UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ (UTFPR)

FÁBIO CÉSAR SCHUARTZ

UMA PROPOSTA PARA DETECÇÃO DISTRIBUÍDA DE INTRUSÕES UTILIZANDO MINERAÇÃO EM FLUXOS DE AMBIENTES BIG DATA

CURITIBA

2023

FÁBIO CÉSAR SCHUARTZ

UMA PROPOSTA PARA DETECÇÃO DISTRIBUÍDA DE INTRUSÕES UTILIZANDO MINERAÇÃO EM FLUXOS DE AMBIENTES BIG DATA

A Proposal for Distributed Intrusion Detection Using Stream Mining in Big Data Environments

Tese apresentada como requisito para obtenção do título de Doutor em Engenharia Elétrica e Informática Industrial, do Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial, da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientador: Prof. Dr. Mauro Sergio Pereira Fonseca

Coorientador: Profª. Drª. Anelise Munaretto Fonseca

CURITIBA

2023

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Curitiba

FABIO CESAR SCHUARTZ

UMA PROPOSTA PARA DETECÇÃO DISTRIBUÍDA DE INTRUSÕES UTILIZANDO MINERAÇÃO EM FLUXOS DE AMBIENTES BIG DATA

Trabalho de pesquisa de doutorado apresentado como requisito para obtenção do título de Doutor Em Ciências da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Telecomunicações E Redes.

Data de aprovação: 26 de Maio de 2023

Dr. Mauro Sergio Pereira Fonseca, Doutorado - Universidade Tecnológica Federal do Paraná

Dra. Anelise Munaretto Fonseca, Doutorado - Universidade Tecnológica Federal do Paraná

Dr. Carlos Alberto Maziero, Doutorado - Universidade Federal do Paraná (Ufpr)

Dr. Daniel Fernando Pigatto, Doutorado - Universidade Tecnológica Federal do Paraná

Dr. Luiz Nacamura Junior, Doutorado - Universidade Tecnológica Federal do Paraná

Dra. Michele Nogueira Lima, Doutorado - Universidade Federal de Minas Gerais (Ufmg)

Dedico este trabalho a minha família, aos meus amigos e professores da universidade, pelos momentos de conforto e suporte durante este longo trajeto.

AGRADECIMENTOS

Este trabalho não poderia ser terminado sem a ajuda de diversas pessoas e instituições às quais presto minha homenagem.

Aos meus orientadores, que me guiaram durante esta longa viagem de aprendizagem, mostrando extrema paciência e profunda habilidade de ensino.

A todos os professores do departamento, que ajudaram de forma direta e indireta na conclusão deste trabalho.

A minha família e amigos, pelo carinho, incentivo e total apoio durantes momentos de dificuldade e reflexão, especialmente durante a pandemia.

Enfim, a todos os que de alguma forma contribuíram para a realização deste trabalho.

If security were all that mattered, computers would never be turned on, let alone hooked into a network with literally millions of potential intruders. (Farmer, Dan, 1997).

RESUMO

Com a enorme expansão das redes de computadores, aumentou a necessidade de proteger os dados e informações que trafegam pela rede. O crescimento em volume, velocidade e variedade dos dados exige um sistema de detecção de intrusão mais robusto, preciso e capaz de analisar uma quantidade gigantesca de dados.

Este trabalho propõe a criação de um sistema de detecção de intrusão utilizando uma arquitetura distribuída. O sistema utiliza três camadas de classificação de fluxo de amostras, coletadas a partir da rede, onde cada camada de classificação pode ser composta de um ou mais classificadores, trabalhando em paralelo. Os classificadores são treinados com diferentes métodos de aprendizagem de máquina.

Cada camada pode selecionar as principais características a serem utilizadas para classificar as amostras, e uma unidade classificadora final, o Decisor, utiliza o método de votação para determinar o resultado final da classificação. A arquitetura utiliza classificadores em fluxo, porém é possível utilizar classificadores tradicionais.

Os resultados obtidos pela arquitetura apresentada, utilizando duas bases de dados para validação do sistema proposto - NSL-KDD e CICIDS2017, mostram ganhos na acurácia de até 18,52% e 3,55%, utilizando as bases NSL-KDD e CICIDS2017, respectivamente. Obteve-se reduções no tempo de classificação de até 35,51% e 94,90%, respectivamente. Um fator negativo gerado pelo trabalho proposto é o aumento no requisito de memória, devido ao uso de diversos classificadores simultaneamente, ao invés de apenas um classificador, utilizado normalmente. Esse aumento na RAM/Hora foi de até 351,42% e 1.016,52% para as bases NSL-KDD e CICIDS2017, respectivamente.

Palavras-chave: detecção de intrusões; aprendizagem de máquina; big data; segurança de redes; mineração de dados.

ABSTRACT

With the enormous expansion of computer networks, the need to protect data and information that travels over the web has increased. The growth in volume, speed, and variety of data requires an intrusion detection system that is more robust, accurate, and capable of analyzing a huge amount of data.

This work proposes the creation of an intrusion detection system using a distributed architecture. The system uses three layers of classification of sample streams, collected from the network, where each classification layer can be composed of one or more classifiers, working in parallel. Classifiers are trained with different machine-learning methods.

Each layer can select the main characteristics to be used to classify the samples, and a final classification unit, the Decider, uses the voting method to determine the final classification result. The architecture uses stream classifiers, but it is possible to use traditional classifiers.

The results obtained by the proposed architecture, using two databases for validation of the proposed system - NSL-KDD and CICIDS2017, show gains in accuracy of up to 18.52% and 3.55%, using the NSL-KDD and CICIDS2017 databases, respectively. Reductions in classification time of up to 35.51% and 94.90%, respectively, were obtained. A negative factor generated by the proposed work is the increase in the memory requirement, due to the use of several classifiers simultaneously, instead of just one classifier, normally used. This increase in RAM/Hour was up to 351.42% and 1016.52% for the NSL-KDD and CICIDS2017 bases, respectively.

Keywords: detection intrusion; machine learning; big data; network security; data mining.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Siglas

AE	Auto-codificador, do inglês *Autoencoder*
AFD	Análise Final dos Dados
AID	Análise Inicial dos Dados
ANN	Rede Neural Artificial, do inglês *Artificial Neural Network*
APD	Análise Profunda dos Dados
AR	Razão de Atributo, do inglês *Attribute Ratio*
ARFF	Formato de arquivo de relação de atributo, do inglês *Attribute-Relation File Format*
ARP	Protocolo de Resolução de Endereços, do inglês *Address Resolution Protocol*
AS	Seleção de Atributos, do inglês *Attribute Selection*
BN	Redes Bayesianas, do inglês *Bayesian Network*
CD	Desvio de Conceito, do inglês *Concept Drift*
DoS	Negação de Serviço, do inglês *Denial of Service*
DSP	Processador de Fluxo de Dados, do inglês *Data Streaming Processor*
DT	Árvores de Decisão, do inglês *Decision Trees*
EPC	Captura de Pacote Embarcado, do inglês *Embedded Packet Capture*
FS	Seleção de Características, do inglês *Feature Selection*
GAD	Grafos Acíclicos Dirigidos
GPU	Unidade de Processamento Gráfica, do inglês *Graphics Processing Unit*
HAT	Árvore Adaptativa de Hoeffding, do inglês *Hoeffding Adaptive Tree*
IA	Endereço de Internet, do inglês *Internet Address*
ICMP	Protocolo de Mensagens de Controle da Internet, do inglês *Internet Control Message Protocol*

IDS	Sistema de Detecção de Intrusão, do inglês *Intrusion Detection System*
IG	Ganho de Informação, do inglês *Information Gain*
IoT	Internet das Coisas, do inglês *Internet of Things*
IP	Protocolo de Internet, do inglês *Internet Protocol*
KNN	K-Vizinho Mais Próximo, do inglês *K-Nearest Neighbor*
LR	Regressão Logística, do inglês *Logistic Regression*
ML	Aprendizado de máquina, do inglês *Machine Learning*
MOA	Análise Massiva Online, do inglês *Massive Online Analysis*
NB	Naive Bayes, do inglês *Naive Bayes*
NDAE	Auto-codificador Não-simétrico de Profundidade, do inglês *Nonsymmetric Deep Autoencoder*
NIDS	Sistema de Detecção de Intrusão Baseado em Rede, do inglês *Network Intrusion Detection System*
NVF	Função Virtualizada de Rede, do inglês *Network Virtualized Function*
OPNVF	Plataforma Aberta para Função Virtualizada de Rede, do inglês *Open Platform for Network Functions Virtualization*
PTT	Pontos de Troca de Tráfego
RBM	Máquina de Boltzmann Restrita, do inglês *Restricted Boltzmann Machine*
RF	Florestas Aleatórias, do inglês *Random Forests*
SDN	Rede Definida por Software, do inglês *Software-Defined Networking*
SMO	Otimização Mínima Sequencial, do inglês *Sequential Minimal Optimization*
SVM	Máquina de Vetores de Suporte, do inglês *Support Vector Machine*
TCP	Protocolo de Controle de Transmissão, do inglês *Transmission Control Protocol*
TCP/IP	Protocolo de Controle de Transmissão/Protocolo de Internet, do inglês *Transmission Control Protocol/Internet Protocol*
UDP	Protocolo de Datagrama do Usuário, do inglês *User Datagram Protocol*
UPCD	Unidade de Processamento e Classificação de Dados

SUMÁRIO

1 INTRODUÇÃO

Com a vasta expansão das redes de computadores, as pessoas estão cada vez mais conectadas, utilizando dispositivos que compartilham informações através da Internet. Este crescente aumento de dispositivos em funcionamento acarreta em uma gama de vulnerabilidades que podem ser exploradas por um agente malicioso (LEU *et al.*, 2015). As pessoas ganham benefícios e as empresas lucram gerenciando seus recursos e transações na rede, criando maiores oportunidades para que usuários mal-intencionados roubem informações pessoais e secretas. O uso crescente da Internet criou uma necessidade maior de proteger os dados e informações armazenados em servidores centralizados, especialmente em sistemas acessados por meio de uma rede pública. Nos últimos anos, várias estatísticas publicadas mostram um número crescente de invasões relatadas (SYMANTEC, 2019; CROWDSTRIKE, 2021).

O aumento em volume, velocidade e variedade de dados nas redes atuais exige uma infraestrutura de segurança robusta. Monitorar e processar dados em altas taxas sem desperdiçar recursos é um grande desafio hoje (LOPEZ *et al.*, 2018). O número gigantesco de dispositivos geram grandes massas de dados, conhecidos como *Big Data*. Esses dados são gerados em forma de fluxos e necessitam ser processados, transferidos e armazenados para gerenciamento em tempo real e de forma segura. Entretanto, devido às tecnologias atuais não terem sido projetadas para esta enorme demanda, este cenário traz novos desafios que necessitam ser resolvidos. Ainda, com a virtualização de recursos e ferramentas, utilizadas em sistemas de computação em nuvem, novas ameaças são introduzidas no problema. Assim, novas vulnerabilidades são geradas devido ao volume, à velocidade e à variedade das grandes massas de dados, além do uso da tecnologia de virtualização (WANG; WANG; XUE, 2021).

A segurança em redes é um grande desafio que tende a tornar-se ainda mais complexo no futuro, devido em grande parte pelo crescimento no número de ameças geradas pela internet das coisas (Internet das Coisas, do inglês *Internet of Things* (IoT)) (GLUHAK *et al.*, 2011). A heterogeneidade dos milhões de dispositivos conectados e o crescimento da velocidade de rede dificultam o gerenciamento e a proteção das redes de comunicação. Estima-se que até 2025, mais de 80 bilhões de dispositivos estarão interconectados, gerando desafios na segurança e na privacidade dos dados, dentro de um cenário onde o gerenciamento e a proteção das redes de comunicação são elementos de alta complexidade (CLAY, 2015).

A internet das coisas proporciona a conectividade entre milhões de dispositivos globalmente e isso permite que uma vulnerabilidade possa afetar esses aparelhos simultaneamente. Um ataque de varredura de portas pode descobrir uma vulnerabilidade que, futuramente, permite um ataque de negação de serviço (Negação de Serviço, do inglês *Denial of Service* (DoS)) em larga escala. De acordo com os relatórios de ameaças divulgados por empresa de segurança (SYMANTEC, 2019; CROWDSTRIKE, 2021), a Internet das Coisas continua a crescer como alvo primário para os cibercriminosos explorarem. O número de ataques à IoT cresceu de aproximadamente 6.000 em 2016 para 50.000 em 2017, um aumento de 600% em apenas

um ano, segundo o relatório. Atualmente, bilhões de dispositivos que utilizam a IoT estão em uso, e a companhia de pesquisa e análises Gartner (GARTNER, 2016) estima que mais de 50% dos novos processos e sistemas de negócios incluem um componente ligado à IoT. Ainda, a empresa F5 Labs (LABS, 2017) divulgou seu relatório global sobre ameaças à IoT, o qual examinou como atacantes desenvolveram *botnets* para visar dispositivos conectados especificamente à IoT, os quais cresceram 280% em relação ao relatório semestral anterior.

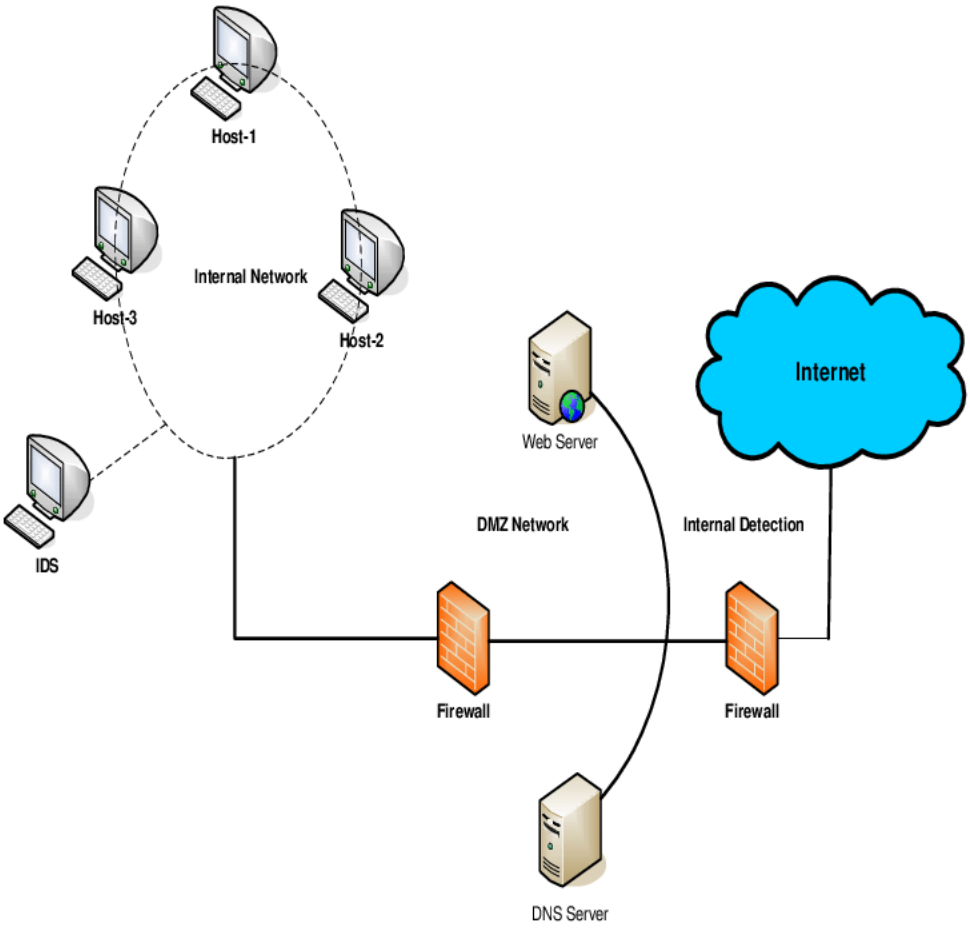
Outro fator muito importante para manter a segurança nos sistemas de comunicação é o tempo de detecção das ameaças (WU *et al.*, 2014). Para caracterizar o tráfego e extrair informações sobre o mesmo, é necessário monitorar, processar e gerenciar grandes quantidades de dados, permitindo assim detectar ameaças de forma eficiente. Atualmente, os sistemas são projetados para coletar dados de diversas fontes, porém gerenciando as informações de segurança em um ponto único. Esses sistemas não possuem bons resultados, pois cerca de 85% das intrusões na rede são detectadas somente depois de semanas de sua ocorrência (em média 206 dias para detectar um vazamento de informações), enquanto o tempo médio de reação às ameaças é de 123 horas depois de ocorridas (CLAY, 2015). Assim, o tempo prolongado para detecção de uma ameaça impossibilita uma defesa eficaz por parte do sistema de prevenção de ameaças atuais. Outro problema encontrado nos sistemas atuais é a enorme quantidade de alarmes gerados, muitos deles sendo falsos positivos e que são, na maior parte, ignorados pelos analistas de segurança. Por não conseguirem lidar com a quantidade de dados que chegam, alertas reais, que representam graves ameaças, muitas vezes passam desapercebidos.

