentos. A implementação do método utilizada neste trabalho é da biblioteca *River*[2],que utiliza a linguagem de programação *Python*. As experiências foram realizadas num computador com um processador Intel Core i5-8600k e 16 GB de RAM DDR4 2400MHz.

Em resumo, o teste foi aplicado a todas as séries contendo métricas derivadas da distância Euclidiana entre os centros dos *micro-clusters*, apresentadas na Seção 3.2. Sobre elas, foi aplicado o teste de Page-Hinkley com diferentes valores de *threshold*. Para cada *threshold*, as seguintes métricas foram calculadas:

- Total de alertas: representa a quantidade de vezes que o teste de Page-Hinkley indicou uma mudança na série de valores;
- Total de verdadeiros positivos: quantidade de vezes que um alerta aconteceu em um momento que um ataque estava ativo;
- Total de falsos positivos: quantidade de vezes que um alerta aconteceu em um momento em que não havia ataque ativo;
- Atraso: atraso entre o início de um ataque e o alerta gerado pelo teste de Page-Hinkley.

3.4 Conjuntos de dados

Para realizar os experimentos, foram utilizados dois conjuntos de dados. O primeiro foi gerado e disponibilizado publicamente por Anthi et al. [1] e o segundo, que é uma modificação do conjunto de dados gerados por Anthi et al., foi disponibilizado por Nakagawa et al. [15]. Para gerar o seu conjunto de dados, Anthi et al. criaram um ambiente controlado com dispositivos IoT conectados à Internet. Diferentes ataques foram realizados contra estes dispositivos IoT e os pacotes de rede foram coletados durante o experimento. Os pesquisadores disponibilizaram o conjunto de dados com todos os cabeçalhos dos pacotes coletados, exceto os endereços IP, que foram omitidos. Todos os pacotes foram rotulados de acordo com os ataques e a origem ou destino dos pacotes. Os dispositivos utilizados neste conjunto de dados estão listados na Tabela 6. Os seguintes ataques foram feitos aos dispositivos:

[2] https://riverml.xyz/0.18.0/api/drift/PageHinkley/

- Man-in-the-middle: ataque onde o atacante se posiciona entre o transmissor e o receptor e intercepta as informações de uma comunicação. Uma forma de criar um ataque deste tipo seria o atacante se aproveitar de alguma fraqueza nos diversos e heterogêneos dispositivos IoT em uma residência e através deles ganhar acesso à rede e monitorar o resto do tráfego interno, potencialmente coletando informações privadas e/ou confidenciais.

- Negação de Serviço (DoS): ataque onde um dispositivo é forçado a uma situação onde ele fica inapto a executar um serviço. Esse tipo de ataque satura o dispositivo com tantas requisições que ele começa a negar novas requisições legítimas devido à sua falta de capacidade de lidar com elas.

- *Scanning*: neste ataque, o atacante escaneia a rede para coletar informações sobre os potenciais alvos. Utilizando esta técnica, o atacante pode verificar quais dispositivos e serviços estão ativos na rede monitorada para explorar potenciais vulnerabilidades nas aplicações que estão ativas.

Tabela 6 – Dispositivos IoT e protocolos presentes nos conjuntos de dados.

Dispositivo IoT	Protocolos
TP-Link NC200 (câmera)	WiFi
Hive Hub	Ethernet & ZigBee
Lifx Smart Lamp	WiFi & ZigBee
Samsung Smart Things Hub	Ethernet & BLE
TP-Link SmartPlug	WiFi

Para realizar o primeiro experimento, nós processamos o conjunto de dados de Anthi et al. para que os dados fiquem disponíveis no formato estabelecido na seção 3.1, o que significa que os pacotes foram separados conforme o dispositivo e protocolo (TCP, UDP ou ICMP) e apenas os campos do cabeçalho que eram de interesse para o nosso experimento foram extraídos. A Tabela 7 demonstra os detalhes para cada um dos cenários do primeiro conjunto de dados que foram preparados para o experimento, considerando os diferentes dispositivos e protocolos. Todos os pacotes de cada cenário foram ordenados pelo *timestamp* para formar as séries e as colunas “Início” e “Fim” da Tabela indicam em que posição da série cada ataque começou e terminou.

Para o segundo experimento, foram seguidos os mesmos procedimentos do primeiro experimento, com a diferença que foi utilizado o conjunto de dados disponibilizado por Nakagawa et al. Como é possível observar na Tabela 7, o conjunto de dados disponibilizado por Anthi et al. tem os ataques acontecendo em sequência, sem que haja momentos de comportamento normal entre eles. Consequentemente, não conseguimos avaliar como o algoritmo se comporta quando os ataques são intercalados com o comportamento normal da rede, o que é importante, pois nossa proposta se baseia em aprendizado incremental.

Tabela 7 – Detalhes referentes ao número de pacotes, tipos de ataque e início e fim dos ataques do conjunto de dados de Anthi et al., utilizado no primeiro experimento.

Dispositivo	**Protocolo**	**Tamanho**	**Tipo de ataque**	**Início**	**Fim**
TPLink Cam	TCP	22376	*Scanning*	9611	18141
			DoS	18142	22322
			MITM	22323	22375
	UDP	7626	*Scanning*	6161	6470
			DoS	6471	7552
			MITM	7553	7625
Hive	TCP	47905	*Scanning*	13128	33777
			DoS	33778	47298
			MITM	47299	47905
	ICMP	2710	DoS	2363	2559
			MITM	2559	2710
LifX	TCP	12513	DoS	2883	12338
			MITM	12339	12513
	ICMP	13127	MITM	12857	13127
Smart Things	TCP	42180	*Scanning*	14618	33567
			DoS	33568	41519
			MITM	41520	42180
TPLink Plug	TCP	14142	DoS	13909	14142
	UDP	10898	DoS	1863	2465
			iot-toolkit	2466	10898

Para permitir essa análise, Nakagawa et al. modificaram o conjunto de dados de Anthi et al. intercalando intervalos de comportamento normal com intervalos de comportamento malicioso. Para tanto, os pesquisadores identificaram o início e fim de cada ataque através dos rótulos, e através dos *timestamps* identificaram quando um fluxo de dados no dispositivo foi iniciado ou interrompido. O uso destas referências é importante, pois seria prejudicial para o experimento se um fluxo contínuo fosse cortado ao meio e perdesse as características típicas do funcionamento de um dispositivo real. Desta forma, os pesquisadores identificaram os períodos autocontidos de tráfego normal e malicioso e os separaram em "recortes". Com estes recortes, foram montados cenários onde fluxos normais e maliciosos foram intercalados e, sobre estes cenários, executamos o segundo experimento. Os detalhes dos tamanhos dos conjuntos, tipos de ataques e momento em que os ataques aconteceram estão na Tabela 8.