Detectar ameaças em fluxos com grandes quantidades de dados requer o desenvolvimento de técnicas modernas de análise que permitam o processamento de fluxo em tempo real. Isso permite processar e analisar os diferentes tipos de dados, resultando na melhor detecção de ameaças. Entretanto, as abordagens atuais de monitoramento em tempo real não possuem escalabilidade e acurácia para detectar comportamentos anômalos. Assim, são necessários novos mecanismos e técnicas para a caracterização de tráfego, que permitem classificar ameaças e detectar anomalias em tempo real sobre grandes volumes de dados. Com isso em mente, é importante termos mecanismos de segurança que possam detectar essas formas de ataque de uma maneira rápida e com acurácia para termos uma Internet das Coisas segura. Entretanto, as técnicas comuns de detecção para os ataques de varredura e negação de serviço não são eficientes e rápidas o suficiente para trabalhar com um grande volume de dados de tráfego. Assim, tem se tornado mais comum o uso de algoritmos de aprendizagem de máquina para detectar tais intrusões, pois a alta capacidade de processamento distribuído por máquinas utilizando o processamento de fluxo permite construir algoritmos complexos e eficientes que podem tratar os dados de maneira *online*.

Para combater tais ataques, diversos pesquisadores e instituições começaram a desenvolver tecnologias de detecção de invasão. Um sistema de detecção de intrusão (Sistema de Detecção de Intrusão, do inglês *Intrusion Detection System* (IDS)) é um sistema utilizado para monitorar as atividades de outro sistema ou de uma rede, procurando por atividades maliciosas

e produzindo mensagens de alerta para a estação de controle (TAN *et al.*, 2014). Os IDS são classificados em dois grupos principais, de acordo com os dados a serem analisados: baseado em hospedeiro ou baseado em rede. O IDS baseado em hospedeiro usa *logs* do sistema operacional na análise, enquanto o IDS baseado em rede opera capturando e avaliando os pacotes de tráfego recebidos pela rede (ALSAFI, 2013). Um IDS pode operar segundo duas abordagens: detecção de assinatura e detecção de anomalia estatística. A detecção de assinatura é usada para pesquisar ataques baseados em padrões extraídos de intrusões conhecidas, enquanto a detecção de anomalias tenta detectar ataques com base no comportamento do tráfego (não para IDS baseado em hospedeiro) que difere do padrão de comportamento normal usando métodos estatísticos, baseados em distância, sistemas baseados em regras, perfis e baseado em modelo (LAZAREVIC; KUMAR; SRIVASTAVA, 2005). A Figura 1 apresenta uma rede composta por três hospedeiros e um IDS monitorando o tráfego de rede.

Figura 1 – Exemplo de um sistema IDS monitorando o tráfego de rede.



Fonte: Alsafi (2013)..

Uma maneira de estudar os ataques ocorridos na rede é utilizar técnicas de aprendizagem de máquina, tal como classificação e regressão, para procurar por padrões no tráfego da rede. A literatura apresenta diversas técnicas para detecção de ameaças utilizando aprendizado de máquina. Essas abordagens de aprendizado de máquina podem ser classificadas

em quatro tipos de aprendizados: supervisionado, não-supervisionado, semi-supervisionado e por reforço. O aprendizado supervisionado ocorre quando o conjunto de dados estiver rotulado, ou seja, cada amostra contém a saída (classificação) correta que ela deve gerar, enquanto no aprendizado não-supervisionado as amostras não possuem esse rótulo. O semi-supervisionado contém amostras com e sem rotulação, e no aprendizado por reforço os agentes tentam encontrar a melhor maneira de atingir uma meta, recebendo uma recompensa quando realizam uma ação que os deixem mais perto do objetivo (PECHT; KANG, 2019).

Um classificador pode estudar diversos exemplos de entradas que produzem resultados conhecidos, através de um aprendizado supervisionado, conforme Harbi (HARBI; BAHRI, 2013). Porém, os ataques evoluem e podem não seguir os padrões de ataques anteriores. Assim, as técnicas de aprendizado não supervisionado podem apresentar melhores resultados, pois procuram por variações em padrões conhecidos, ao invés de classificar o tráfego em apenas uma categoria. Uma área de pesquisa atual recebendo grande atenção é o de *deep learning* (Vinayakumar *et al.*, 2019). Esta é uma sub-área avançada da aprendizagem de máquina, que permite sobrepujar as limitações do aprendizado superficial. Sua característica superior de camadas de aprendizagem pode resultar em desempenho superior ou equivalente, comparado às técnicas de aprendizado superficiais.

Na área de redes, utiliza-se o aprendizado supervisionado para realizar a classificação de ataques, enquanto o não-supervisionado é utilizado para detectar padrões e anomalias. Na classificação de ataques, os métodos mais utilizados na literatura são as Árvores de Decisão, do inglês *Decision Trees* (DT), o K-Vizinho Mais Próximo, do inglês *K-Nearest Neighbor* (KNN), as Redes Bayesianas, do inglês *Bayesian Network* (BN) e a Máquina de Vetores de Suporte, do inglês *Support Vector Machine* (SVM) (CHEN *et al.*, 2015).

1.1 Motivação

Até o presente, nenhuma combinação de tecnologias pode proteger completamente um sistema. Assim, pesquisadores estão procurando técnicas melhores para minimizar e sobrepujar algumas limitações das técnicas de prevenção existentes. Embora existam diversas técnicas e propostas na literatura para a detecção de intrusão, a maioria delas ainda não são eficientes o suficiente para trabalhar com um grande fluxo de dados (*Big Data*) em tempo real, enquanto outras propostas não apresentam acurácia suficiente. Este trabalho propõe um sistema para detecção de ameaças em tempo real através da análise de grandes volumes de dados pelo processamento de fluxos, utilizando uma arquitetura distribuída. A arquitetura utiliza ferramentas de código aberto dentro de uma topologia onde vários classificadores analisarão os dados coletados de diversas fontes, procurando por possíveis ataques à rede. Estes classificadores realizarão operações de classificação em um conjunto de dados que pode ter seu número de características reduzido, conforme exista a necessidade, e os resultados obtidos por cada classificador serão enviados a um decisor final, o qual irá realizar a classificação final de um determi-

nado pacote de dados, com o objetivo de obter um aumento do grau de acerto na classificação de ataques e redução de falsos-positivos e falsos-negativos.

1.2 Objetivos

1.2.1 Objetivo Geral

A tese propõe uma arquitetura para detecção de intrusão sobre uma plataforma distribuída capaz de fornecer um serviço integrado de coleta, transporte, monitoramento e caracterização de tráfego em tempo real, capaz de detectar ameaças e reagir prontamente às interrupções de serviços e às ameaças de segurança.

1.2.2 Objetivos Específicos

- Desenvolver uma plataforma distribuída que implementa a arquitetura proposta para análise em tempo real do tráfego de rede, utilizando equipamentos de baixo custo e ferramentas de código aberto e gratuitas para verificar a tese proposta;
- Coletar, analisar e caracterizar um grande volume de dados, incluindo dados de tráfego em tempo real;
- Obter um conjunto de dados, a partir da extração de parâmetros, que correspondem ao uso real de uma rede;
- Selecionar as características relevantes para detecção de ameaças, através da redução da dimensionalidade dos dados;
- Usar em paralelo diferentes tipos de algoritmos para classificação dos dados, pelo processo supervisionado;
- Utilizar um número diferente de características em camadas diferentes do processo de classificação visando reduzir o tempo de classificação do dado sem comprometer a acurácia;
- Utilizar um sistema de votação com peso para classificação final do dado;
- Demonstrar a maior acurácia e menor tempo de resposta na detecção e classificação de ameaças à segurança, comparando os resultados obtidos com outros esquemas propostos na literatura;
- Verificar a influência no desempenho da arquitetura proposta através da automatização de ferramentas, técnicas e procedimentos de forma integrada e inteligente contra as ameaças de segurança;

1.2.3 Resultados Obtidos

Um protótipo da arquitetura foi desenvolvido e avaliado, e comparando os resultados obtidos pela arquitetura proposta e os demais métodos de classificação não de fluxo obtivemos respectivamente ganhos na precisão e no *recall* de até 15,84% e 29,70% no melhor caso, e 8,28% e 9,45% no pior caso. Para os classificadores de fluxo de dados, obtivemos melhorias no Kappa em até 97,39% no melhor caso, e de até 1,26% no pior caso. Para avaliação do sistema, utilizaram-se dois conjuntos de dados com as classes marcadas, contendo dados normais e ataques, proveniente das bases de dados KDD'99 e CICIDS2017. Através do uso de técnicas de mineração de dados e mineração de fluxo de dados, a arquitetura apresentou ganhos na acurácia e na redução no tempo de detecção de ameaças comparado com outras propostas descritas na literatura. Ainda, pode-se demonstrar vantagens em se utilizar classificadores de fluxo de dados ao invés de classificadores convencionais (mineração de dados), como, por exemplo, aumento na acurácia e redução no tempo de detecção de ameaças. A arquitetura demonstrou ser escalável através da utilização de recursos de múltiplos equipamentos de forma incremental.

Os artigos abaixo foram elaborados com os resultados obtidos deste doutorado e apresentado em diversos simpósios brasileiros, publicado como capítulo de livro e como artigo em periódico internacional.

SCHUARTZ, F. C. ; FONSECA, M. S. P. ; MUNARETTO, A. Sistema Distribuído para Detecção de Ameaças em Tempo Real Utilizando Big Data. In: **XXXV Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT'2017)**, 2017, São Pedro, SP.

SCHUARTZ, F. C. ; FONSECA, M. S. P. ; MUNARETTO, A. Sistema Distribuído para Detecção de Ameaças em Redes Utilizando *Deep Learning*. In: **XXXVI Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT'2018)**, 2018, Campina Grande, PE.

SCHUARTZ, F. C. ; FONSECA, M. S. P. ; MUNARETTO, A. Distributed System for Threat Detection in Networks Using Machine Learning. In: **1st Blockchain, Robotics and AI for Networking Security Conference**, 2019, Rio de Janeiro. RJ.

SCHUARTZ, F. C. ; FONSECA, M. S. P. ; MUNARETTO, A. Uma Comparação entre os Sistemas de Detecção de Ameaças Distribuídas de Rede Baseado no Processamento de Dados em Fluxo e em Lotes. In: **XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)**, 2019, Gramado. RS.

SCHUARTZ, F. C. ; FONSECA, M. S. P. ; MUNARETTO, A. Sistema Distribuído para Detecção de Ameaças em Redes Utilizando *Deep Learning*. In: **Henrique Ajuz Holzmann. (Org.). Técnicas de Processamento de Sinais e Telecomunicações**. 1ed.Ponta Grossa: Atena Editora, 2019, v. 1, p. 101-112.

SCHUARTZ, F. C. ; FONSECA, M. S. P. ; MUNARETTO, A. Improving threat detection in networks using deep learning. **ANNALS OF TELECOMMUNICATIONS**, v. 75, p. 133-142, 2020. https://doi.org/10.1007/s12243-019-00743-5.

SCHUARTZ, F. C. ; FONSECA, M. S. P. ; MUNARETTO, A. A Distributed Platform for Intrusion Detection System Using Data Stream Mining in a Big Data Environment. In: **6th Cyber Security in Networking Conference**, 2022, Rio de Janeiro. RJ.

SCHUARTZ, F. C. ; FONSECA, M. S. P. ; MUNARETTO, A. A Distributed Platform for Intrusion Detection System Using Data Stream Mining in a Big Data Environment. **ANNALS OF TELECOMMUNICATIONS**, em processo de publicação.

O restante do trabalho é dividido em quatro partes. O capítulo dois apresenta a revisão de literatura, apresentada em ordem cronológica. O terceiro capítulo contém a descrição de uma prova de conceito da proposta, metodologia para a obtenção de resultados e amostra de resultados obtidos. No capítulo quatro, apresentamos os dados obtidos no trabalho e comparamos com os descritos na literatura por outros autores. No capítulo cinco, é apresentada a conclusão e perspectivas do trabalho, assim como assuntos que foram identificados como importantes para serem explorados em novas pesquisas.

2 REVISÃO DA LITERATURA

Neste capítulo serão apresentados diversos trabalhos publicados pela comunidade, obtidos através da pesquisa sistemática, que trazem informações sobre o estado da arte nas diversas áreas de conhecimento empregadas neste trabalho.

Em 2000, Domingos e Hulten (2000) propõem uma variação na árvore de decisão de Hoeffding. Ao invés de reutilizar amostras, aguardam-se novas amostras chegarem, tornando o trabalho com fluxo de dados mais rápido. A classificação através de fluxo de dados funciona criando um modelo de aprendizado de máquina que pode ser atualizado continuamente com novos dados. Isso permite que o modelo aprenda e se adeque às mudanças nos dados, o que é importante em aplicações de big data, onde os dados estão sempre mudando. Entretanto, não podemos armazenar todos os dados recebidos. O principal problema da construção de uma árvore de decisão é a necessidade de reutilizar os exemplos para calcular os melhores atributos de divisão. A característica mais interessante da árvore de Hoeffding proposta é que ela constrói uma árvore idêntica à tradicional, com alta probabilidade se o número de instâncias for grande o suficiente.

Em 2003, um sistema de detecção de anomalias na rede baseado em tempo real foi proposto por Balupari *et al.* (2003). O sistema analisa o fluxo de dados gerado pelo *TcpTrace* em tempo real, que reporta periodicamente estatísticas de todas as conexões Protocolo de Controle de Transmissão/Protocolo de Internet, do inglês *Transmission Control Protocol/Internet Protocol* (TCP/IP) abertas na rede. Com isso, são construídos perfis estatísticos que mostram o comportamento normal dos serviços na rede, permitindo qualquer comportamento anormal ser caracterizado como uma invasão. Esta abordagem é capaz de monitorar quaisquer serviços na rede sem o conhecimento prévio do seu comportamento.

Em 2011, Holtz, David e Júnior (2011) propõem uma arquitetura para IDS em uma rede distribuída, onde a análise dos dados é realizada em um ambiente de computação em nuvem. Utilizando equipamentos de rede, como servidores e estações de trabalho, foram coletados dados do tráfego na rede, arquivos *logs* do sistema operacional e dados de aplicações gerais em diversos pontos da rede, sendo estes agregados, processados e comparados através da estrutura *Map-Reduce*. A detecção de intrusões e atividades maliciosas podem ser identificadas analisando as correlações dos eventos. O sistema proposto pode manusear grandes fluxos de dados, utilizando a ferramenta de processamento em lotes *Apache Hadoop* (Apache Software Foundation, 2006).

Em 2013, Devikrishna e Ramakrishna (2013) propõem um sistema de detecção de intrusão capaz de capturar dados diretamente da rede, permitindo o sistema atuar como um IDS em tempo real. Para testar o sistema, foi utilizada a base de dados KDD'99 (Information and Computer Science University of California, 1999), amplamente conhecida e testada na comunidade, gerando um fluxo de pacotes em sequência. A técnica utilizada neste modelo para a detecção é baseada em assinaturas. O modelo não possui processamento em paralelo.

Ainda em 2013, o trabalho de Chae *et al.* (2013) mostram que entre a precisão e o valor Razão de Atributo, do inglês *Attribute Ratio* (AR), existe uma correlação inversa no método de seleção de características, sendo a maior precisão de 99,79% quando se utilizam 22 características. A precisão do método é maior do que a precisão dos dados completos e também é tão alta quanto a precisão de outros métodos.

Em 2015, Jaswal, Kumar e Rawat (2015) apresentam um sistema híbrido que utiliza as técnicas de mineração *k-Means*, SVM e regras de associação em arquivos de *logs* da rede para avaliar a eficiência de algoritmos híbridos em detectar diversos tipos de ataques. Os resultados mostram que o número de ameaças detectadas na base de dados é maior que o apresentado em trabalhos anteriores que utilizam apenas uma técnica.

Também em 2015, Desale, Kumathekar e Chavan (2015) discutem técnicas de classificação e melhoria de desempenho em sistemas de detecção de intrusão utilizando diferentes classificadores. Os resultados gerados pela análise dos classificadores mostram que Naive Bayes e Árvore de Hoeffding apresentam os melhores resultados. Em termos de acurácia, Naive Bayes apresenta melhor resultado, porém leva mais tempo para classificar que a Árvore de Hoeffding.

Em 2016, no trabalho apresentado por Lobato, Andreoni e Duarte (2016), é proposto um sistema de detecção de ameaças em tempo real por processamento de dados utilizando a arquitetura *Lambda* (MARZ NATHAN; WARREN, 2013). Os resultados mostram uma acurácia com valores acima de 95% e taxa de falsos-positivos abaixo de 6%, analisando até 3,57 milhões de amostras por minuto.

Ainda em 2016, Rathore *et al.* (2016) propõem um IDS utilizando *Map-Reduce* e usando o *Hadoop* para processar um arquivo sequencial e calcular os valores dos parâmetros a serem utilizados em diversos classificadores. A utilização de um número menor de parâmetros para a classificação e o uso de um ambiente paralelo do *Hadoop* permitiu o processamento de grandes bases de dados em um período de tempo menor que outros IDS propostos, com similar eficiência e precisão. O sistema apresentado utiliza uma base de dados previamente coletada e processamento em lotes.

Em 2016, Lopez, Mattos e Duarte (2016) propõem um Sistema de Detecção de Intrusão Baseado em Rede, do inglês *Network Intrusion Detection System* (NIDS) baseado no analisador de tráfego *Bro* e na visão global de rede provida pela ferramenta *OpenFlow*. Esta proposta é denominada *BroFlow*, capaz de detectar ataques de negação de serviço em tempo real através de algoritmos simples implementados; reação imediata aos ataques detectados; e posicionamento estratégico de sensores em um infraestrutura de rede através de heurísticas propostas, de maneira quase-ótima. Um protótipo do sistema é desenvolvido e testado, mostrando que o *BroFlow* garante o encaminhamento de pacotes legítimos ao máximo da taxa de transferência, reduzindo em até 90% o atraso de rede causado por um ataque.

Também em 2016, Lopez, Lobato e Duarte (2016) apresentam um trabalho que descreve e analisa as três principais plataformas de processamento de fluxo distribuído de código aberto: *Storm* (Apache Software Foundation, 2015), *Flink* (Apache Software Foundation, 2014) e *Spark*

(Apache Software Foundation, 2013). Utilizando dois experimentos para detecção de anomalias em tráfego de redes, foram analisadas a eficiência de processamento e a resiliência à falha dos nodos, onde o *Storm* e o *Flink* apresentaram desempenho 15 vezes maior que o *Spark*, o qual, por sua vez, demonstrou ser mais robusto às falhas de nodos.

Em 2017, Andreoni *et al.* (2017) apresentam a ferramenta *CATRACA*, um sistema NIDS *online* implementado como uma Função Virtualizada de Rede, do inglês *Network Virtualized Function* (NVF). A ferramenta é executada sobre a plataforma Plataforma Aberta para Função Virtualizada de Rede, do inglês *Open Platform for Network Functions Virtualization* (OPNVF) e é baseada no processamento de fluxos de *Big Data*, resultando em um serviço acurado de detecção de ameaças em tempo real. Ao detectar uma ameaça, o sistema é capaz de reagir e criar regras de bloqueio para o *firewall*. O processo de detecção diferencia tráfego normal de ataques DoS e varreduras.

Ainda em 2017, um sistema de detecção de intrusão em redes NIDS baseado em anomalias foi construído por Van, Thinh e Sach (2017) utilizando técnicas de aprendizagem profunda (*deep learning*). Essas técnicas mostraram o grande poder dos modelos generativos com boa classificação, capazes de deduzir parte do seu conhecimento de dados incompletos e adaptar-se. O trabalho foi capaz de detectar intrusões baseadas em anomalias e classificá-las em cinco grupos, com acurácia baseada nas fontes de dados de rede.

Em 2017, Lopez *et al.* (2017) analisam e classificam o tráfego de usuários residenciais de uma rede de acesso à *Internet* de uma grande operadora de telecomunicações, pelo período de uma semana, obtendo o perfil dos alarmes de segurança gerados por um NIDS. O trabalho apresenta a classificação de alertas com uma sensibilidade de 93% na diferenciação dos fluxos legítimos e anômalos, validando a base de dados coletada.

Também em 2017, no trabalho de Kim e Aminanto (2017) são mostradas algumas limitações em IDS anteriores que utilizam aprendizagem de máquina clássicas e introduzem o aprendizado de características, incluindo a construção, extração e seleção de características para superar os desafios. Também discutem algumas técnicas de *deep learning* e suas aplicações para utilização em IDS.

Em 2017, Alom e Taha (2017) apresentam um trabalho sobre IDS utilizando técnicas de *deep learning* não-supervisionadas. As amostras de entrada são codificadas numericamente para a aplicação de técnicas não-supervisionadas, como Auto-codificador, do inglês *Autoencoder* (AE) e Máquina de Boltzmann Restrita, do inglês *Restricted Boltzmann Machine* (RBM), para extração de características e redução de dimensionalidade. Então é aplicado agrupamento iterativo *k-means* para aglomeração em um espaço dimensional menor com apenas 3 atributos.

Ainda em 2017, Schuartz, Fonseca e Munaretto (2017) propõem uma plataforma distribuída para detecção de ameaças em tempo real por análise de fluxos. O sistema proposto trabalha com diferentes algoritmos de aprendizado de máquina em cima de uma plataforma distribuída, permitindo diversos classificadores operarem simultaneamente. Os resultados obtidos

mostram uma acurácia média acima de 90% e tempo de detecção aproximado de $95\mu s$ por fluxo.

Em 2017, CorrÊa, Enembreck e Silla (2017) apresentam um estudo sobre a utilização da técnica de mineração de dados conhecida como Árvore Adaptativa de Hoeffding, do inglês *Hoeffding Adaptive Tree* (HAT) para criar um modelo de previsão com objetivo de detectar intrusões em uma rede. Os resultados mostram uma acurácia no modelo de previsão do HAT em torno de 95%, com tempo de processamento de cada amostra na base de dados sendo aproximadamente $100\mu s$.

Também em 2017, Schuartz, Fonseca e Fonseca (2017) apresentam uma proposta de uma plataforma distribuída para detecção de ameaças em tempo real, utilizando *big data*. A plataforma proposta é capaz de detectar diversos tipos de ataques com precisão acima de 90% e baixo número de falsos-positivos e falsos-negativos. O sistema utiliza 41 atributos da base de dados para o treinamento e detecção de ameaças, resultando na detecção de ataques específicos, devido ao treinamento imparcial dos diversos tipos de ataques e pela representatividade de tais ataques dentro da base de dados.

Em 2018, Shone *et al.* (2018) propõem uma técnica de *deep learning* para detecção de intrusões utilizando um Auto-codificador Não-simétrico de Profundidade, do inglês *Nonsymmetric Deep Autoencoder* (NDAE) para aprendizado de características não-supervisionadas. O modelo de classificação proposto foi desenvolvido utilizando o *TensorFlow* e uma Unidade de Processamento Gráfica, do inglês *Graphics Processing Unit* (GPU) e foi avaliada através das bases de dados *KDD'99* e *NSL-KDD* (TAVALLAEE *et al.*, 2009).

Em 2018, Lopez *et al.* (2016) propõem uma função de rede virtualizada em uma plataforma de código aberto para um serviço de detecção de ameaças em tempo real. A função combina computação em nuvem e técnicas de processamento de fluxos distribuídos para detectar rapidamente ameaças. Os resultados mostram que a função proposta é capaz de escalonar dinamicamente, analisando mais do que cinco milhões de mensagens por segundo.

Em 2019, Schuartz, Fonseca e Munaretto (2019) apresentam uma proposta de plataforma distribuída para detecção de ameaças em tempo real usando *Big Data*. A plataforma proposta compara a capacidade de detecção de ameaças através do processamento por fluxos em relação ao processamento por lotes. São utilizados três algoritmos de aprendizado de máquina, tanto no modo de processamento em lotes como no processamento em fluxos. A base de dados é avaliada utilizando todas as características extraídas, e ainda executando uma redução no espaço para apenas 10 características.

Em 2020, Alghushairy, Alsini e Ma (2020) propõem dois métodos que visam aumentar a eficiência na detecção de valores discrepantes locais em fluxos de dados através da redução na pontuação de discrepâncias. Ao detectar valores discrepantes, informações essenciais podem ser obtidas para tomar melhores decisões em diversas aplicações. Entretanto, os dados não podem ser processados inteiramente na memória do computador porque o volume de dados

continuam aumentando indefinidamente. Os métodos propostos utilizam mineração de fluxo de dados, fornecendo melhor precisão de detecção de discrepâncias.

Em 2021, Seth, Singh e Chahal (2021) propõem uma abordagem adaptativa para detecção de intrusão usando aprendizagem orientada a fluxos, adaptando ao Desvio de Conceito, do inglês *Concept Drift* (CD)[1] no ambiente do mundo real. O classificador *Adaptive Random Forest* com detector de mudança ADWIN é usado para detectar mudanças em um fluxo de dados e se adaptar à detecção de desvios nos dados transmitidos, resultando em adaptação ágil contra intrusões desconhecidas. A abordagem proposta também supera a necessidade de retreinar o modelo com o tempo.

Em 2022, Gadal *et al.* (2022) apresentam um método híbrido para detecção de anomalias utilizando formulação *K-mean cluster* e categorização Otimização Mínima Sequencial, do inglês *Sequential Minimal Optimization* (SMO). O SMO utiliza a seleção de recursos nos estágios de pré-processamento para melhorar o conjunto de dados. O nível de subconjunto de consistência e o algoritmo de busca genética foram usados para escolher determinados recursos do conjunto de dados NLS-KDD e eliminar os recursos que são inadequados para o processo antes das etapas de agrupamento e categorização. Em seguida, K-means foi usado para agrupamento para eliminar o treinamento dos conjuntos de dados de treinamento, mantendo o tempo de processamento abaixo de um determinado limite.

A Tabela 1 apresenta um resumo comparativo das principais características dos trabalhos relevantes e da proposta apresentada neste trabalho. Para realizar este comparativo, foram consideradas as seguintes características: capacidade de detecção de ameças por assinaturas [1], capacidade de detecção de ameaças por anomalias [2], possui processamento distribuído pela rede [3], capaz de detectar ameaças em tempo real [4], capaz de manusear grandes quantidades de dados (*Big Data*) [5], possui mais de um método de classificação de ameaças [6], realiza uma redução na dimensionalidade das características dos dados [7] e possui aprendizagem em profundidade (*deep learning*) [8].

[1] Desvio de conceito significa que as propriedades estatísticas da variável de destino, que o modelo está tentando prever, mudam ao longo do tempo de maneiras imprevistas. Se ocorrer um desvio de conceito, o padrão induzido de dados anteriores pode não ser relevante para os novos dados, levando a previsões e resultados de decisão ruins (WIDMER; KUBAT, 1996).

Tabela 1 – Resumo comparativo das principais características dos trabalhos relevantes e da proposta deste trabalho. [1] Detecção de ameaças por assinaturas, [2] Detecção de ameaças por anomalias, [3] Processamento distribuído pela rede, [4] Detecta ameaças em tempo real, [5] Manuseio de (*Big Data*), [6] Mais de um método de classificação de ameaças, [7] Redução na dimensionalidade das características dos dados, [8] *Deep learning*.

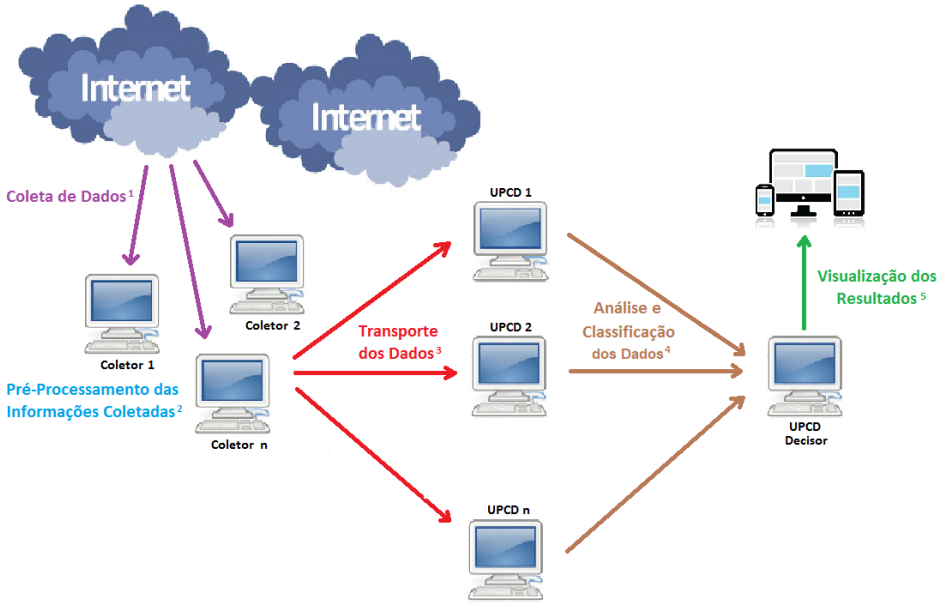
Atividade	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
(BALUPARI *et al.*, 2003)		X		X				
(HOLTZ; DAVID; JÚNIOR, 2011)	X		X		X		X	
(DEVIKRISHNA; RAMAKRISHNA, 2013)	X							
(JASWAL; KUMAR; RAWAT, 2015)	X					X		
(DESALE; KUMATHEKAR; CHAVAN, 2015)	X					X		
(LOBATO; ANDREONI; DUARTE, 2016)	X	X	X	X				
(RATHORE *et al.*, 2016)					X		X	
(LOPEZ; MATTOS; DUARTE, 2016)		X	X	X	X			
(LOPEZ; LOBATO; DUARTE, 2016)								
(ANDREONI *et al.*, 2017)	X		X	X	X		X	
(VAN; THINH; SACH, 2017)		X						X
(LOPEZ *et al.*, 2017)	X		X		X			
(KIM; AMINANTO, 2017)							X	X
(ALOM; TAHA, 2017)							X	X
(SCHUARTZ; FONSECA; MUNARETTO, 2017)	X		X	X	X	X		
(CORRÊA; ENEMBRECK; SILLA, 2017)	X			X	X			
(SCHUARTZ; FONSECA; FONSECA, 2017)	X		X	X	X	X		
(SHONE *et al.*, 2018)	X						X	X
(LOPEZ *et al.*, 2016)	X		X	X	X		X	
(SCHUARTZ; FONSECA; MUNARETTO, 2019)	X		X	X	X	X	X	
(ALGHUSHAIRY; ALSINI; MA, 2020)		X		X	X			
(SETH; SINGH; CHAHAL, 2021)		X		X	X			
(GADAL *et al.*, 2022)		X		X	X	X	X	
Arquitetura Proposta	X		X	X	X	X	X	X

Fonte: Autoria própria..

3 MODELO DA ARQUITETURA PROPOSTA

A arquitetura proposta nesta tese visa construir uma plataforma aberta e distribuída para coleta, distribuição, análise e classificação de dados, em grandes quantidades, com o objetivo de detectar ameaças que trafegam pela rede, em tempo real.

Figura 2 – Visualização geral da arquitetura proposta, composta em cinco módulos: (1) coleta de dados, (2) pré-processamento das informações coletadas, (3) transporte dos dados, (4) análise e classificação dos dados e (5) visualização dos resultados.



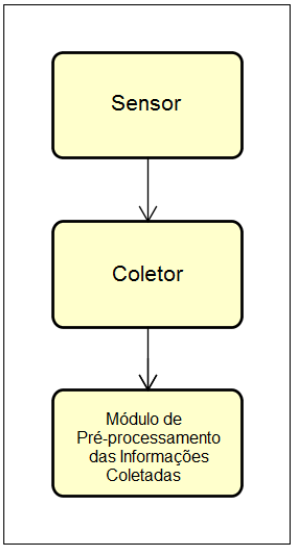
Fonte: Autoria própria..

A Figura 2 apresenta a arquitetura proposta neste trabalho, que é composta por cinco módulos detalhados a seguir: (1) módulo de coleta de dados; (2) módulo de pré-processamento das informações coletadas; (3) módulo de transporte dos dados; (4) módulo de análise e classificação dos dados e (5) visualização dos resultados.

3.1 Módulo de Coleta de Dados

A primeira parte da arquitetura proposta tem como objetivo trabalhar com os dados que trafegam na rede. Antes do sistema conseguir analisar o comportamento da rede e procurar por invasões e anomalias, é necessário coletar os dados provenientes da rede. O sensor e o coletor são dois componentes importantes de um sistema de detecção de ameaças. O sensor é responsável por coletar dados do ambiente, enquanto o coletor é responsável por armazenar e processar esses dados. A Figura 3 mostra os procedimentos executados dentro do módulo de Coleta de Dados.

Figura 3 – Módulo de Coleta de Dados. Os dados são coletados da rede através de arquivos *dump*.



Fonte: Autoria própria..

3.1.1 Sensor

O sensor é um elemento básico de monitoramento e normalmente coleta valores medidos na rede, como, por exemplo, o tráfego de uma porta de comunicação, a carga da CPU de um servidor ou o espaço livre de uma unidade de disco. Um exemplo de sensor é o *Packet Sniffing*, que permite monitorar o tráfego aberto, não cifrado, *web*, de *e-mails*, de transferência de arquivos, entre outros. O sensor apenas analisa os cabeçalhos dos pacotes de dados e não analisa a carga, reduzindo, assim, o tempo para coleta de informações do fluxo (ANSARI; RAJEEV; CHANDRASHEKAR, 2003).

3.1.2 Coletor

O objetivo do coletor é realizar a coleta de dados que trafegam na rede. Os dados são capturados de fontes independentes, em diversos locais da rede. Estas fontes são, em geral, estruturas por onde passa um grande tráfego de informações, como Pontos de Troca de Tráfego (PTT). Esses dados podem ser coletados diretamente do equipamento roteador, através da captura dos pacotes *Ethernet* que passam pelas interfaces do sistema e de *logs* do sistema operacional e outras aplicações em servidores.

Atualmente, os roteadores contam com a funcionalidade Captura de Pacote Embarcado, do inglês *Embedded Packet Capture* (EPC), que basicamente permite o roteador trabalhar como um *sniffer*. Com esta funcionalidade, pode-se configurar o roteador para capturar os pacotes que entram e/ou saem de uma interface e exportá-los para serem analisados posteriormente (HSU; WANG, 2013).