Tabela 8 – Detalhes referentes ao número de pacotes, tipos de ataque e início e fim dos ataques do conjunto de dados de Nakagawa et al., utilizado no segundo experimento.

Dispositivo	**Protocolo**	**Tamanho**	**Tipo de ataque**	**Início**	**Fim**
TPLink Cam	TCP	39849	DoS	12990	17170
			MITM	21337	21552
			Scanning	27153	35683
	UDP	8550	DoS	3130	4211
			MITM	5232	5450
			Scanning	6546	7162
Hive	TCP	192920	DoS	67203	80723
			MITM	112088	112694
			Scanning	154275	174924
	ICMP	36249	DoS	18371	18759
			MITM	26629	26932
LifX	TCP	64200	DoS	28582	38037
			MITM	50857	51381
	ICMP	89074	MITM	64115	64654
Smart Things	TCP	122834	DoS	42819	48770
			MITM	64190	64850
			Scanning	86466	107415
TPLink Plug	TCP	8897	DoS	7154	7855
	UDP	12478	DoS	1868	3073
			iot-toolkit	3560	11992

4 RESULTADOS

Este capítulo irá apresentar os resultados obtidos com a execução dos experimentos descritos no Capítulo 3. O fluxo de etapas apresentado na Figura 1 será executado para dois conjuntos de dados distintos. Para o experimento que identificamos como "Experimento 1", utilizamos o conjunto de dados gerado por Anthi et al., onde cada cenário sempre tem um período inicial de tráfego normal seguido por vários ataques, sem que haja mais nenhum período de tráfego normal entre eles. Para o experimento identificado como "Experimento 2", utilizamos o conjunto de dados gerado por Nakagawa et al.. A principal diferença deste conjunto de dados para o primeiro é que cada ataque sempre é precedido por um período de tráfego normal, o que possibilitou a análise de diversas transições entre períodos de tráfego normal e de tráfego malicioso.

O capítulo primeiramente apresenta as combinações de hiperparâmetros que tiveram os melhores resultados em cada cenário para os experimentos 1 e 2, considerando os alertas gerados pelo teste de Page-Hinkley. Em seguida, serão analisados, de forma mais detalhada, os cenários com os melhores resultados para verificar como as séries derivadas das distâncias Euclidianas entre os *micro-clusters* se comportaram quando houve incidência de ataques.

4.1 Análise dos valores de hiperparâmetros do *DenStream*

Com todos os cenários disponíveis, a primeira análise verificou se os valores de hiperparâmetros que culminaram nos melhores resultados de clusterização e alertas tinham alguma similaridade. Isto poderia apontar se seria possível encontrar combinações que pudessem ser utilizadas para diferentes dispositivos e protocolos em uma rede. Para isto, ranqueamos as combinações de hiperparâmetros para cada cenário conforme o número de falsos positivos, de verdadeiros positivos e o atraso nas detecções. Então, selecionamos as três melhores combinações de hiperparâmetros para cada par de dispositivo e protocolo.

No experimento todas as combinações ranqueadas apresentadas nessa seção estão relacionadas aos resultados dos *potential core-micro-clusters*, pois a série gerada com a observação desta métrica é a que apresentou os melhores resultados. Referente ao experimento 1, a Tabela 9 nos mostra que o valor dos hiperparâmetros tendem a ser similares dentro de cada um dos cenários específicos, porém, ao considerar todos os cenários de diferentes dispositivos e protocolos, nenhum valor se destacou. Isso sugere que as combinações de hiperparâmetros mais otimizadas para cada dispositivo e cenário são diferentes e é importante se atentar aos valores selecionados para a fase de clusterização do *DenStream*. Ainda no primeiro experimento, outro resultado a se destacar é que, exceto pelo

Tabela 9 – Melhores resultados das séries de Distâncias Euclidianas máximas entre *potential core-micro-clusters* (Experimento 1).

Conjuntos de dados		**Hiperparâmetros *DenStream***				**Parâmetros Page-Hinkley**	**Métricas teste de Page-Hinkley**			
Dispositivo	**Protocolo**	λ	β	μ	ϵ	**Threshold**	**Total alertas**	**Total verdadeiro positivo**	**Total falso positivo**	**Atraso**
TPLinkCam	**TCP**	0,25	0,4	10	0,02	200	1	1	0	49
		0,25	0,4	10	0,05	200	1	1	0	49
		0,25	0,4	10	0,1	200	1	1	0	49
	UDP	0,01	0,1	1000	0,02	1	2	1	1	56
		0,01	0,1	1000	0,05	1	2	1	1	56
		0,01	0,1	1000	0,1	1	2	1	1	56
Hive	**TCP**	0,1	0,2	100	0,02	200	2	1	1	282
		0,1	0,2	100	0,05	200	2	1	1	282
		0,1	0,3	100	0,02	200	2	1	1	276
	ICMP	0,01	0,01	1000	0,02	0,001	1	1	0	17
		0,01	0,01	1000	0,02	0,5	1	1	0	17
		0,01	0,01	1000	0,02	1	1	1	0	17
Lifx	**TCP**	0,1	0,3	100	0,25	50	1	1	0	57
		0,1	0,3	100	0,5	50	1	1	0	57
		0,1	0,3	100	0,75	50	1	1	0	57
	ICMP	0,01	0,01	1000	1	0,001	1	1	0	135
		0,01	0,01	1000	1	0,5	1	1	0	135
		0,01	0,01	1000	1	1	1	1	0	135
SmartThings	**TCP**	0,25	0,4	100	0,02	150	2	1	1	83
		0,25	0,4	100	0,05	150	2	1	1	83
		0,25	0,4	100	0,1	150	2	1	1	83
TPLinkPlug	**TCP**	0,25	0,1	100	0,02	5	1	1	0	34
		0,25	0,1	100	0,05	5	1	1	0	34
		0,25	0,1	100	0,1	5	1	1	0	34
	UDP	0,01	0,2	100	0,02	0,5	1	1	0	139
		0,01	0,2	100	0,02	1	1	1	0	139
		0,01	0,2	100	0,02	5	1	1	0	139

cenário TP-Link Plug UDP, nos três resultados do topo do ranqueamento de cada cenário, o parâmetro ϵ é o que teve alteração. Este é o hiperparêmetro que define o raio máximo da vizinhança de *micro-clusters* e esse resultado sugere que variar o seu valor, até certo ponto, não afeta o comportamento de clusterização destas séries.

Para o experimento 2, com o intuito de manter a coerência entre as análises, também iremos apresentar os resultados relacionados às séries dos *potential core-micro-clusters*. Reforçando os resultados obtidos com o experimento 1, na Tabela 10 podemos observar que, em cada cenário, os valores dos hiperparâmetros seguem com a tendência de serem similares e, quando variam, o hiperparâmetro que varia é o ϵ. Também podemos observar que, apesar dos equipamentos e protocolos serem os mesmos para os dois experimentos, os melhores valores para os hiperparâmetros não são iguais entre eles. Isto reforça que é necessário ter uma atenção especial com quais valores serão selecionados para os hiperparâmetros na aplicação deste método.

4.2 Análise do comportamento das séries

4.2.1 Experimento 1

Considerando apenas as combinações de valores de hiperparâmetros que levaram aos três melhores resultados para cada cenário, o teste de Page-Hinkley foi capaz de detectar os ataques para todos os dispositivos e todos os protocolos e os atrasos na detecção

Tabela 10 – Melhores resultados das séries de Distâncias Euclidianas máximas entre *potential core-micro-clusters* (Experimento 2).

Conjuntos de dados		**Hiperparâmetros *DenStream***				**Parâmetros Page-Hinkley**	**Métricas teste de Page-Hinkley**			
Dispositivo	**Protocolo**	λ	β	μ	ϵ	**Threshold**	**Total alertas**	**Total verdadeiro positivo**	**Total falso positivo**	**Atraso**
TPLinkCam	**TCP**	0,1	0,4	10	0,5	0,001	8	4	4	3
		0,1	0,4	10	0,5	0,5	8	4	4	3
		0,1	0,4	10	0,5	1	8	4	4	4
	UDP	0,05	0,2	10	1	20	6	6	0	30
		0,05	0,2	10	1	50	6	6	0	55
		0,05	0,2	10	1	100	6	6	0	76
Hive	**TCP**	0,05	0,5	10	0,1	20	8	4	4	221
		0,05	0,5	10	0,25	150	8	4	4	263
		0,05	0,5	10	0,25	200	8	4	4	304
	ICMP	0,01	0,01	1000	0,02	1	4	4	0	11
		0,01	0,01	1000	0,02	5	4	4	0	11
		0,01	0,01	1000	0,02	10	4	4	0	11
Lifx	**TCP**	0,05	0,1	100	0,02	0,5	3	2	1	115
		0,05	0,1	100	0,02	1	3	2	1	160
		0,05	0,1	100	0,02	5	3	2	1	205
	ICMP	0,25	0,01	2000	0,02	1	1	1	0	20
		0,25	0,01	2000	0,02	5	1	1	0	20
		0,25	0,01	2000	0,02	10	1	1	0	20
SmartThings	**TCP**	0,01	0,01	1500	0,25	20	11	5	6	17
		0,01	0,01	1500	0,5	0,001	11	4	7	61
		0,01	0,01	1500	0,5	0,5	11	4	7	61
TPLinkPlug	**TCP**	1	0,4	10	0,02	50	1	1	0	66
		1	0,4	10	0,05	50	1	1	0	66
		1	0,4	10	0,1	50	1	1	0	67
	UDP	0,01	0,01	1000	0,25	1	6	4	2	171
		0,01	0,01	1000	0,25	5	6	4	2	173
		0,01	0,01	1000	0,25	10	6	4	2	175

foram relativamente baixos. O menor atraso foi de 17 iterações, onde cada iteração representa um pacote de dados, encontrado no cenário do dispositivo Hive protocolo ICMP, enquanto o maior valor de atraso foi 282 iterações, encontrado no cenário Hive TCP. Isso indica que a distância Euclidiana máxima entre os *potential core-micro-clusters* poderia ser utilizada para detectar os ataques nestes dispositivos. A Figura 2 e a Figura 3 ilustram essas situações. As áreas coloridas indicam intervalos nos quais ataques estavam ocorrendo e os marcadores azuis indicam em qual iteração o teste de Page-Hinkley detectou uma mudança na série.

A Figura 2 apresenta o cenário do dispositivo Lifx e protocolo TCP, onde um ataque de negação de serviço se inicia na iteração 2883 e se estende até a iteração 12338. Em seguida, um ataque do tipo *man-in-the-middle* é introduzido e se estende até o final do cenário. No gráfico, podemos notar que, durante o período anterior ao primeiro ataque, a série monitorada assume valores estáveis. Quando o primeiro ataque se inicia, a série demonstra uma queda aguda, detectada pelo teste de Page-Hinkley. A transição entre os ataques não foi detectada.

A Figura 3 mostra o cenário para o dispositivo TP-Link Cam e o protocolo TCP, onde um ataque de *scanning* foi iniciado na iteração 9611 e durou até a iteração 18141, sendo imediatamente sucedido por um ataque de negação de serviço que se estendeu até a iteração 22322. Na sequência desse último, houve um ataque *man-in-the-middle* que durou até o encerramento do cenário. No gráfico, nota-se que, durante o período que antecedeu