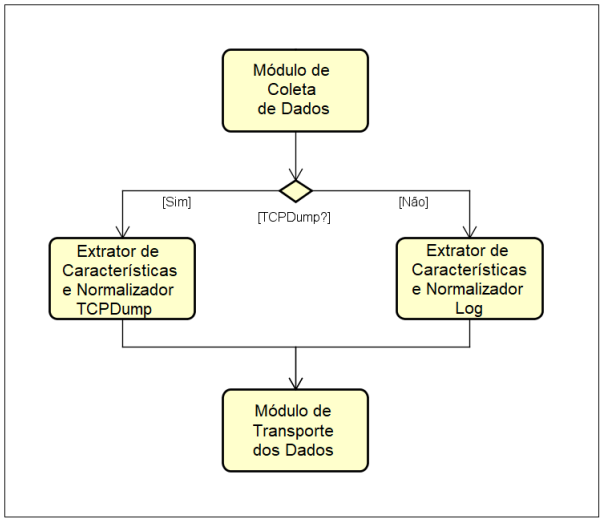
Nos sistemas computacionais, os intrusos deixam rastros nos registros de atividades do sistema, o qual denominamos de *logs*. Os *logs* podem desempenhar um papel preventivo, na medida em que podem registrar as eventuais tentativas de ataque (MASUD *et al.*, 2008). Em um sistema Linux, algumas ferramentas auxiliam na coleta dos *logs*, tais como: *TCP Wrappers*, *portmap/rpcbind*, *logdaemon*, *PingLogger* e *Smurflog*.

Após coletados os dados, é necessário enviá-los para o pré-processamento, onde serão tratados para posterior análise. Dependendo do tipo de dado coletado (pacotes ou *logs*), é tomada a decisão se os dados são enviados para o Normalizador *TCPDump* (pacotes) ou para o Normalizador *Log* (arquivos de *log*).

3.2 Módulo de Pré-processamento das Informações Coletadas

Neste módulo, os dados coletados da rede são recebidos e processados com o objetivo de extrair as características relevantes, sendo normalizados e distribuídos, na forma de fluxos de amostras, para serem classificadas pelo sistema. A normalização de dados é um processo de organização de dados de forma que sejam consistentes e fáceis de entender, e pode ajudar a identificar padrões e tendências nos dados. A Figura 4 mostra os procedimentos executados dentro do módulo de Pré-processamento das Informações Coletadas.

Figura 4 – Módulo de Pré-processamento das Informações Coletadas. As características são extraídas, normalizadas e transformadas em fluxos de amostras.



Fonte: Autoria própria..

3.2.1 Extrator de Características e Normalizador - *TCPDump* e *Logs*

Cada sequência de amostras coletada da rede possui seu próprio conjunto de sensor, coletor, extrator de características, normalizador e agente de mensagens. Este conjunto executa em uma máquina no local de coleta dos dados.

O normalizador funciona calculando a magnitude de cada valor em um conjunto de dados e, em seguida, redimensionando os valores para que todos tenham a mesma magnitude. Isso significa que, depois de normalizar os dados, todos os valores estarão na mesma escala.

Ao receber as sequências de amostras proveniente do normalizador, o agente de mensagens os publica em uma fila, conhecida pelas diversas UPCDs do projeto, localizadas em máquinas em locais diferentes da rede, ou até mesmo no mesmo local do agente, independente da posição geográfica.

Os dados coletados contém diversas informações, como mensagens enviadas pelo Protocolo de Mensagens de Controle da Internet, do inglês *Internet Control Message Protocol* (ICMP) e Protocolo de Resolução de Endereços, do inglês *Address Resolution Protocol* (ARP), entre outros. A ideia central de usar técnicas de extração de características é que os dados contêm algumas informações que são redundantes ou irrelevantes, e podem, assim, ser removidas sem impactar em muita perda de informação (KHALID; KHALIL; NASREEN, 2014).

Para trabalhar os dados coletados nos algoritmos de aprendizagem de máquina, é necessário obter informações através da extração de dados dos cabeçalhos dos pacotes, utilizando janelas de tempo suficiente para agregar informações para a composição dos fluxos de amostras. Os fluxos podem ser definidos por uma sequência de pacotes que possuem um mesmo Protocolo de Internet, do inglês *Internet Protocol* (IP), seja um mesmo Endereço de Internet, do inglês *Internet Address* (IA) de origem, um mesmo IA de destino e/ou uma mesma porta, por exemplo.

Inicialmente é realizada uma etapa de extração de características, com o objetivo de simplificar o conjunto de dados. Assim, utilizando uma janela de tempo, em segundos, é possível examinar os cabeçalhos dos pacotes TCP/IP que pertencem ao intervalo e extrair *N* características a respeito daquele intervalo. Por exemplo, é possível contar a quantidade de portas com mesmo origem e destino, ou o número de pacotes de Protocolo de Controle de Transmissão, do inglês *Transmission Control Protocol* (TCP), Protocolo de Datagrama do Usuário, do inglês *User Datagram Protocol* (UDP) e ICMP dentro daquele intervalo de tempo (SI; LI; HUANG, 2020; YANG; KPOTUFE; FEAMSTER, 2020). Esta redução, chamada de Seleção de Características, do inglês *Feature Selection* (FS) ou Seleção de Atributos, do inglês *Attribute Selection* (AS) é o processo no qual automaticamente procuramos pelo melhor sub-conjunto de características na base de dados.

O pré-processamento dos dados no Coletor permite transportar informações de maneira resumida, evitando consumir banda extra enviando informações desnecessárias à UPCD. As informações coletadas da internet são transformadas em um fluxo de amostras calculadas a partir

de um intervalo de tempo de dois segundos (valor comumente adotado pela comunidade), onde diversas características são extraídas. Essas sequências de amostras, então, são enviadas para o agente de mensagens.

3.3 Módulo de Transporte dos Dados

Este módulo é responsável por transportar o fluxo de amostras provenientes do módulo de Pré-processamento das Informações Coletadas para o módulo de Análise e Classificação dos Dados. Considerando que a coleta dos dados pode ser efetuada em locais distintos da rede e que estes pontos de coleta podem (ou não) estar em locais físicos diferentes de onde os dados serão processados, torna-se necessário utilizar um sistema de agente de mensagens.

3.3.1 Agente de Mensagens - Publicador

Para transportar os dados de um ponto ao outro na rede, é utilizado um agente de mensagens, isto é, uma ferramenta para a publicação e assinatura (*publish and subscribe*) de uma fila de mensagens. Cada módulo de Pré-processamento das Informações Coletadas produz um fluxo de amostras através de dados coletados em pontos diferentes da rede e o publica em uma fila. Cada UPCD escolhe quais filas assinar, podendo ele escolher uma ou mais filas (KUL; SAYAR, 2021).

Cada agente de mensagens utiliza apenas um tópico para publicação dos fluxos de amostras. Cada registro será distribuído para todos os assinantes daquele publicador. Assim, os fluxos de amostras são transportados do local onde o Sensor e o Coletor se encontram, para diversos locais da rede, utilizando os recursos oferecidos pelo próprio agente de mensagens.

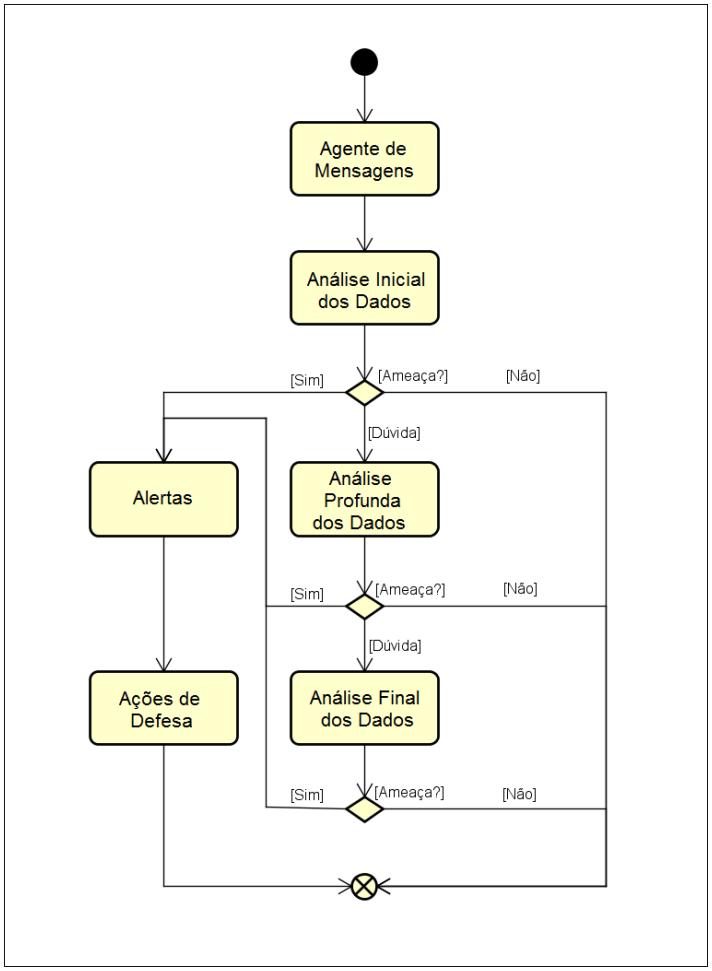
3.4 Módulo de Análise e Classificação dos Dados

Este módulo é composto por uma unidade de processamento e classificação do fluxo de amostras. As amostras são recebidas através de um ou mais agentes de mensagens e classificadas por diversos métodos de aprendizagem de máquina. Uma visão geral do módulo de Análise e Classificação dos Dados é apresentada na Figura 5.

3.4.1 Agente de Mensagens - Assinante

A primeira função executada dentro do módulo é de receber as informações provenientes de um fluxo de amostras criadas pelo coletor e publicadas na rede. Esta operação é realizada através da estrutura de transporte *publisher-subscriber*. Executando em uma máquina independente da localização (ou, caso necessário, no mesmo local de origem dos dados), cada UPCD

Figura 5 – Representação do Módulo de Análise e Classificação dos Dados. O agente de mensagens recebe as informações, que são analisadas pela AID. Caso haja necessidade, a UPCD pode executar análises mais profundas sobre os dados. Existindo uma ameça, alertas e ações de defesa são empregados.



Fonte: Fonte própria..

possui seu próprio agente de mensagens, configurado para assinar uma ou mais publicações dos fluxos de amostras. Inicialmente, cada UPCD assina apenas uma fonte de dados e toda vez que uma nova amostra é publicada pela fonte assinada, o agente de mensagens da UPCD recebe a informação através do sistema próprio de publicação/assinatura do agente.

O agente também executa a função de *buffer*, com relação aos dados recebidos. Caso exista um fluxo muito intenso de amostras sendo publicado, que seja superior à capacidade de análise da UPCD, essas amostras são temporariamente armazenadas em uma fila interna pelo próprio agente de mensagens, garantindo assim que todas as amostras recebidas serão processadas, sem ocorrer perda de informações.

A velocidade do fluxo de amostras que passa pela UPCD é controlada pela AID. Normalmente toda amostra recebida é imediatamente enviada à AID. Caso exista sobrecarga no tráfego de dados, o assinante armazena as amostras recebidas em uma fila de espera. Caso o

tráfego de dados vindo de um assinante mantenha-se intenso ao ponto de sobrecarregar a fila de espera, é colocado em execução uma nova UPCD para auxiliar na classificação dos dados, ou solicitado ajuda a outra UPCD no processamento das informações.

Ainda, o agente de mensagens pode assinar ou remover a assinatura de outras fontes de dados de acordo com a solicitação de uma unidade AFD, caso exista a necessidade. Assim, uma UPCD pode auxiliar temporariamente outras UPCD com a análise do tráfego de rede. Esta decisão é tomada por um Decisor quando a acurácia da classificação está abaixo de um determinado valor pré-estabelecido (necessitando assinar mais fontes de amostras) ou acima de um outro valor pré-estabelecido (não é mais necessário assinar outras fontes de amostras). Esses valores podem ser modificados pela UPCD em tempo real.

3.4.2 Unidade de Processamento e Classificação de Dados - UPCD

O módulo de Análise e Classificação dos Dados é composto por uma ou mais UPCDs. Estas unidades recebem o fluxo de amostras provenientes do agente de mensagens e realizam o processamento à procura de ameaças. O projeto proposto executa diversas UPCDs em diversos locais da rede, cada uma podendo estar em diferentes distâncias do Sensor, todas trabalhando em paralelo dentro de uma estrutura de processamento de fluxo distribuída para evitar problemas com a escalabilidade.

Uma UPCD pode receber informações de um ou mais agentes de mensagens, podendo aumentar ou diminuir as inscrições nos publicadores conforme a necessidade e a situação em que se encontra. Assim, consegue-se aprofundar o conhecimento em caso de dúvida sobre uma eventual ameaça que esteja ocorrendo ou devido a uma solicitação de outra UPCD para confirmar ou não um determinado tipo de ataque. Cada UPCD trabalha independente das outras UPCDs, porém podendo trocar informações caso exista a necessidade. Cada UPCD pode ser especializada em um determinado tipo de ataque, ou pode procurar pelo mesmo tipo de ataque, utilizando técnicas diferentes.

Inicialmente o agente de mensagens recebe o fluxo de amostras, que é analisado pela AID. Este primeiro passo tenta detectar, com rapidez, se existe uma ameaça na rede ou não. Caso exista, serão gerados alertas e tomadas ações defensivas definidas pelo administrador, como por exemplo, uma alteração nas regras do *firewall*. Não detectada uma ameaça, a UPCD retorna ao início do ciclo, recebendo a próxima informação do publicador e repetindo a operação. Entretanto, a UPCD pode não estar segura se os dados representam uma ameaça ou não. Nesse caso de dúvida, ou seja, quando a acurácia de classificação está abaixo de um valor pré-determinado e configurável em tempo de execução, a amostra vai para uma segunda camada de processamento, a APD. Igualmente à AID, caso seja detectada uma ameaça, alertas e ações defensivas serão executadas conforme programadas pelo administrador do sistema. Caso negativo, a UPCD retorna ao início e recebe a próxima informação a ser analisada. Novamente, no caso de incerteza (acurácia de classificação abaixo de um valor pré-definido), o dados são

encaminhados a uma terceira unidade, chamada de Análise Final dos Dados (AFD). Esta unidade é responsável pela decisão final se existe um ataque ocorrendo na rede ou não. Caso exista, alertas e ações defensivas definidas pelo administrador são acionadas, caso contrário, retoma-se ao início para receber a próxima amostra a ser classificada. Na AFD, a classificação é binária, ou seja, a amostra analisada será classificada como normal ou ataque. Caso ainda exista dúvida sobre a classificação, a AFD irá tomar uma decisão final baseada em regras pré-estabelecidas.

Cada uma das unidades AID, APD e AFD utiliza uma quantidade pré-estabelecida de características extraídas dos dados para tentar classificar o fluxo de amostras, criando uma análise em profundidade, quando necessário. Devido cada unidade de classificação utilizar elementos diferentes para classificação, veremos a seguir uma visão mais aprofundada de cada unidade. O objetivo desta hierarquia é ter resultados mais rápidos para ataques mais simples e utilizar técnicas mais complexas para problemas mais complexos quando necessário. Cada módulo será explicado com mais detalhes.

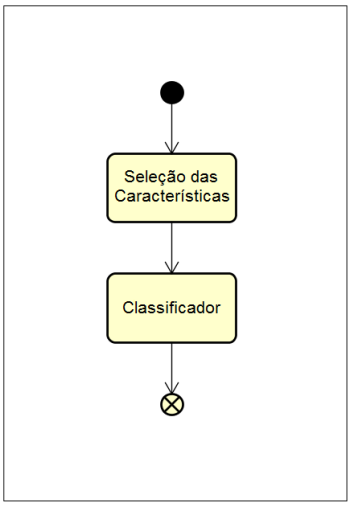
3.4.3 Análise Inicial dos Dados - AID

A primeira tentativa de classificação dos dados ocorre na unidade chamada Análise Inicial dos Dados (AID). Com o objetivo de rapidamente detectar se uma amostra representa uma ameaça ou não, a AID realiza um processo de seleção das características mais importantes uma primeira vez. As características encontram-se ordenadas em uma lista, de acordo com seu grau de relevância para detecção de ataques. A AID irá selecionar um número reduzido das características que são mais importantes e então, iniciar o processo de análise utilizando um classificador. O número de características selecionadas para classificação pode ser alterado de acordo com a necessidade de redução no tempo de detecção em detrimento da acurácia, ou vice-versa. O classificador irá gerar um resultado que corresponde a uma ameaça ou normal, com um determinado grau de acurácia. A Figura 6 mostra os procedimentos executados dentro da AID.

Como cada UPCD pode ser especializada em um tipo de ataque, durante o treinamento do classificador, é realizado um pré-processamento para classificação dos dados, resultando na ordenação das características mais relevantes ao ataque a ser detectado. Assim, quando uma amostra é recebida do agente de mensagens, somente as características mais relevantes serão utilizadas para classificá-la. Esta redução não ocorre no Coletor pois a UPCD necessita classificar as amostras com diferentes quantidades de características para obter uma maior relação acurácia/tempo e a retransmissão das mesmas amostras com número diferentes de características em média acaba aumentando a quantidade de dados transportados em diversas situações.

Após a classificação ser feita, o resultado indicará se a amostra é uma ameaça ou não, com um determinado grau de confiança. Se o classificador estiver seguro que a amostra não

Figura 6 – A AID escolhe um número reduzido de características para classificar a amostra recebida. A seguir, é realizada a classificação da amostra em ataque ou normal, com um grau de acurácia.



Fonte: Autoria própria..

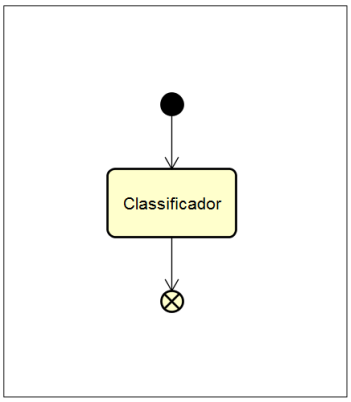
representa uma ameaça, a AID receberá a próxima amostra do agente de mensagens e o processo repete. Caso o classificador esteja seguro que a amostra representa uma ameaça, então a AID envia a informação à unidade de Alerta, para então retornar ao início do ciclo, analisando a próxima amostra recebida. Entretanto, pode acontecer do classificador não estar confiante no resultado da classificação, pois o grau de certeza está abaixo de determinado valor pré-definido. Nesse caso, a AID está em dúvida e é necessário passar a amostra para a próxima camada de classificação, para tentar decidir se representa uma ameaça ou não. Após decidido se a amostra representa uma ameaça ou não, a AID retorna ao início, pronto para classificar o próximo dado recebido do agente de mensagens.

3.4.4 Análise Profunda dos Dados - APD

A segunda unidade que tenta classificar a amostra recebida é a Análise Profunda dos Dados (APD). Nesta etapa, o classificador tenta utilizar todas as características para determinar se a amostra é uma ameaça ou não. O uso de um número maior de características requer um tempo maior de processamento, porém aumenta a acurácia de classificação e reduz o número de falsos-positivos e falsos-negativos. A Figura 7 mostra os procedimentos executados dentro da APD.

Semelhante à AID, o dado é processado por um classificador, resultando em uma classificação como ameaça ou normal, com um determinado grau de confiança. No caso do grau de confiança ser maior que um valor pré-estabelecido, o procedimento seguinte é igual ao da AID. Se não existir ameaça, ele retorna ao início para processar o próximo dado. Se existir uma ameaça, ele avisa a unidade de Alertas e depois retorna ao início. Entretanto, caso a APD esteja

Figura 7 – A APD utiliza todas as características para analisar a amostra recebida. O resultado da classificação é um ataque ou normal.



Fonte: Autoria própria..

ainda em dúvida se o dado é uma ameaça ou não, ele encaminha o dado para uma terceira e final unidade de classificação.

3.4.5 Análise Final dos Dados - AFD

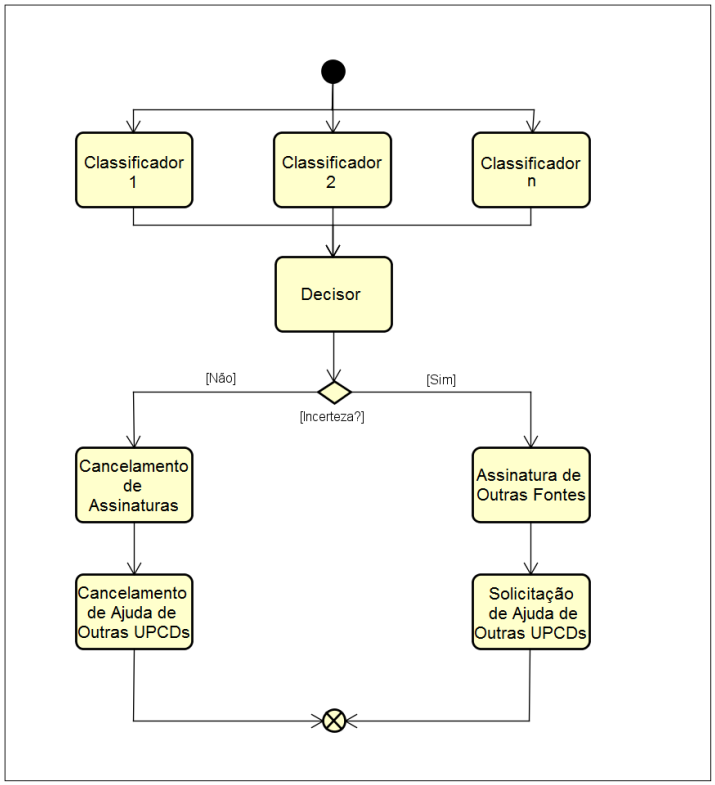
Abordagens utilizando diversos classificadores utilizam coleções de algoritmos de aprendizagem de máquina para obter um desempenho preditivo mais alto do que poderia ser obtido a partir de um único classificador de aprendizagem de máquina (RYU; KANTARDZIC; WALGAMPAYA, 2010). A ideia principal de uma abordagem com diversos classificadores é combinar vários algoritmos de aprendizagem de máquina para explorar os pontos fortes de cada algoritmo empregado, obtendo assim um classificador mais poderoso.

No mundo dos ataques de rede, uma vez que as assinaturas de diferentes ataques são bastante distintas umas das outras, é normal ter diferentes conjuntos de características, bem como diferentes algoritmos de aprendizagem de máquina para detectar diferentes tipos de ataques. Um único IDS não pode cobrir todos os tipos de dados de entrada ou identificar diferentes tipos de ataques com alta acurácia (ELMOMEN; DIN; WAHDAN, 2011; DURAISAMY; CHAKRABARTI; MIDHUNCHAKKARAVARTHY, 2018). Muitos pesquisadores têm mostrado que problemas de classificação podem ser resolvidos com alta precisão ao usar modelos combinando diversos classificadores em vez de classificadores únicos (HANSEN; SALAMON, 1990).

Nesta unidade tenta-se classificar o dado de forma definitiva. A unidade Análise Final dos Dados (AFD) é composta de diversos classificadores e um Decisor. A amostra recebida é encaminhada para os diferentes classificadores, que utilizam métodos diferentes de aprendizagem de máquina para classificação de dados. Cada classificador obterá uma resposta se uma amostra é um ataque ou não, com determinado grau de acurácia. Estes n resultados, e seus graus respectivos de incerteza, são encaminhados para um Decisor, que fará a classificação final através de uma votação, utilizando as acurácias como pesos na votação. Este procedimento

ocorre em paralelo e considera-se que existam recursos disponíveis para todos os módulos continuarem procurando ataques. A Figura 8 mostra os procedimentos executados dentro da AFD.

Figura 8 – A AFD utiliza todos os atributos da amostra recebida para realizar a classificação do dado através de classificadores distintos. Cada classificador envia sua resposta a um Decisor, que irá determinar se a amostra é um ataque ou normal.



Fonte: Autoria própria..

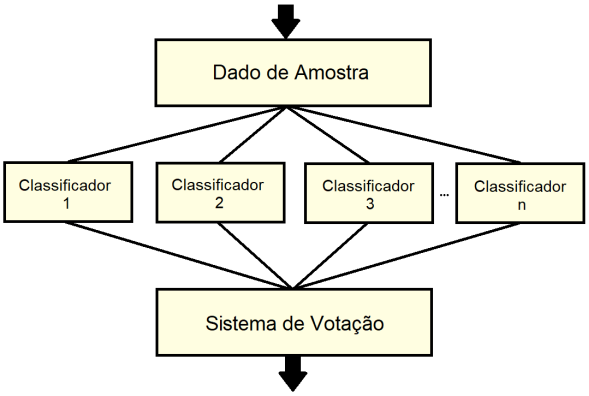
A AFD irá sempre obter como resposta se a amostra é um ataque ou não de acordo com um valor de decisão pré-estabelecido. Caso a incerteza esteja alta, mesmo tendo decidido se o dado é um ataque ou não, a AFD pode solicitar ao agente de mensagens assinar outras fontes de amostras ou pedir ajuda para outra UPCD para classificar, temporariamente, amostras da sua fonte de dados com o objetivo de verificar se existe, ou não, uma tentativa de intrusão naquele ponto da rede. Quando a incerteza estiver abaixo de determinado valor pré-determinado, a AFD pode solicitar ao agente de mensagens o cancelamento de assinaturas de outras fontes, ou solicitar outras UPCDs a não ajudar mais na classificação das amostras provenientes da sua fonte, evitando assim, o consumo de recursos extras da arquitetura proposta.

3.4.6 Decisor

O Decisor é a etapa final de classificação de uma amostra quando ainda não se obteve um grau de confiança desejável na classificação da mesma. O Decisor irá receber da AFD um

dado de amostra que foi classificado com baixo grau de confiança e irá submeter esta amostra para que cada classificador, separadamente, realize uma nova classificação. A classificação obtida, junto com a acurácia obtida por tal classificador durante sua fase de treinamento é, então, submetida ao Sistema de Votação para se obter um resultado definitivo. A combinação dos classificadores não requer nenhum conhecimento prévio do comportamento dos classificadores e não requer qualquer metodologia complexa para decidir. O método conta o número de classificadores que concordam em sua decisão e, assim, decide a classe para a qual o padrão de entrada pertence. Considerando que cada classificador possui uma acurácia devido ao seu treinamento, a decisão final também terá um grau de confiança, baseado nas acurácias dos classificadores que possuem a mesma classificação que o Decisor. O Decisor é apresentado na Figura 9.

Figura 9 – Representação do Decisor. Os classificadores recebem um dado de amostra proveniente da AFD, realizam a classificação e enviam o resultado e o grau de acurácia para o Sistema de Votação.



Fonte: Autoria própria..

Assim, resultados diferentes podem ser obtidos a partir do conjunto de diferentes classificadores, e esses resultados são enviados ao Sistema de Votação. Cada classificador tem um peso para denotar a contribuição do classificador para o Sistema de Votação. Para cada classe a ser identificada (ataque ou normal), uma soma ponderada dos classificadores pode ser calculada como:

$$V_i = \sum_{d=1}^{N} \alpha.w_d, \begin{cases} 1 & \text{se } C_d = i \\ 0 & \text{caso contrário} \end{cases} \tag{1}$$

onde N é o número de classificadores, $i = 1, 2, ...$ representa as classes, C é o rótulo da classe, C_d é o rótulo da classe prevista pelo classificador d, e w_d é o peso do classificador d. Para um padrão desconhecido, a classe final a ser classificada é determinada maximizando $arg\ max_{j=1}^{c} V_j$.

3.4.7 Alertas

A unidade responsável pelos alertas será ativada somente se uma amostra for classificada como um ataque. Ela irá enviar uma mensagem para os meios configurados pelo administrador contendo um aviso de ameaça detectada, qual tipo de ameaça e informações adicionais que ajudem a identificar o atacante, tal como endereço IP de origem. A seguir, é acionado a unidade Ações de Defesa.

3.4.8 Ações de Defesa

Esta unidade é responsável pelas ações automáticas a serem tomadas, de acordo com a configuração estipulada pelo administrador. Um exemplo de ação a ser tomada é o de solicitar a configuração do *firewall* na fonte para bloquear/ignorar pacotes cuja origem seja o IP de origem do ataque, evitando, assim, um ataque DDoS. Outras ações podem ser tomadas, dependendo do tipo de ataque detectado, podendo-se enviar mensagens para controladores Rede Definida por Software, do inglês *Software-Defined Networking* (SDN), caso esta tecnologia esteja disponível.

3.5 Módulo de Visualização dos Resultados

Este módulo é responsável por apresentar os resultados obtidos do sistema, tais como estatísticas da rede, totais de ameaças detectadas, tipos de ameaças, recomendações, ações tomadas, etc. Os relatórios podem ser visualizados em um monitor, enviados pela rede ou de outras formas que forem necessárias.

3.6 Conclusão do Capítulo

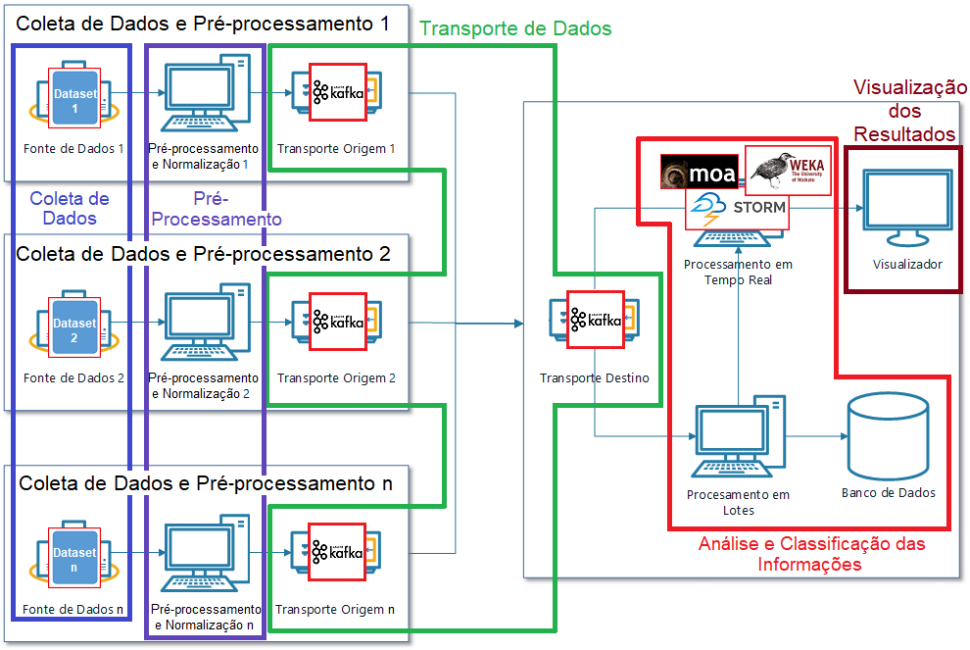
Neste capítulo apresentamos a visão completa da arquitetura proposta, que é composta de cinco estágios. Inicialmente recolhemos dados provenientes da rede, e a seguir, extraímos as características, normalizando os valores quando necessário. Assim, extraímos as características mais relevantes para cada tipo de ataque, construindo uma lista ordenada decrescente da importância das características. Então, as amostras são transportadas através de um agente de mensagens para as unidades classificadoras de dados, onde serão classificadas por uma ou mais unidades classificadoras trabalhando em paralelo. Por fim, as amostras classificadas são apresentadas e, caso necessário, ações são tomadas se amostras de ataques forem detectadas.

No capítulo 4 apresentamos o protótipo da arquitetura proposta. Utilizando *software* de código aberto, descrevemos as ferramentas utilizadas em cada estágio proposto. Ainda, descrevemos os classificadores, o processo de seleção de características e as bases de dados utilizadas para realizar a prova de conceito da arquitetura.

4 PROTÓTIPO DA ARQUITETURA PROPOSTA

O protótipo da arquitetura proposta para a detecção em tempo real de ameaças é composto de ferramentas de código aberto, distribuídas na *Internet*. A arquitetura utiliza o modelo de análise de dados em tempo real, composto por cinco etapas: coleta de dados, pré-processamento das informações coletadas, transporte dos dados, análise e classificação das informações e visualização dos resultados. A composição do protótipo consiste na integração das ferramentas, conforme ilustrado na Figura 10.

Figura 10 – Representação da arquitetura proposta. São utilizadas as ferramentas Apache Kafka, Apache Storm, WEKA e MOA, todas de código aberto, além das bases de dados KDD'99 e CICIDS2017.



Fonte: Autoria própria..

Inicialmente, o módulo Coleta de Dados obtém dados provenientes de pontos diferentes da rede através da leitura de arquivos contendo as bases de dados. Esses pacotes são enviados para o módulo de Pré-Processamento das Informações Coletadas, onde os pacotes são processados em janelas de tempo determinadas e caracterizados em fluxos de amostras, que serão posteriormente utilizados para detectar diversos tipos de ataques. No módulo de Transporte dos Dados, os fluxos de amostras são publicados na rede pela ferramenta Apache Kafka, permitindo que vários clientes possam assinar cada fluxo gerado, em separado. Em outros pontos da rede, utilizando uma topologia criada pela plataforma Storm, o módulo de Análise e Classificação das Informações, composto por Unidades de Processamento e Classificação de Dados (UPCD), recebe um ou mais fluxos de amostras publicados na rede, através da ferramenta Apache Kafka, no modo de assinante. Cada UPCD assina o fluxo de amostras que for de seu interesse. Dentro do Storm, são criados classificadores utilizando as ferramentas Weka e MOA. Cada UPCD é especializada em detectar um determinado tipo de ataque (por exemplo, DoD, varreduras, etc)

ou treinada para detectar diversos tipos de ataques contido na base de dados. O classificador irá processar o fluxo de amostras recebido e irá caracterizar o mesmo em um ataque ou em um fluxo normal de dados. Diversas camadas de classificadores são empregadas, utilizando diferentes quantidades de características, diferentes métodos de classificação, solicitando fontes adicionais de fluxos de amostras ou solicitando confirmação dos resultados incertos a outras UPCDs. Diferentes classificadores, dentro de uma mesma UPCD, irão encaminhar seus resultados para um Decisor, que determinará se a amostra representa um ataque ou não. O Decisor será empregado no caso de incerteza na classificação de alguma amostra durante as etapas anteriores. Sua função é coletar as informações de classificação dos diferentes algoritmos, dentro de uma mesma UPCD, e tomar a decisão final: se a amostra é uma atividade normal ou um tipo de ataque. O Decisor recebe o peso de cada um dos classificadores, ou seja, a acurácia que o algoritmo obteve ao aprender com a base de treinamento. Baseado nas informações recebidas, ele determina se a amostra é um determinado tipo de ataque ou não, dependendo de quanto cada classificador está seguro do seu resultado. Os resultados são exibidos pelo último módulo, a Visualização dos Resultados.