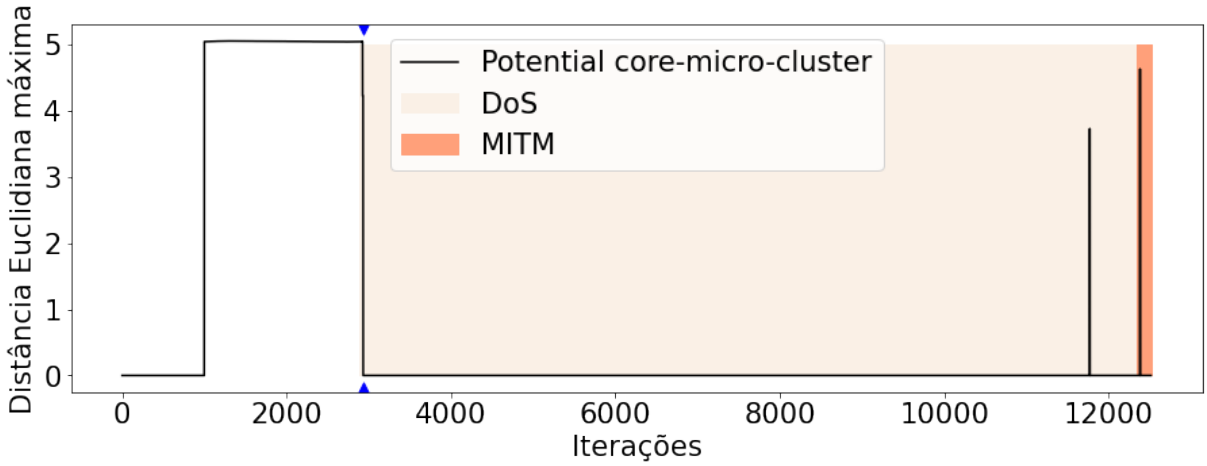

Figura 2 – Distância máxima entre *potential core-micro-clusters* para o dispositivo Lifx e protocolo TCP. Parâmetros: λ - 0,1 / β - 0,3 / μ - 100 / ϵ - 0,25 / v - 1

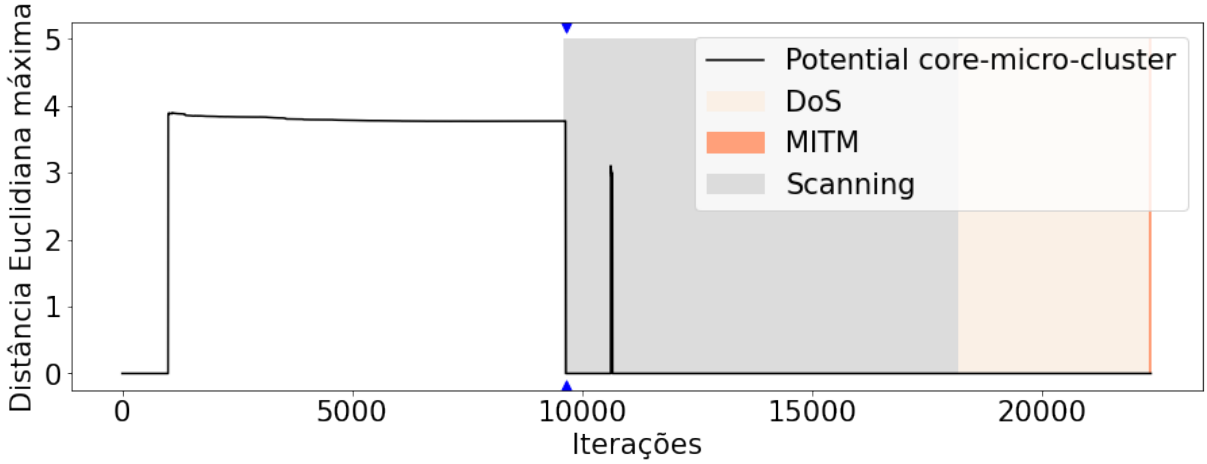

Figura 3 – Distância máxima entre *potential core-micro-clusters* para o dispositivo Tp-Link Cam protocolo TCP. Parâmetros: λ - 0,25 / β - 0,4 / μ - 10 / ϵ - 0,1 / v - 1

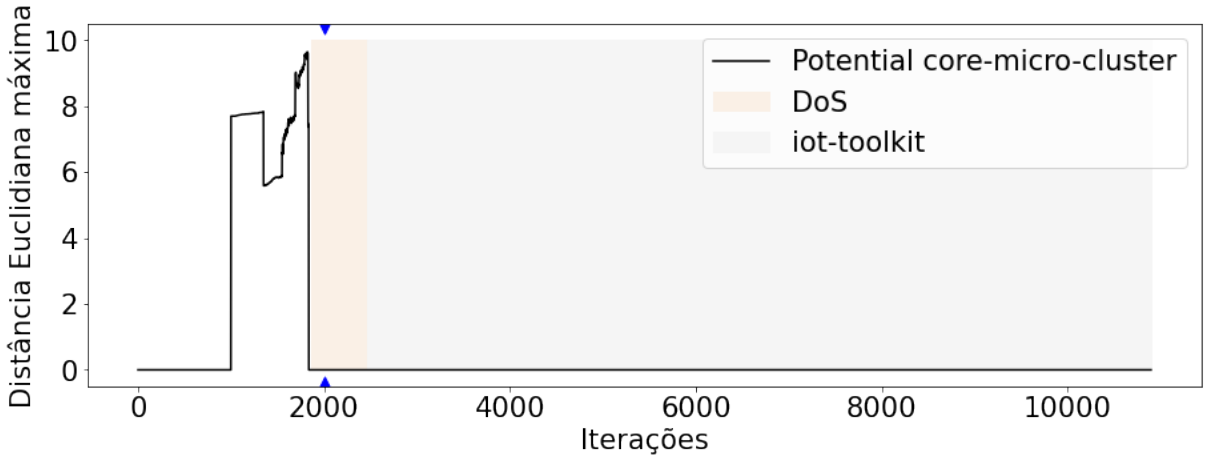

Figura 4 – Distância máxima dos *potential core-micro-clusters* para o dispositivo Tp-Link Plug protocolo UDP. Parâmetros: λ - 0,01 / β - 0,2 / μ - 100 / ϵ - 0,02 / v - 1

o primeiro ataque, a série observada manteve-se estável. Tal como no exemplo anterior, a série apresenta uma quebra quando se inicia o primeiro ataque, que é rapidamente detectado pelo teste de Page-Hinkley.

Para ilustrar um pouco mais dos resultados, a Figura 4 e a Figura 5 mostram os resultados com os protocolos UDP e ICMP.

A Figura 4 mostra o cenário para o dispositivo TP-Link Plug e o protocolo UDP, onde um ataque de negação de serviço foi iniciado na iteração 1863 e durou até a iteração 2465, sendo imediatamente sucedido por um ataque *IoT-toolkit* que durou até o cenário ser encerrado na iteração 10898. No gráfico, observamos que neste cenário, antes da introdução do primeiro ataque, os valores da série flutuaram mais do que nos outros casos, mas o teste de Page-Hinkley ainda foi capaz de produzir um alerta com um atraso de 139 iterações. Além disso, a série demonstra claramente uma mudança no seu comportamento quando o ataque começa, o que significa que a distância máxima entre os *potential core-micro-clusters* serviu como um indicador confiável da ocorrência do ataque.