4.1 Apache Kafka

Como ferramenta para publicação e assinatura, optou-se pelo *Apache Kafka* (Apache Software Foundation, 2016), uma plataforma de *streaming* distribuída de software livre. O Kafka é projetado para sistemas distribuídos de alto desempenho e, comparado com outros sistemas de mensagens, possui melhor rendimento, *built-in* de particionamento, replicação e tolerância a falhas inerentes.

4.2 Apache Storm

O *Apache Storm* (Apache Software Foundation, 2015) é um Processador de Fluxo de Dados, do inglês *Data Streaming Processor* (DSP), isto é, um processador de eventos em tempo real. Ele é composto por topologias que formam Grafos Acíclicos Dirigidos (GAD) compostos por *spouts* (elementos de entrada) e *bolts* (elementos de processamento). Os *spouts* e *bolts* trabalham em paralelo, permitindo o processamento em paralelo de diferentes amostras de fluxos em tempo real. A topologia proposta funciona como um canal de dados, os quais são processados em forma de fluxos à medida que vão avançando.

As UPCDs são executadas em cima de uma topologia criada pelo *Apache Storm*, utilizando, assim, as ferramentas de comunicação do próprio Storm para realizar o transporte dos dados e das comunicações entre UPCDs. Nessa topologia, o assinante realiza a função de um *Spout*, enquanto as unidades AID, APD e AFD, Alertas e Ações de Defesa são *Bolts*, conforme a nomenclatura utilizada pela plataforma *Apache Storm*.

O *Apache Storm* é altamente escalável, possui uma solução tolerante a falhas e garante que os dados serão processados pela topologia, sem a perda dos mesmos.

4.3 Apache Zookeeper

Para coordenar o grupo de nodos (*cluster)* e manter os dados compartilhados, utilizando técnicas de sincronização robustas, será utilizada a ferramenta *Apache ZooKeeper* (Apache Software Foundation, 2010). O *Apache Storm* não possui estados, por isso depende do *Apache Zookeeper* para monitorar os estados dos nodos que estão trabalhando. Com isso, garante-se o processamento dos dados mesmo se algum dos nodos conectados no *cluster* morrer ou alguma mensagem for perdida.

4.4 Weka

O *Weka* (HALL *et al*., 2009) é um *software* de código aberto composto por uma coleção de algoritmos de aprendizagem de máquina utilizado para tarefas de mineração de dados. Ela contém ferramentas que podem ser aplicadas diretamente sobre uma base de dados ou ser incorporada em código *Java* para o pré-processamento de dados, classificação, regressão, clusterização, associação de regras e visualização dos resultados obtidos.

Essa classificação é realizada através da ferramenta *WEKA*, cuja biblioteca de funções e procedimentos para classificação são utilizados dentro do código *Java*. A unidade AID é representada por um *bolt* dentro da topologia do *Apache Storm*. Ela é composta por código *Java*, utilizando bibliotecas e funções tanto do *WEKA*, como do *Apache Storm*. A comunicação entre *bolts* é feita através da estrutura do *Apache Storm* e a classificação é executada através das funções do *WEKA*.

4.5 MOA

O aplicativo Análise Massiva Online, do inglês *Massive Online Analysis* (MOA) (BIFET *et al*., 2011) é um ambiente de *software* para implementar algoritmos e executar experimentos para aprendizado online através de uma estrutura de código aberto que permite construir e executar experimentos de aprendizado de máquina ou mineração de dados em fluxos de dados em evolução. As técnicas de classificação geralmente constroem modelos que são usados para prever tendências futuras de dados (BIFET *et al*., 2011).

O MOA consiste em diferentes algoritmos de aprendizado e também em diferentes geradores de fluxo que podem ser encontrados na interface gráfica do usuário ou na versão da linha de comando. O MOA é construído com base na experiência com WEKA e VFML (BIFET *et al*., 2018). Utilizam-se os recursos de Classificação, *Concept Drift* e *Clustering*. Ele permite

ao usuário fazer o processamento ou a execução de uma consulta no fluxo contínuo de dados, ajudando na detecção ou previsão das condições em um pequeno intervalo de tempo após o recebimento dos dados. Essa classificação não oferece opção de armazenamento, isto é, os dados de entrada são processados diretamente assim que chegam.

4.5.1 *Concept Drift*

O *Concept Drift* (TSYMBAL, 2004) refere-se a quando ocorre uma mudança repentina no conjunto de dados que não era previsível. A maioria das pesquisas de desvio de conceito em mineração de fluxos de dados é feita usando estruturas tradicionais de mineração de dados, como o WEKA. A distribuição que gera os itens de um fluxo de dados pode mudar ao longo do tempo. Três abordagens básicas para lidar com o *Concept Drift* podem ser distinguidas: seleção de instâncias, ponderação de instâncias e aprendizado em conjunto. Essas mudanças, dependendo da área de pesquisa, são chamadas de evolução temporal, deslocamento co-variável, não estacionário ou *concept drift* (PATEL; PATEL; BHATT, 2018).

4.5.2 Requisitos para Classificação de Fluxo de Dados

Um algoritmo de classificação deve atender a vários requisitos para trabalhar com as suposições e ser adequado para aprender com fluxos de dados. Os requisitos são detalhados a seguir (BIFET; KIRKBY, 2009).

1. Requisito 1: Processar um exemplo de cada vez e inspecioná-lo apenas uma vez;
2. Requisito 2: Usar uma quantidade limitada de memória;
3. Requisito 3: Trabalhar em um período limitado de tempo;
4. Requisito 4: Estar pronto para prever a qualquer momento.

Quanto ao requisito 1, a principal característica de um fluxo de dados é que os dados "fluem"de uma amostra após a outra. Não há permissão para acesso aleatório dos dados fornecidos. Cada amostra deve ser aceita conforme a ordem em que chega. Uma vez inspecionado ou ignorado, uma amostra é descartada sem capacidade de recuperá-la novamente. Embora este requisito exista na entrada de um algoritmo, não existe uma regra que impeça um algoritmo de lembrar das amostras internamente em um curto prazo. Um exemplo disso pode ser o algoritmo armazenando um lote de amostras para uso em um método de aprendizado convencional. Embora o algoritmo seja livre para operar dessa maneira, ele terá que descartar as amostras armazenadas em algum momento para cumprir o requisito 2.

No requisito 2, a principal motivação para empregar o modelo de fluxo de dados é que ele permite o processamento de dados maiores do que a memória de trabalho disponível. O pe-

rigo de processar grandes quantidades de dados é que a memória se esgota facilmente se não houver um limite intencional definido para seu uso. Essa restrição de memória é uma restrição física que só pode ser relaxada se for usado armazenamento externo, como arquivos temporários, por exemplo. Qualquer solução alternativa precisa ser feita levando em consideração o requisito 3.

O requisito 3 considera que, para que um algoritmo seja dimensionado confortavelmente para qualquer número de exemplos, sua complexidade de tempo de execução deve ser linear ao número de amostras. Isso pode ser alcançado na configuração do fluxo de dados se houver um limite superior constante, de preferência pequeno, na quantidade de processamento. Além disso, para que um algoritmo seja capaz de trabalhar em tempo real, ele deve processar os exemplos tão rápido (ou mais rápido) quanto eles chegam. A falha em fazê-lo significa perda de dados inevitavelmente. Assim, quanto mais lento for o algoritmo, menor será o valor para usuários que exigem resultados dentro de um período de tempo razoável.

Para o requisito 4, um algoritmo ideal deve ser capaz de produzir o melhor modelo possível a partir dos dados observados, depois de processar um número qualquer de amostras. Na prática, é provável que haja períodos em que o modelo permaneça constante, como, por exemplo, quando um algoritmo baseado em lote está armazenando amostras para processar o próximo lote. O processo de geração do modelo deve ser o mais eficiente possível. Ou seja, o modelo final é manipulado diretamente na memória pelo algoritmo à medida que ele processa amostras, em vez de ter que recalcular o modelo com base nas estatísticas de execução.

4.6 Classificadores

Diversos métodos de aprendizagem de máquina foram propostos para monitorar e analisar o tráfego de rede, tanto para ataques com assinaturas, quanto para detecção de anomalias. Segundo (MAHFOUZ; VENUGOPAL; SHIVA, 2020), vários desses métodos de classificação identificam a anomalia observando variações sobre um modelo básico de tráfego normal. Normalmente esses modelos são treinados com um conjunto de dados contendo tráfego livre de ataques, coletados sobre um período longo de tempo.

Os métodos de aprendizagem de máquina que detectam ataques ou anomalias podem ser de um entre três tipos: supervisionado, não-supervisionado ou semi-supervisionado. Aprendizagem supervisionada é o tipo de modelo que usa a variável de entrada (X) e a variável de saída (Y) para fornecer uma base de aprendizagem para apoiar julgamentos futuros, aprendendo a função de mapeamento $Y = f(X)$. A aprendizagem supervisionada usa dados de treinamento que são um conjunto de exemplos com registros de entrada emparelhados e suas saídas desejadas. Neste aprendizado, a resposta correta é conhecida com antecedência, e o algoritmo de aprendizagem faz previsões iterativamente sobre os dados de treinamento e encerra apenas quando um nível aceitável de desempenho é alcançado. Portanto, esse método é apropriado quando existe um valor-alvo específico.

O problema de aprendizagem supervisionada pode ser definido como um problema de classificação ou um problema de regressão. A variável de saída do problema de classificação é uma categoria, como “branco” ou “preto” e “doença” ou “sem_doença”. Por outro lado, a variável de saída do problema de regressão é um valor real, como “quantidade de dinheiro” ou “altura” (FABRIS; MAGALHAES; FREITAS, 2017),

Os métodos de aprendizagem supervisionados mais utilizados são: DT, SVM, Rede Neural Artificial, do inglês *Artificial Neural Network* (ANN), KNN, Regressão Logística, do inglês *Logistic Regression* (LR), Florestas Aleatórias, do inglês *Random Forests* (RF), Naive Bayes, do inglês *Naive Bayes* (NB) (MAHFOUZ; VENUGOPAL; SHIVA, 2020). Neste trabalho, usaremos classificadores com aprendizagem supervisionada, mais especificamente métodos utilizados amplamente na comunidade e que possuam ambas implementações nas ferramentas *Weka* e *MOA*.

4.6.1 Árvores de Decisão

Uma árvore de decisão, ou árvore de classificação, é um sistema de suporte à decisão que utiliza um gráfico na forma de árvore para a tomada de decisões e seus possíveis efeitos posteriores. O algoritmo é usado para aprender uma função de classificação que decide o valor de um atributo dependente (uma variável), considerando os valores dos atributos independentes de entrada, conforme Bhargava (BHARGAVA *et al.*, 2013).

A árvore de decisão J48 é a implementação do algoritmo ID3 (*Iterative Dichotomiser 3*) pelo *WEKA*. Maiores detalhes do algoritmo pode ser encontrado no trabalho de Quinlan (QUINLAN, 2014).

4.6.2 Naive Bayes

Naive Bayes é uma técnica probabilística para construção de classificadores baseado no teorema de Bayes, onde assume-se uma forte independência entre os atributos.

Classificadores *Naive Bayes* são escalonáveis e podem processar um grande número de variáveis lineares (parâmetros) em uma tarefa de aprendizagem. Em uma única iteração dos dados de treinamento, o algoritmo calcula a probabilidade de distribuição condicional de cada atributo de um determinado rótulo, seguido pela aplicação do teorema de Bayes para determinar a distribuição da probabilidade condicional do rótulo, usado para a previsão do resultado, de acordo com Aggarwal (AGGARWAL, 2014).

Maiores informações da implementação do *Naive Bayes* pelo *WEKA* pode ser encontrado no trabalho de Langley (JOHN; LANGLEY, 1995).

4.6.3 Multilayer Perceptron

O perceptron é um algoritmo de aprendizagem clássico para o modelo neural de aprendizagem. O algoritmo é bastante diferente do algoritmo da árvore de decisão ou do algoritmo *KNN*. Primeiramente, ele é *online*. Isso significa que, em vez de considerar todo o conjunto de dados ao mesmo tempo, ele olha apenas para um exemplo. Assim, ele processa aquele exemplo e então segue para o próximo. Em segundo lugar, é orientado por erros. Isso significa que, enquanto estiver indo bem, não se preocupa em atualizar seus parâmetros.

O algoritmo mantém uma "previsão" de bons parâmetros (pesos e tendência) enquanto é executado. Ele processa um exemplo de cada vez. Para um determinado exemplo, ele faz uma previsão. Então verifica se o valor previsto está correto (por se tratar de dados de treinamento, temos acesso aos rótulos verdadeiros). Se estiver correto, não faz nada. Somente quando a previsão está incorreta ele muda seus parâmetros de tal forma que faria melhor neste exemplo da próxima vez. Em seguida, passa para o próximo exemplo. Depois de atingir o último exemplo no conjunto de treinamento, ele reinicia o laço de repetição por um número especificado de iterações.

O algoritmo de treinamento e o algoritmo de previsão para o Perceptron é mostrado na Figura 11.

Figura 11 – Algoritmos para treinamento e previsão do método Perceptron.

Algoritmo PERCEPTRONTRAIN($\mathbf{D}$, $MaxIter$)

$w_d \leftarrow 0$, for all $d = 1 \ldots D$ // inicializa pesos
$b \leftarrow 0$ // inicializa bias
for $iter = 1 \ldots MaxIter$ **do**
  for all $(x,y) \in \mathbf{D}$ **do**
    $a \leftarrow \sum_{d=1}^{D} w_d \, x_d + b$ // calcula ativação
    if $ya \leq 0$ **then**
      $w_d \leftarrow w_d + yx_d$, for all $d = 1 \ldots D$ // atualiza pesos
      $b \leftarrow b + y$ // atualiza bias
    end if
  end for
end for
return $w_0, w_1, \ldots, w_D, b$

Algoritmo PERCEPTRONTEST($w_0, w_1, \ldots, w_D, b, \hat{\mathbf{x}}$)

$a \leftarrow \sum_{d=1}^{D} w_d \, \hat{x}_d + b$ // calcula ativação
return SIGN(a)

Fonte: Adaptado de (III, 2020)..

A forma particular de atualização do Perceptron é simples. O peso w_d é aumentado em yx_d e a tendência (*bias*) é aumentada em y. O objetivo da atualização é ajustar os parâmetros

para que sejam melhores para o exemplo atual. Em outras palavras, se olharmos este exemplo duas vezes seguidas, deveria-se fazer um trabalho melhor na segunda vez (III, 2020).

4.7 Seleção de Características

A FS (também chamada de seleção de atributos) é o processo de seleção de características do conjunto de dados que são mais relevantes para o problema de modelagem preditiva no qual se está trabalhando (KARIMI; KASHANI; HAROUNABADI, 2013). A definição de relevância varia de método para método. Com base em seu entendimento de significância, uma técnica de seleção de características formula matematicamente um critério para avaliar um conjunto de características gerado por um método que pesquisa o espaço de características.

São definidos dois graus de relevância: forte e fraco. Uma característica s é fortemente relevante se sua remoção deteriorar o desempenho de um classificador. Uma característica s é fracamente relevante se não for fortemente relevante porém a remoção de um subconjunto de características contendo s deteriora o desempenho do classificador. Uma característica é irrelevante se não for nem fortemente nem fracamente relevante (KOHAVI; JOHN, 1997). As técnicas de seleção de características auxiliam na criação de um modelo preditivo preciso, escolhendo características que fornecerão melhor precisão e menos complexidade, ao mesmo tempo em que requerem menos dados.

Uma análise é efetuada para ordenar as características em uma ordem de relevância no processo de classificação. Assim, podemos remover as características irrelevantes no processo de classificação, reduzindo o tempo necessário para processar os dados e aumentando a acurácia. Esta análise é feita no módulo de Análise e Classificação às Informações, dentro da unidade AID, para que possam ser analisadas menos características na classificação.

4.7.1 Ganho de Informação

Para realizar a seleção das características, este trabalho utiliza o método de IG, ou seja, uma técnica baseada em filtros que fornece conjuntos mais estáveis de características selecionadas devido à sua natureza robusta contra *overfitting*.

O IG é a técnica de seleção de características mais utilizada, consistindo em uma seleção baseada em filtro (KARIMI; KASHANI; HAROUNABADI, 2013; Alhaj *et al.*, 2016). A técnica utiliza uma classificação de atributo simples e reduz o ruído causado por características irrelevantes e, em seguida, detecta uma característica que tem a maior parte da base de informações em uma classe específica. A melhor característica é determinada calculando a entropia da característica. A entropia é um medida de incerteza que pode ser usada para inferir a distribuição das características de forma concisa (BEREZIÇNSKI; JASIUL; SZPYRKA, 2015).

A entropia pode ser calculada usando a seguinte fórmula:

$$Entropia(S) = \sum_{i}^{c} -P_i log_2 P_i \tag{2}$$

onde c é o número de valores na classe de classificação e P_i é o número de amostras para a classe i. Após receber o valor da entropia, o valor do ganho de informação é calculado através de:

$$Ganho(S, A) = Entropia(S) - \sum_{Valores(A)} \frac{|S_v|}{|S|} Entropia(S_v) \tag{3}$$

onde S é a amostra, A é um atributo, v é um valor possível para o atributo A, $Valores(A)$ é um conjunto de possíveis valores para A. $|S_v|$ é o número de amostras para o valor v. $|S|$ é o número de amostras para todos os dados das amostras e $Entropia(S_v)$ é a entropia para a amostra que possui o valor de v.