A Figura 5 mostra o cenário para o dispositivo Hive e o protocolo ICMP, em que um ataque de negação de serviço foi iniciado na iteração 2363 e durou até a iteração 2559, sendo imediatamente sucedido por um ataque *man-in-the-middle* que durou até o cenário ser encerrado na iteração 2710. A série observada manteve-se estável até o início do primeiro ataque, quando ocorre um crescimento muito rápido seguido de uma nova estabilidade, que se encerra com um pico ocorrido logo após a introdução do segundo ataque. Portanto, neste cenário, a série mostra uma mudança para uma transição entre dois ataques diferentes, o que não aconteceu nos outros cenários analisados. No entanto, o teste de Page-Hinkley não foi capaz de produzir um alerta para esta alteração na série. Isto pode ser atribuído aos valores dos parâmetros que não são suficientemente sensíveis

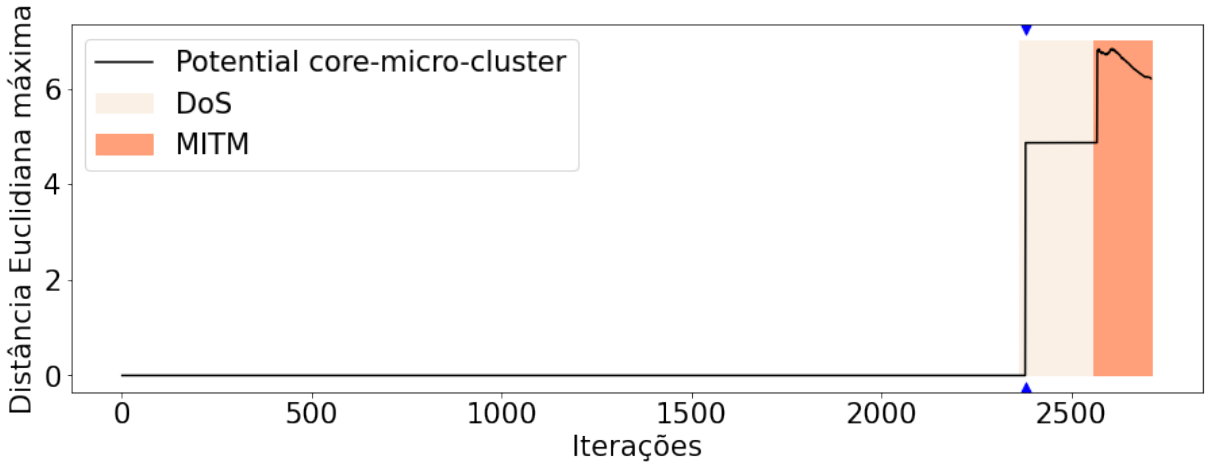

Figura 5 – Distância máxima dos *potential core-micro-clusters* para o dispositivo Hive protocolo ICMP. Parâmetros: λ - 0,01 / β - 0,01 / μ - 1000 / ϵ - 0,02 / v - 1

para detectar alterações dessa magnitude. A mudança observada na série, indicativa de uma alteração no tipo de ataque, não é tão acentuada como as tipicamente observadas durante a transição do tráfego normal para o tráfego de ataque.

4.2.2 Experimento 2

Neste experimento, o principal objetivo era avaliar como o algoritmo se comportaria em um cenário onde períodos de comportamento normal e de ataque se intercalam. No experimento 1, foi possível detectar a transição entre tráfego normal e malicioso em todos os cenários disponíveis, porém, não houve nenhum cenário onde alertas tenham sido gerados no início de ataques realizados em sequência a outros ataques. Neste segundo experimento, o teste de Page-Hinkley foi capaz de detectar quase todas as transições entre tráfego normal e malicioso, o que sugere que o aprendizado incremental do *DenStream* é adequado para este tipo de aplicação de detecção. A única exceção foi para o dispositivo Lifx TCP, onde a primeira transição foi detectada, porém, não foram encontradas combinações de valores de hiperparâmetros que gerassem uma série que sinalizasse a segunda transição entre tráfego normal e malicioso.

As Figuras 6, 7 e 8 ilustram bem o comportamento de aprendizado incremental do *DenStream*. Nos três casos é possível observar as mudanças de comportamento da série entre os períodos de tráfego normal e tráfego malicioso.

A Figura 6 mostra o cenário para o dispositivo Hive protocolo TCP, onde três ataques ocorreram: um de negação de serviço que iniciou na iteração 67203 e durou até a interação 80723, outro do tipo man-in-the-middle que iniciou na iteração 112088 e durou até a iteração 112694 e por último um ataque do tipo *Scanning* que iniciou na iteração 154275 e durou até a iteração 174924. Nos períodos de tráfego normal, podemos notar

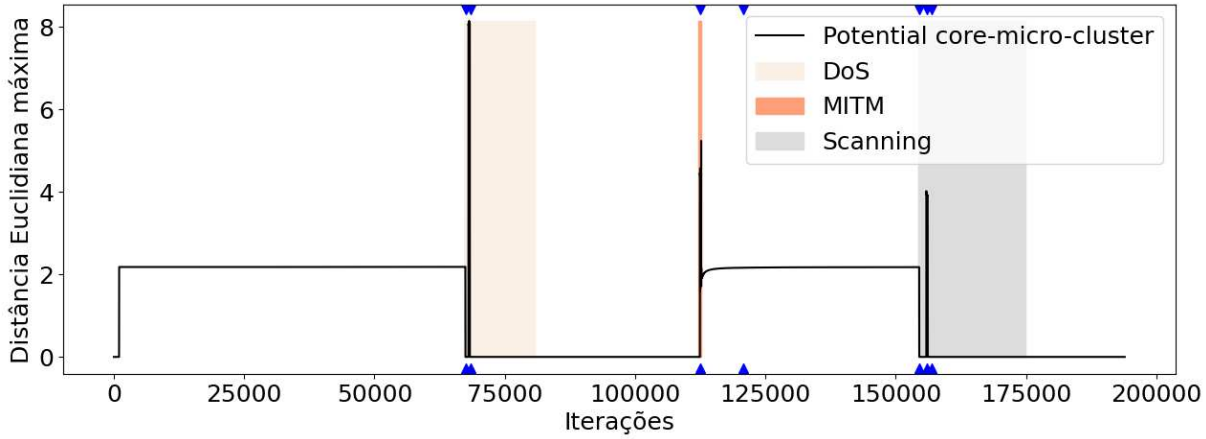

Figura 6 – Distância máxima dos *potential core-micro-clusters* para o dispositivo Hive protocolo TCP. Parâmetros: λ - 0,05 / β - 0,5 / μ - 10 / ϵ - 0,25 / v - 1

que os valores se mantém estáveis. Para os três ataques, podemos notar variações bem abruptas de valores da série no início dos ataques que foram detectados pelo teste de Page-Hinkley. O atraso médio das detecções para este cenário foi de 282 iterações.

A Figura 7 mostra o cenário para o dispositivo TP-Link Cam protocolo UDP, onde três ataques ocorreram: um de negação de serviço que iniciou na iteração 3130 e durou até a interação 4211, outro do tipo man-in-the-middle que iniciou na iteração 5232 e durou até a iteração 5450 e por último um ataque do tipo *Scanning* que iniciou na iteração 6546 e durou até a iteração 7162. Na primeira transição, podemos observar novamente a mudança abrupta assumida pelos valores da série após o início do ataque de negação de serviço. Na segunda transição, vemos uma mudança mais sutil nos valores assumidos pela série, mas há uma mudança de comportamento detectada pelo teste de Page-Hinkley. Na terceira e última transição, novamente, há a queda abrupta dos valores da série. Diferente de outros cenários apresentados anteriormente, este cenário do protocolo UDP mostra uma série que tem uma grande variação dos valores durante o período do tráfego normal. Isso sugere que o tráfego UDP tem um comportamento mais variável e que causa alteração nos clusters mesmo durante o período de tráfego normal. O atraso médio das detecções para este cenário foi de 30 iterações.

A Figura 8 mostra o cenário para o dispositivo TP-Link Plug protocolo UDP, onde dois ataques ocorreram: um de negação de serviço que inicia na iteração 1868 e dura até a iteração 3073 e outro do tipo *iot-toolkit* que inicia na iteração 3560 e dura até a iteração 11972. Nos períodos de tráfego normal, podemos observar que a série tende a assumir um valor estável e próximo de 5. Na primeira transição, podemos observar que apesar da variação de valor ser menos abrupta, o comportamento da série é bem diferente dentro do período do ataque. No segundo ataque, o comportamento caracterizado pela grande variação do valor da série é visto novamente. Neste cenário, o atraso médio das detecções

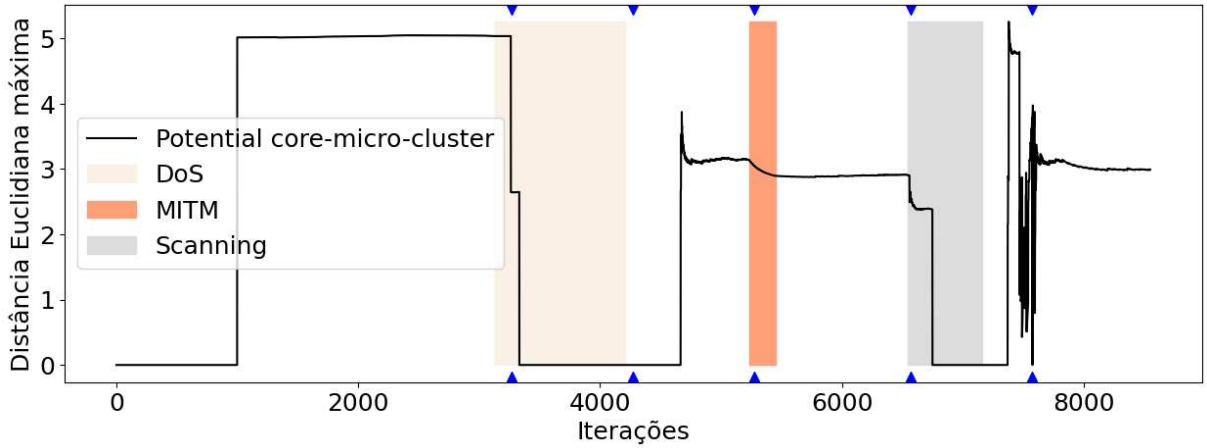

Figura 7 – Distância máxima dos *potential core-micro-clusters* para o dispositivo Tp-link Cam protocolo UDP. Parâmetros: λ - 0,05 / β - 0,2 / μ - 10 / ϵ - 1 / v - 1

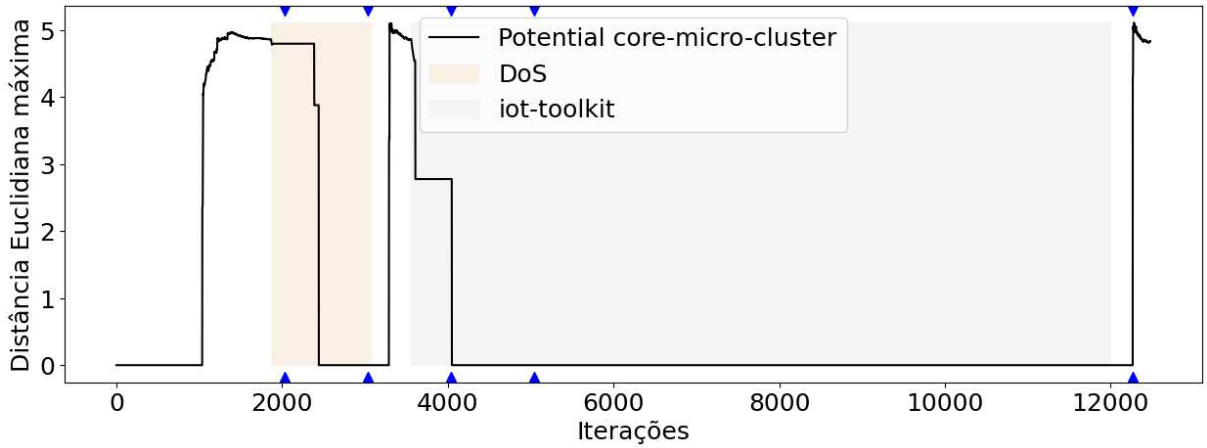

Figura 8 – Distância máxima dos *potential core-micro-clusters* para o dispositivo Tp-link plug protocolo UDP. Parâmetros: λ - 0,01 / β - 0,01 / μ - 1000 / ϵ - 0.25 / v - 1

foi de 172 iterações.