Em geral, a complexidade computacional da técnica baseada em filtro é $O(m.n^2)$, onde m é o número de dados de treinamento, e n é o número de características. A complexidade é menor quando comparado às técnicas embarcadas e baseadas em *wrapper* (HASTIE *et al.*, 2004). A natureza complexa de técnicas baseadas em *wrapper* cria um risco elevado de *overfitting*. Assim, a técnica de seleção de características irá produzir um número significativo, relevante e menor de características, além de possuir menor complexidade computacional, reduzindo o tempo de execução de algoritmos de classificação usados no processo de detecção de anomalias e ataques (Kurniabudi *et al.*, 2020).

4.8 NSL-KDD

A base de dados original, KDD'99, possui uma lista com 41 características extraídas a partir de um fluxo de dados, entre eles: duração do fluxo, tipo de protocolo, tipo de serviço, IP de origem, IP de destino, porta, etc. Entretanto, nem todas as características são relevantes para se detectar determinado tipo de ataque, então é necessário realizar uma redução de características, visando obter uma melhor acurácia na detecção de ataques. A lista de prioridades das características com a redução no número de atributos ocorrerá no módulo AID, dentro da UPCD.

Uma análise estatística no conjunto de dados KDD'99 encontrou questões importantes que afetam o desempenho dos sistemas avaliados e resulta em uma avaliação muito pobre das abordagens de detecção de anomalias (HEBA *et al.*, 2010). A deficiência mais importante no conjunto de dados KDD é o grande número de registros redundantes e analisando os conjuntos de treinamento e teste do KDD, descobriu-se que cerca de 78% e 75% dos registros são duplicados no conjunto de treinamento e de teste, respectivamente (TAVALLAEE *et al.*, 2009).

Essa grande quantidade de registros redundantes no conjunto de treinamento fará com que os algoritmos de aprendizagem sejam tendenciosos para os registros mais frequentes e,

assim, evite que eles aprendam registros raros, que geralmente são mais prejudiciais às redes, como os ataques U2R (AHMAD; ABDULLAH; ALGHAMDI, 2010). A existência desses registros repetidos no conjunto de teste, por outro lado, fará com que os resultados da avaliação sejam tendenciosos pelos métodos que apresentam melhores taxas de detecção nos registros frequentes (HASAN *et al.*, 2016).

Para superar o problema do conjunto de dados KDD'99, os pesquisadores propuseram um novo conjunto de dados, NSL-KDD, que consiste em registros selecionados do conjunto completo de dados KDD (TAVALLAEE *et al.*, 2009). O conjunto de dados NSL-KDD não inclui registros redundantes no conjunto de treinamento, portanto, os classificadores não serão tendenciosos para registros mais frequentes. O número de registros nos conjuntos de treinamento e de teste são razoáveis, o que torna possível a execução das classificações no conjunto completo sem a necessidade de selecionar aleatoriamente uma pequena parte. Consequentemente, os resultados da avaliação de diferentes trabalhos de pesquisa serão consistentes e comparáveis. Na Tabela 2 podemos ver os números de amostras para cada tipo de ataques nos *datasets* KDD'99 e NSL-KDD.

Tabela 2 – Resumo comparativo do número de amostras para cada tipo de ataque nos *datasets* KDD'99 e NSL-KDD.

Dataset	Normal	DoS	Probing	R2L	U2R	Total
KDD'99 Completo	972780	3883370	41102	1126	52	4898430
NSL-KDD	60593	229853	4166	16347	70	311029
(NSL) KDDTrain+	67343	45927	11656	995	52	125973
(NSL) KDDTest+	9711	7458	2421	2887	67	22544

Fonte: Autoria própria..

O conjunto de dados NSL-KDD tem as seguintes vantagens sobre o conjunto de dados KDD original (TAVALLAEE *et al.*, 2009):

1. Não inclui registros redundantes no conjunto de treinamento, portanto, os classificadores não serão tendenciosos para registros mais frequentes;

2. Não há registros duplicados no conjunto de teste proposto, portanto, o desempenho dos classificadores não é influenciado pelos métodos que apresentam melhores taxas de detecção nos registros frequentes;

3. O número de registros selecionados de cada grupo é inversamente proporcional à porcentagem de registros no conjunto de dados KDD original. Assim, as taxas de classificação de diferentes métodos de aprendizado de máquina variam em uma faixa mais ampla, resultando em uma avaliação mais precisa das diferentes técnicas de aprendizado;

4. O número de registros no conjunto de treinamento e teste é razoável, o que torna acessível a execução dos experimentos no conjunto completo sem a necessidade de

selecionar aleatoriamente uma pequena parte. Assim, os resultados da avaliação de diferentes trabalhos de pesquisa serão consistentes e comparáveis.

O sistema proposto utiliza uma base de dados simulada para representar o sensor, coletor e pré-processador. A base de dados KDDTest+, aqui referenciada como NSL-KDDTest+, é lida de um arquivo em disco por um programa dedicado a esta tarefa. Então, cada linha de registro, contendo 41 características, é enviada diretamente para a próxima etapa, de publicação. Embora o NSL-KDD seja uma base de dados proveniente do KDD'99, ela é consagrada amplamente pela comunidade, fornecendo informações de ataques comuns há mais de 20 anos.

4.9 CICIDS2017

Com o objetivo de avaliar o desempenho da abordagem proposta para detecção de anomalias, foram realizados experimentos utilizando um conjunto de dados disponibilizado publicamente na Internet. Dos conjuntos de dados existentes, a maioria está desatualizada e não são mais confiáveis. Alguns dos conjuntos sofrem de uma falta de diversidade e volume no tráfego, outros não cobrem a variedade dos ataques conhecidos enquanto outros possuem conjunto de características e *metadata* incompletos (SHARAFALDIN; LASHKARI; GHORBANI, 2018).

O conjunto de dados CICIDS2017 (SHARAFALDIN; LASHKARI; GHORBANI, 2017) contém tráfego legítimo e os ataques atuais mais comuns encontrados na literatura. O conjunto de dados utilizado refere-se ao dia 07 de julho de 2017, onde ataques do tipo *Botnet*, *PortScan* e *DDos* ocorreram em horários específicos pré-determinados. O CICIDS2017 consiste de fluxos de rede rotulados, incluindo o *payload* completo dos pacotes, no formato *pcap*, os perfis correspondentes com os fluxos rotulados, e arquivos *CSV* para fins de aprendizado de máquina e aprendizado profundo. Embora esta base de dados contenha ataques de apenas três tipos, o sistema proposto pode trabalhar com qualquer tipo de ataque desde que possam ser treinados e reconhecidos por métodos de inteligência artificial modular, dentro do módulo de Análise e Classificação das Informações (UPCDs).

4.10 Características Utilizadas

No agrupamento de fluxos de rede, os pacotes de rede são abstraídos em fluxos na camada de rede, onde um fluxo é caracterizado como a sequência de pacotes que vai de um endereço IP de origem para o mesmo endereço IP de destino, durante uma janela fixa de tempo. Para este tipo de agrupamento, as características originais inferidas são variáveis numéricas que estão relacionadas ao conjunto completo de todos os pacotes transmitidos entre dois endereços de IP.

Inicialmente é extraído um total de 84 características do tráfego de rede a partir de um arquivo *pcap*, através do extrator de características baseado em fluxos *CICFlowMeter* (CIC-

FLOWMETER, 2017), (LASHKARI *et al.*, 2017), incluindo IP de origem, porta de origem, IP de destino, porta de destino e protocolo. Todas as características são definidas e explicadas no site da ferramenta *CICFlowMeter* (CICFLOWMETER, 2017).

A seguir, são encontrados os conjuntos contendo as melhores características para detectar cada tipo de ataque, dentre as 84 características extraídas inicialmente. Em seu trabalho, Sharafaldin et al. apresenta os conjuntos contendo as melhores características para cada tipo de ataque (SHARAFALDIN; LASHKARI; GHORBANI, 2018). Os ataques detectados neste artigo são três: *Botnet*, *PortScan* e *DDoS*. As vantagens da redução de dimensionalidade podem ser encontradas no trabalho de Kezih e Taibi (Kezih; Taibi, 2013). A Tabela 3 mostra a lista das melhores características selecionadas para detecção de cada ataque.

Tabela 3 – Melhores características para detecção dos três tipos de ataque (*Botnet*, *PortScan* e *DDos*), segundo (SHARAFALDIN; LASHKARI; GHORBANI, 2018).

Tipo de Ataque	Características
Botnet	Subflow F. Bytes Total Len F. Packets F. Packet Len Mean B. Packets/s
PortScan	Init Win F. Bytes B. Packets/s PSH Flag Count
DDoS	B. Packet Len Std Avg Packet Size Flow Duration Flow IAT Std

Fonte: Autoria própria..

Conforme mostrado na Tabela 3, para detectar um ataque do tipo *Botnet*, as melhores características são encaminhamento de *bytes* de sub-fluxo (*Subflow Forwarding Bytes*), encaminhamento do comprimento e média dos pacotes (*Total Length Forwarding Packets* e *Forwarding Packet Length Mean*) e pacotes de retorno (*Backward Packets per Second*). Para ataques do tipo *PortScan*, utilizam-se os *bytes* da janela de encaminhamento inicial (*Initial Windows Forwarding Bytes*), pacotes de retorno (*Backward Packets per Second*) e *push flags* (*PSH Flag Count*). Por último, os ataques de *DDoS* são melhores detectados utilizando as características de comprimento do pacote de retorno (*Backward Packets Length Standard*), tamanho médio dos pacotes (*Average Packet Size*), e dois intervalos de tempo de chegada (*Flow Duration* e *Flow IAT Standard*). Com base nessas informações publicadas na literatura, optou-se por utilizar as três melhores características para detectar ataques do tipo PortScan dentro da unidade AID, e as quatro melhores características para ataques do tipo Botnet e DDoS.

4.11 Critérios de Desempenho

Várias medidas de desempenho foram propostas na literatura para avaliar um modelo de Aprendizado de máquina, do inglês *Machine Learning* (ML) que pode ser utilizado em um IDS (MAHFOUZ; VENUGOPAL; SHIVA, 2020).

A precisão da classificação é o número total de previsões corretas dividido pelo número total de previsões feitas para um conjunto de dados. Porém, como medida de desempenho, a precisão é inadequada para problemas de classificação desequilibrada. O principal motivo é que o elevado número de exemplos da classe majoritária (ou classes) irá superar o número de exemplos da classe minoritária, o que significa que mesmo modelos inábeis podem atingir pontuações de precisão de 90% ou 99%, dependendo de quão severo o desequilíbrio de classe acontece. Uma alternativa ao uso de precisão de classificação é usar duas métricas: a precisão e o *recall*.

Precisão e *recall* são métricas de avaliação reconhecidas na área de recuperação de informações. A precisão se refere à parte das instâncias relevantes entre as instâncias recuperadas, ou seja, ela quantifica o número de previsões de classe positiva que realmente pertencem à classe positiva. *Recall* refere-se à porção de instâncias recuperadas relevantes do número total das instâncias relevantes, isto é, ela quantifica o número de previsões de classe positivas feitas de todos os exemplos positivos no conjunto de dados. E *F-measure* é a média harmônica de precisão e *recall*, fornecendo uma pontuação única que equilibra as preocupações de precisão e *recall* em um valor.

$$Precisão = \frac{VP}{VP + FP} \tag{4}$$

$$Recall = \frac{VP}{VP + FN} \tag{5}$$

$$F - measure = \frac{2 * Precisão * Recall}{Precisão + Recall} \tag{6}$$

onde VP é o número de verdadeiro-positivos, FP é o número de falso-positivos e FN é o número de falso-negativos.

Para os classificadores em fluxo, utilizando o MOA, avaliamos os experimentos em termos de memória, tempo e desempenho de classificação. A memória é medida em GBs e baseada em horas de RAM (BIFET *et al.*, 1970), ou seja, um GB de memória utilizado por uma hora corresponde a um RAM/Hora. O tempo de processamento é medido em segundos e é baseado no tempo de CPU usado para treinamento e teste (GOMES *et al.*, 2017). Para avaliar o desempenho da classificação, realizamos uma avaliação prévia de validação cruzada de 10 vezes (BIFET *et al.*, 2015). Essa avaliação é diferente da validação cruzada padrão utilizada no aprendizado em lotes. A validação cruzada tradicional não é aplicável à classificação de fluxo

de dados, porque as amostras podem ser fortemente dependentes do tempo, tornando muito difícil organizar as amostras em conjuntos que reflitam as características dos dados.

Três estratégias diferentes foram propostas em (BIFET *et al.*, 2015) para avaliação prévia de validação cruzada: validação cruzada distribuída *k-fold*, validação dividida distribuída *k-fold* e validação de inicialização distribuída *k-fold*. Essas estratégias compartilham a característica de treinar e testar k modelos em paralelo, enquanto eles diferem em como os conjuntos são construídos. Neste trabalho usamos a validação cruzada distribuída *k-fold* conforme recomendado em (BIFET *et al.*, 2015). Nesta estratégia, cada amostra é usada para teste em um modelo selecionado aleatoriamente e para treinamento de todos os outros.

O segundo modelo de avaliação utilizado é o modelo de avaliação de desempenho Kappa de Cohen (*Cohen's Kappa)*. A estatística Kappa de Cohen é uma medida popular para avaliar a precisão da classificação sob desequilíbrio de classe e é usada em cenários de classificação estática, bem como classificação de fluxo de dados. A estatística *kappa* é frequentemente usada para testar a confiabilidade entre observadores. A importância da confiabilidade do avaliador reside no fato de que ela representa o quanto os dados coletados no estudo são representações corretas das variáveis medidas. A medida da extensão em que os coletores de dados (avaliadores) atribuem a mesma pontuação à mesma variável é chamada de confiabilidade entre avaliadores.

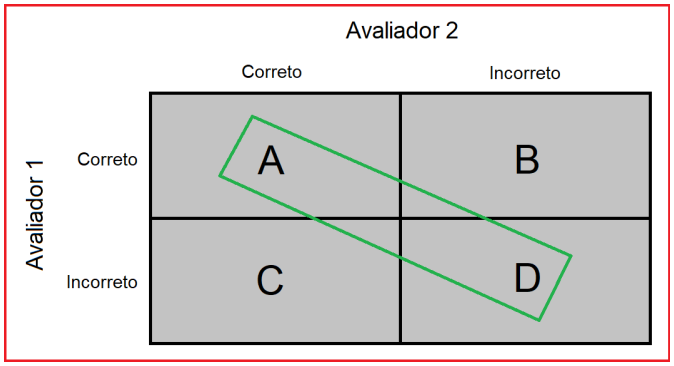
O kappa de Cohen, simbolizado pela letra grega minúscula κ, é uma estatística robusta útil para testes de confiabilidade entre avaliadores ou intra-observadores. Semelhante aos coeficientes de correlação, pode variar de -1 a $+1$, onde 0 representa a quantidade de concordância que pode ser esperada do acaso e 1 representa a concordância perfeita entre os avaliadores. Embora valores de kappa abaixo de 0 sejam possíveis, Cohen observa que eles são improváveis na prática. Como em todas as estatísticas de correlação, o kappa é um valor padronizado e, portanto, é interpretado da mesma forma em vários estudos. Cohen sugeriu que o resultado Kappa fosse interpretado da seguinte forma: valores ≤ 0 como indicando nenhuma concordância e $0{,}01 \sim 0{,}20$ como nenhum a leve, $0{,}21 \sim 0{,}40$ como regular, $0{,}41 \sim 0{,}60$ como moderado, $0{,}61 \sim 0{,}80$ como substancial e $0{,}81 \sim 1{,}00$ como acordo quase perfeito (MCHUGH, 2012).

Considere a Figura 12 a seguir:

onde, A é o número total de amostras que ambos os avaliadores disseram estar corretas, ou seja, os avaliadores estão de acordo. B é o número total de amostras que o Avaliador 2 disse estar incorreto, mas o Avaliador 1 disse que estava correto, neste caso um desacordo. C é o número total de amostras que o Avaliador 1 disse estar incorreto, mas o Avaliador 2 disse que estava correto. Isso também é uma divergência. E, por último, D é o número total de amostras que ambos os avaliadores disseram estar incorretas. Ambos avaliadores estão de acordo.

Para calcular o valor de kappa, primeiro precisamos saber a probabilidade de concordância. Essa fórmula é derivada somando o número de testes em que os avaliadores concordam e dividindo pelo número total de testes. Usando o exemplo da Figura 12, obtemos a probabilidade de acordo, P_o:

Figura 12 – Grade 2x2 usada para interpretar os resultados dos avaliadores.



Fonte: Autoria própria..

$$P_o = \frac{A + D}{A + B + C + D} \tag{7}$$

O próximo passo é calcular a probabilidade de concordância aleatória, P_e. P_e é o número total de vezes que o Avaliador 1 disse estar correto dividido pelo número total de instâncias, multiplicado pelo número total de vezes que o Avaliador 2 disse estar correto dividido pelo número total de instâncias, somado ao número total de vezes que o Avaliador 1 disse estar incorreto multiplicado pelo número total de vezes que o Avaliador 2 disse estar incorreto.

$$P_{correto} = \left(\frac{A + B}{A + B + C + D}\right) * \left(\frac{A + C}{A + B + C + D}\right) \tag{8}$$

$$P_{incorreto} = \left(\frac{C + D}{A + B + C + D}\right) * \left(\frac{B + D}{A + B + C + D}\right) \tag{9}$$

$$P_e = P_{correto} + P_{incorreto} \tag{10}$$

A fórmula para o Kappa de Cohen, κ, é a probabilidade de concordância menos a probabilidade de concordância aleatória dividida por 1 menos a probabilidade de concordância aleatória.

$$\kappa = \frac{P_o - P_e}{1 - P_e} \tag{11}$$

Porém, como a precisão pode ser enganosa em conjuntos de dados com desequilíbrio de classe ou dependências temporais, também utilizamos os modelos de avaliação Kappa temporal e Kappa M. A medida Kappa M tem vantagens sobre a estatística Kappa, pois possui valor zero para um classificador de classe majoritária. Para conjuntos de dados que exibem dependências temporais é aconselhável avaliar o Kappa Temporal pois ele substitui o classificador da classe majoritária pelo classificador No-Change (BIFET *et al.*, 2015).