5 CONSIDERAÇÕES FINAIS

Neste trabalho, investigamos a possibilidade de utilizar o algoritmo de clusterização *DenStream* para detectar ataques ao tráfego de redes IoT. A ideia central foi analisar se métricas de monitoramento relacionadas à distância entre clusters formados pelo algoritmo poderiam oferecer indicadores da ocorrência de ataques. O processo de aplicação do *DenStream* para análise de tráfego começou com a preparação dos dados. Os pacotes coletados do tráfego de rede foram separados por dispositivo e protocolo (TCP, UDP ou ICMP). Os campos dos cabeçalhos dos protocolos de interesse (IP, TCP, UDP e ICMP) foram então selecionados para compor os pontos de dados analisados pelo algoritmo.

Durante a aplicação do *DenStream*, foram recolhidas, em cada iteração, métricas relacionadas com a distância entre *potential core-micro-clusters* e *outlier core-micro-clusters* no espaço de pesquisa. Ao aplicar o teste de Page-Hinkley para detectar automaticamente alterações nas distâncias medidas, verificamos que, para todos os dispositivos, o comportamento das distâncias máximas entre *potential core-micro-clusters* sinaliza claramente a ocorrência de ataques. Por este motivo, monitorar estas distâncias pode ser o núcleo de uma técnica de detecção de ataques nestes dispositivos.

No Experimento 1, onde o tráfego de ataque não foi separado e não intercalou com o tráfego normal, não houve cenários onde múltiplos ataques tenham sido detectados, o que contrasta com os resultados obtidos no Experimento 2, onde o método foi capaz de detectar os diferentes ataques em quase todos os cenários. Isso sugere que o *DenStream* pode ter dificuldade em apontar a ocorrência de ataques em sequência.

A seleção dos valores dos hiperparâmetros para o *DenStream* é um aspecto crítico que merece uma análise cuidadosa. Os experimentos mostraram que cada combinação de dispositivo e protocolo exigiu um conjunto diferente de valores de hiperparâmetros para produzir os melhores resultados. Isto representa um desafio significativo quando se trata de implementar este algoritmo em cenários práticos, uma vez que o ajuste dos hiperparâmetros para cada dispositivo seria uma tarefa impraticável para os usuários finais. Por conseguinte, devem ser desenvolvidas técnicas adicionais para automatizar o processo de ajuste dos hiperparâmetros.

Em trabalhos futuros, podem ser feitas algumas propostas. Em primeiro lugar, testar o desempenho desta implementação com diferentes conjuntos de dados pode ser útil para validar ainda mais o método proposto. Em segundo lugar, trabalhar com outras formas de produzir as observações que serão analisadas pelo *DenStream*. Além de realizar uma análise pacote a pacote, poderia também ser testada a opção de analisar fluxos IP. Em terceiro lugar, validar o desempenho desta implementação em cenários maiores, com

mais dispositivos.

REFERÊNCIAS

[1] 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.

[2] PISHVA, D. Internet of things: Security and privacy issues and possible solution. In: *2017 19th International Conference on Advanced Communication Technology (ICACT)*. [S.l.: s.n.], 2017. p. 797–808.

[3] YANG, K. et al. Active learning for wireless iot intrusion detection. *IEEE Wireless Communications*, v. 25, n. 6, p. 19–25, 2018.

[4] CHOW, R. The last mile for IoT privacy. *IEEE Security & Privacy*, v. 15, n. 6, p. 73–76, 2017.

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

[6] MOUSTAFA, N.; TURNBULL, B.; CHOO, K.-K. R. An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. *IEEE Internet of Things Journal*, v. 6, n. 3, p. 4815–4830, 2019.

[7] DEPREN, O. et al. An intelligent intrusion detection system (ids) for anomaly and misuse detection in computer networks. *Expert Systems with Applications*, v. 29, n. 4, p. 713 – 722, 2005. ISSN 0957-4174. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417405000989>.

[8] TOLIUPA, S.; BRAILOVSKYI, M.; PARKHOMENKO, I. Building intrusion detection systems based on the basis of methods of intellectual analysis of data. *Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska*, v. 8, n. 4, p. 28–31, Dec. 2018. Disponível em: <https://ph.pollub.pl/index.php/iapgos/article/view/795>.

[9] Salo, F. et al. Data mining techniques in intrusion detection systems: A systematic literature review. *IEEE Access*, v. 6, p. 56046–56058, 2018.

[10] SHEIKH, N. U. et al. *A Lightweight Signature-Based IDS for IoT Environment*. 2018.

[11] ZARPELãO, B. B. et al. A survey of intrusion detection in internet of things. *Journal of Network and Computer Applications*, v. 84, p. 25 – 37, 2017. ISSN 1084-8045. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1084804517300802>.

[12] ZHENG, S. et al. User perceptions of smart home IoT privacy. *Proc. ACM Hum.-Comput. Interact.*, Association for Computing Machinery, New York, NY, USA, v. 2, n. CSCW, nov. 2018.

[13] AGGARWAL, C. C. et al. A framework for clustering evolving data streams. In: ELSEVIER. *Proceedings 2003 VLDB conference*. [S.l.], 2003. p. 81–92.

[14] GAMA, J. *Knowledge discovery from data streams*. [S.l.]: CRC Press, 2010.

[15] NAKAGAWA, F. H. Y.; JUNIOR, S. B.; ZARPELãO, B. B. Attack detection in smart home iot networks using clustream and page-hinkley test. In: *2021 IEEE Latin-American Conference on Communications (LATINCOM)*. [S.l.: s.n.], 2021. p. 1–6.

[16] LOPES, S. F. F. The importance of the itil framework in managing information and communication technology services. v. 8, p. 292–296, 05 2021.

[17] TUTTLE, B.; VANDERVELDE, S. D. An empirical examination of cobit as an internal control framework for information technology. *International Journal of Accounting Information Systems*, v. 8, n. 4, p. 240–263, 2007. ISSN 1467-0895. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1467089507000425>.

[18] HOANG, L.; NKEMBI, A.; PHAM, P. L. Real-time risk assessment detection for weak people by parallel training logical execution of a supervised learning system based on an iot wearable mems accelerometer. *Sensors*, v. 23, p. 1516, 01 2023.

[19] INTERNET of Things Applications Part 2: The Mining Industry. <https://centricdigital.com/blog/digital-trends/internet-of-thingsapplications-pt2-the-mining-industry/>. Accessado: 2023-03-20.

[20] SMART Cities—International Case Studies, Korea Res. Inst. Human Settlements, Inter Amer. Develop. Bank, Washington, DC, USA, 2016. <http://www.iadb.org/en/topics/emerging-andsustainable-cities/international-case-studies-of-smart-cities,20271.html>. Accessado: 2023-03-20.

[21] L. Franceschi-Bicchierai. Internet of Things Teddy Bear Leaked 2 Million Parent and Kids Message Recordings. Mar. 5, 2018. <https://motherboard.vice.com/en_us/article/pgwean/internet-of-things-teddy-bear-leaked-2-millionparent-and-kids-message-recordings>. Accessado: 2023-03-20.

[22] ZUBIAGA, A.; PROCTER, R.; MAPLE, C. *A Longitudinal Analysis of the Public Perception of the Opportunities and Challenges of the Internet of Things*. 2018.

[23] EXECUTIVE ORDER 14028, IMPROVING THE NATION'S CYBERSECURITY. 2021. <https://www.nist.gov/itl/executive-order-14028-improving-nations-cybersecurity>. Accessado: 2023-04-24.

[24] AL-FUQAHA, A. et al. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys Tutorials*, v. 17, n. 4, p. 2347–2376, 2015.

[25] NESHENKO, N. et al. Demystifying iot security: An exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations. *IEEE Communications Surveys Tutorials*, v. 21, 04 2019.

[26] TRAPPE, W.; HOWARD, R.; MOORE, R. Low-energy security: Limits and opportunities in the internet of things. *IEEE Security Privacy*, v. 13, p. 14–21, 01 2015.

[27] OLAWUMI, O. et al. Three practical attacks against zigbee security: Attack scenario definitions, practical experiments, countermeasures, and lessons learned. In: . [S.l.: s.n.], 2014.

[28] ANGRISHI, K. Turning internet of things(iot) into internet of vulnerabilities (iov) : Iot botnets. 02 2017.

[29] MARKOWSKY, L.; MARKOWSKY, G. Scanning for vulnerable devices in the internet of things. In: *2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*. IEEE Press, 2015. p. 463–467. ISBN 978-1-4673-8359-2. Disponível em: <https://doi.org/10.1109/IDAACS.2015.7340779>.

[30] ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. *Computer Networks*, p. 2787–2805, 10 2010.

[31] GAMA, J.; RODRIGUES, P. P. Data stream processing. In: _____. *Learning from Data Streams: Processing Techniques in Sensor Networks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 25–39. ISBN 978-3-540-73679-0.

[32] MUTHUKRISHNAN, S. *Data streams: Algorithms and applications*. [S.l.]: Now Publishers Inc, 2005.

[33] CAO, F. et al. Density-based clustering over an evolving data stream with noise. In: *SDM*. [S.l.: s.n.], 2006.

[34] STAVROULAKIS, P. P.; STAMP, M. Handbook of information and communication security. In: *Handbook of Information and Communication Security*. [S.l.: s.n.], 2010.

[35] XENAKIS, C.; PANOS, C.; STAVRAKAKIS, I. A comparative evaluation of intrusion detection architectures for mobile ad hoc. *Computers Security*, v. 30, p. 63–80, 04 2011.

[36] AXELSSON, S. Intrusion detection systems: A survey and taxonomy. Citeseer, 2000.

[37] LIAO, H.-J. et al. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, v. 36, n. 1, p. 16–24, 2013. ISSN 1084-8045. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1084804512001944>.

[38] LOHIYA, R.; THAKKAR, A. A review on machine learning and deep learning perspectives of ids for iot: Recent updates, security issues, and challenges. *Archives of Computational Methods in Engineering*, v. 28, 10 2020.

[39] SILVA, J. et al. Data stream clustering: A survey. *ACM Computing Surveys*, v. 46, 03 2014.

[40] OTOUM, Y.; NAYAK, A. As-ids: Anomaly and signature based ids for the internet of things. *Journal of Network and Systems Management*, v. 29, 07 2021.

[41] OTOUM, Y.; LIU, D.; NAYAK, A. Dl-ids: a deep learning–based intrusion detection framework for securing iot. *Transactions on Emerging Telecommunications Technologies*, v. 33, 03 2022.

[42] SCARANTI, G. F. et al. Unsupervised online anomaly detection in software defined network environments. *Expert Systems with Applications*, v. 191, p. 116225, 2022. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417421015384>.

[43] YIN, C. et al. Improved clustering algorithm based on high-speed network data stream. *Soft computing (Berlin, Germany)*, Springer Berlin Heidelberg, Berlin/Heidelberg, v. 22, n. 13, p. 4185–4195, 2018. ISSN 1432-7643.

[44] RESENDE, P. A. A.; DRUMMOND, A. C. A survey of random forest based methods for intrusion detection systems. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 51, n. 3, may 2018. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3178582>.

[45] MOHAMMADI, M. et al. A comprehensive survey and taxonomy of the svm-based intrusion detection systems. *Journal of Network and Computer Applications*, v. 178, p. 102983, 2021. ISSN 1084-8045. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1084804521000102>.

[46] RAMíREZ-GALLEGO, S. et al. A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing*, v. 239, p. 39–57, 2017. ISSN 0925-2312. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0925231217302631>.

[47] PAGE, E. S. CONTINUOUS INSPECTION SCHEMES. *Biometrika*, v. 41, n. 1-2, p. 100–115, 06 1954. ISSN 0006-3444. Disponível em: <https://doi.org/10.1093/biomet/41.1-2.100>.

TRABALHOS PUBLICADOS PELO AUTOR

1. Gabriel Keith Tazima, Bruno Bogaz Zarpelão **Behavior of the DenStream Clustering Algorithm for Attack Detection in the Internet of Things**, Semina: Ciências Exatas E Tecnológicas, 44, 2023, e48956. (Qualis 2017 - 2020: B4) https://doi.org/10.5433/1679-0375.2023.v44.48956