O Kappa M é uma medida que indica quando estamos nos saindo melhor do que um classificador da classe majoritária. A estatística Kappa M é definida como:

$$\kappa_m = \frac{p_o - p_m}{1 - p_m} \tag{12}$$

onde p_o é a acurácia prévia do classificador e p_m é a acurácia prévia de um classificador de classe majoritária. Se o classificador estiver sempre correto, então $\kappa_m = 1$. Se as previsões estão corretas com a mesma frequência que as de uma classe majoritária classificador, então $\kappa_m = 0$.

Considerando a presença de dependências temporais em fluxos de dados, utiliza-se a medida estatística Kappa temporal (ŽLIOBAITé; BIFET; READ, 2014), definida como:

$$\kappa_{temp} = \frac{p - p_{per}}{1 - p_{per}} \tag{13}$$

onde p_{per} é a acurácia do classificador persistente. O classificador persistente é um classificador que prevê o mesmo rótulo observado anteriormente, ou seja, $\hat{y}_t = y_{t-1}$, para qualquer observação X_t, dado por:

$$p_{per} = P(y_t = y_{t-1}) = \sum_{i=1}^{k} P(y = i)P(y_t = i|y_{t-1} = i) \tag{14}$$

O classificador persistente é baseado no mesmo princípio que é frequentemente usado como linha de base na previsão de séries temporais, isto é, o próximo valor de previsão é igual ao último valor observado.

A estatística kappa temporal pode assumir valores de 1 até $-\infty$. A interpretação é semelhante à de κ. Se o classificador estiver perfeitamente correto então $k_{per} = 1$. Se o classificador estiver alcançando a mesma precisão que o classificador persistente, então $k_{per} = 0$. Os classificadores que superam o classificador persistente ficam entre 0 e 1. Às vezes, pode acontecer que $k_{per} < 0$, o que significa que o classificador de referência está tendo um desempenho pior do que a linha de base do classificador persistente.

5 RESULTADOS E DISCUSSÃO

Este trabalho utiliza duas bases de dados distintas para realizar as classificações: NSL-KDD e CICIDS2017. Assim, serão descritos inicialmente os resultados obtidos utilizando uma base de dados conhecida amplamente na comunidade (NSL-KDD) e depois serão apresentados os resultados obtidos utilizando uma base de dados mais recente (CICIDS2017). Foram realizados testes utilizando dois métodos de classificação, através da mineração de dados convencional e através da mineração de fluxos de dados, para cada base de dados, resultando assim em quatro conjuntos de resultados, como detalhado a seguir:

a) Mineração de dados utilizando a base de dados NSL-KDD;

b) Mineração de dados utilizando a base de dados CICIDS2017;

c) Mineração de fluxo de dados utilizando a base de dados NSL-KDD;

d) Mineração de fluxo de dados utilizando a base de dados CICIDS2017.

Todos os experimentos foram realizados usando um computador com processador Intel Core i7-4790K, 4.00 GHz, com 32 GB de RAM, executando o sistema operacional Linux Ubuntu 20.04 LTS. Foram realizadas quatro etapas de classificação para cada conjunto de resultados.

Na primeira etapa, utilizou-se uma camada de classificação, com um classificador único nessa camada utilizando todas as características. O objetivo de executar o processo de classificação com apenas um classificador foi obter a acurácia que cada método de classificação resulta quando utilizado isoladamente, servindo assim como a acurácia base para comparações posteriores com outras técnicas empregadas neste trabalho.

Na segunda etapa, novamente utilizou-se uma camada de classificação e apenas um classificador. Porém, foi utilizado o processo de redução de características para classificação das amostras. Assim, obteve-se novos resultados sobre a acurácia, falsos-positivos e falsos-negativos, e o tempo de classificação das amostras.

A próxima etapa, a terceira, utilizaram-se duas camadas de classificação e um único classificador. Na primeira camada, o classificador utiliza um conjunto reduzido de características para rápida classificação das amostras. Entretanto, caso a acurácia esteja abaixo de um valor pré-estabelecido, a amostra é enviada para a segunda camada de detecção, no qual o classificador utiliza todas as características da amostra para realizar a classificação com maior acurácia.

E, na quarta e última etapa, utilizaram-se três camadas de classificação, com múltiplos classificadores, utilizando redução de características na primeira camada, sem redução de características na segunda camada, e sem redução de características e sistema de votação na terceira camada. Na terceira camada de classificação, diversos classificadores obtêm um resultado e enviam este para um módulo de decisão, que irá contar os votos recebidos de cada classificador e obter um resultado final.

Abaixo será explicado detalhadamente cada etapa e, então, serão mostrados e analisados todos os resultados obtidos.

5.1 Módulo de Coleta de Dados

Para todas as etapas realizadas em cada conjunto de resultados, é necessário inicialmente obter os dados da rede através de um sensor. Entretanto, para a simulação realizada, optou-se por utilizar a leitura de arquivos contendo as bases de dados ao invés de capturar os pacotes diretamente da rede. Ambas as bases de dados NSL-KDD e CICIDS2017 possuem arquivos de dados no Formato de arquivo de relação de atributo, do inglês *Attribute-Relation File Format* (ARFF). Esse formato de arquivo já contém amostras obtidas através da análise dos pacotes em janelas de intervalos de dois segundos.

A base de dados NSL-KDD possui 41 características para cada amostra, assim como seu rótulo. A base de dados CICIDS2017 possui 84 características e o rótulo da amostra. As amostras são lidas dos arquivos, linha após linha, e enviadas para o próximo módulo da arquitetura.

5.2 Módulo de Pré-Processamento das Informações Coletadas

Este módulo é responsável por receber os dados coletados do módulo anterior, extrair as características relevantes, normalizar as amostras e enviar os resultados para o próximo módulo de transporte. A extração das características é realizada em cima de dados recebidos em intervalos de dois segundos. Esta janela de tempo pode ser maior ou menor, conforme a necessidade de processar as classificações de acordo com o volume do tráfego na rede. Em situações de tráfego elevado, acima da velocidade de classificação do sistema, pode-se optar por janelas de tempo maiores, assim como janelas de tempo menores para situações de tráfego baixo.

Uma vez obtidas as características, realiza-se o trabalho de normalização das mesmas, atribuindo-se valores inteiros para características extraídas em outros formatos (por exemplo, em forma de texto). Obtida a amostra, com suas características e seu rótulo, a amostra é enviada para o próximo módulo, onde ocorre o transporte da mesma.

Neste trabalho optou-se por utilizar características já extraídas dos pacotes de rede (41 características para o NSL-KDD e 84 características para o CICIDS2017). O objetivo do uso dos arquivos ARFF é de utilizar as mesmas características que diversos trabalhos publicados na comunidade usaram, permitindo, assim, a comparação dos resultados obtidos com resultados de outros pesquisadores.

5.3 Módulo de Transporte dos Dados

Neste módulo, a sequência de amostras e suas características extraídas no módulo anterior são transportadas para o módulo seguinte através do sistema publicação-assinatura. O fluxo de amostras pode ser extraído em um ponto qualquer da rede e utilizado para classificação por quaisquer UPCDs, independente do local onde este processo esteja sendo executado.

Utilizando a ferramenta de publicação-assinatura Apache Kafka, cada amostra completa é publicada em uma fila específica. Cada módulo de Pré-Processamento pode, por exemplo, gerar amostras com diferentes números de características, utilizar janelas de tempo diferentes para extração das características, ou extrair características diferentes, permitindo a publicação das amostras em filas específicas. Cada fila tem amostras com características que podem interessar um ou mais assinantes.

Enquanto as amostras são publicadas em um local da rede, outros locais possuem o agente de mensagens Kafka atuando como assinante. Cada UPCDs solicita ao agente de mensagens quais as filas que ele deseja assinar e receber o fluxo de amostras. Uma UPCD pode assinar uma ou mais filas. Quando uma nova amostra é publicada em uma fila, os assinantes são notificados e essa amostra é transportada pela rede para o *buffer* de armazenamento dos mesmos, onde, então, a amostra é lida pelas UPCDs que assinaram aquela fila. Assim, a ferramenta Kafka permite não apenas o transporte das amostras, mas também o armazenamento das mesmas em uma fila de *buffer*, possibilitando a arquitetura classificar todas as amostras geradas mesmo quando o fluxo de amostras gerado se torna temporariamente maior que a capacidade de processamento das UPCDs atualmente empregadas no sistema. Caso este cenário aconteça, a plataforma pode, por exemplo, criar temporariamente novas UPCDs para atender o elevado fluxo de amostras.

Neste trabalho, fez-se a opção de gerar apenas um fluxo de amostras com todas as características, provenientes de um arquivo ARFF, permitindo a comparação dos resultados obtidos com outros resultados apresentados na literatura.

5.4 Módulo de Análise e Classificação dos Dados

Neste módulo, as amostras são classificadas em normais ou ataques através das UPCDs. Cada UPCD é composta de três unidades classificadoras: AID, APD e AFD. Para podermos comparar a eficiência da arquitetura proposta, obtivemos resultados utilizando combinações diferentes de unidades.

Para obter todos os resultados relevantes ao trabalhos, realizamos quatro ciclos de coleta de dados. No primeiro ciclo, utilizamos a base de dados NSL-KDD e classificadores de dados. No segundo ciclo, mantivemos os classificadores mas utilizamos uma base de dados diferente, a CICIDS2017. Uma vez coletados os resultados utilizando duas bases de dados di-

ferentes, repetimos os dois ciclos mas agora utilizando classificadores de fluxo de dados, um ciclo com a base de dados NSL-KDD e outro ciclo com a base de dados CICIDS2017.

5.4.1 Classificadores de Dados

Inicialmente utilizamos métodos comuns de aprendizagem de máquina para classificar amostras sequenciais. Cada classificador é treinado previamente utilizando um conjunto de dados de treino pertencentes as duas bases de dados (NSL-KDD e CICIDS2017). Uma vez treinados os classificadores, as amostras do conjunto de dados de validação são passadas para os mesmos, em um processo de classificação supervisionado. Uma cópia das amostras classificadas é armazenada em uma base de dados temporária que pode, durante o processo de classificação, ser usada para re-treinar os classificadores, se solicitado.

5.4.1.1 NSL-KDD

A base de dados NSL-KDD é dividida em dois arquivos no ARFF: KDDTest+ e KDDTrain+. Para o treinamento dos classificadores, utilizamos os dados supervisionados contidos no arquivo KDDTrain+. Em cada ciclo, temos cada classificador treinado através do método *Cross-validation* com 10 *folds*. O *Cross-validation* possui um viés menor do que outros métodos usados para contar as pontuações de eficiência do modelo e a técnica *k-Fold* oferece um resultado mais estável e confiável, uma vez que o treinamento e o teste são realizados em várias partes diferentes do conjunto de dados, podendo tornar a pontuação geral ainda mais robusta se aumentarmos o número de dobras para testar o modelo em muitos subconjuntos de dados diferentes (YADAV; SHUKLA, 2016).

Cada classificador foi treinado através de 125.973 amostras contidas no arquivo KDDTrain+, sendo fornecido o arquivo KDDTest+, com 22.544 amostras para classificação. Durante o processo de validação, cada classificador foi alimentado por 148.517 amostras contidas no arquivo KDD+.

5.4.1.2 CICIDS2017

A base de dados CICIDS2017 possui registros de cinco dias de simulação, de segunda à sexta-feira. Entretanto, apenas as amostras da sexta-feira serão utilizados neste trabalho, pois é quando ocorrem os ataques dos tipos Botnet, PortScan e DDoS. A base dividida em três arquivos no tipo ARFF: Botnet, PortScan e DDoS. Cada arquivo contém registros de um tipo específico de ataque. Para termos um conjunto de dados de treinamento e um conjunto de dados de teste, removemos aleatoriamente de cada arquivo 20% de registros normais e 20% de registros de ataque para criarmos os arquivos de validação. A criação dos arquivos de treinamento e validação de tamanho reduzido é necessária especialmente para acelerar

o tempo de processamento dos classificadores (CHOUVATUT; JINDALUANG; BOONCHIENG, 2015).

A Tabela 4 mostra as composições dos arquivos antes e depois da criação dos arquivos de treinamento e validação.

Tabela 4 – Arquivos da base de dados CICIDS2017: originais, treinamentos e validações.

Arquivo	Normal	Bot	PortScan	DDos
Botnet Original	189.067	1.966	0	0
Botnet Train	151.254	1.573	0	0
Botnet Test	37.813	393	0	0
PortScan Original	127.537	0	158.930	0
PortScan Train	102.030	0	127.144	0
PortScan Test	25.507	0	31.786	0
DDoS Original	97.718	0	0	128.027
DDoS Train	78.174	0	0	102.422
DDoS Test	19.544	0	0	25.605

Fonte: Autoria própria..

Para o treinamento dos classificadores, utilizamos os dados supervisionados contidos nos arquivos BotnetTrain, PortScanTrain e DDoSTrain. Em cada etapa, temos cada classificador treinado através do método *Cross-validation* com 10 *folds*. Para validação, os classificadores foram alimentados com os dados contidos nos arquivos BotnetTest, PortScanTest e DDoSTest.

5.4.2 Classificadores de Fluxo de Dados

Para realizar o treinamento dos classificadores de fluxo, optou-se pelo método *k-fold distributed split-validation*. Este método é usado quando há abundância de dados e k classificadores. Cada vez que uma nova amostra chega, é decidido com probabilidade $\frac{1}{k}$ se ela será usada para teste. Se for usado para teste, será usado por todos os classificadores. Caso contrário, ela é usada para treinamento e atribuído a apenas um classificador. Fazendo isso, cada classificador recebe amostras diferentes, e elas são testadas usando os mesmos dados (BIFET *et al.*, 2018).

O procedimento de avaliação de um algoritmo de aprendizado determina quais exemplos são usados para treinar o algoritmo e que são usados para testar a saída do modelo pelo algoritmo. Neste trabalho, optamos pelo procedimento de avaliação *Interleaved Test-Then-Train* ou *Prequential*, pois cada exemplo individual pode ser usado para testar o modelo antes de ser usado para treinamento e, a partir disso, a precisão pode ser atualizada de forma incremental. O modelo está sempre sendo testado em exemplos que não foram vistos. Usando este método, o modelo em questão será testado nos exemplos que desconhece (PATEL; PATEL; BHATT, 2018).

5.4.2.1 NSL-KDD

Para a base de dados NSL-KDD, os classificadores de fluxo realizam o treinamento e a classificação através de um único arquivo. Assim, são realizados dois procedimentos antes de realizar a classificação da base de dados NSL-KDD. Inicialmente os arquivos KDDTrain+.arff e KDDTest+.arff são mesclados em um único arquivo KDD+.arff. Em segundo, é realizada uma randomização das amostras para obtermos um fluxo mais uniforme de ataques (BIFET *et al.*, 2018).

5.4.2.2 CICIDS2017

A base de dados CICIDS2017 é dividida em três arquivos no ARFF: Botnet, PortScan e DDoS. Cada arquivo contém registros de um tipo específico de ataque. Para os classificadores de fluxo, utilizamos os arquivos completos, sem necessidade de fragmentação em arquivos de treinamento e de teste.

Semelhante ao procedimento utilizado com a base de dados NSL-KDD, utilizamos o método de treinamento *k-fold distributed split-validation* e o método de avaliação *Interleaved Test-Then-Train*. Para a validação, cada classificador foi alimentado com as amostras contidas nos arquivos Botnet, PortScan e DDoS.

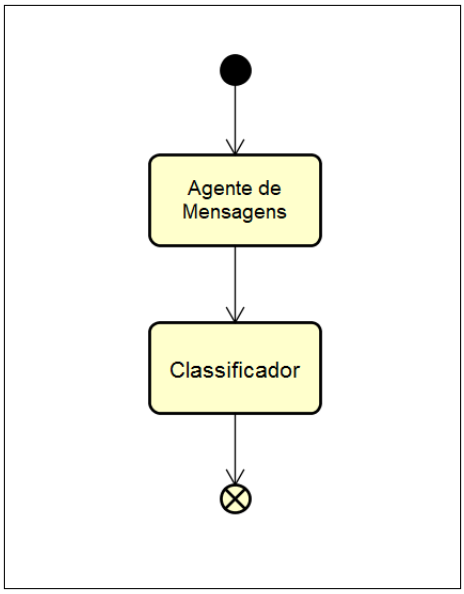
5.4.3 Coleta de Dados com uma Única Unidade Classificadora

Inicialmente, a plataforma utilizou somente a unidade APD. Nesta etapa, realizamos as classificações utilizando três classificadores diferentes, operando cada classificador dentro de uma UPCD própria, onde os dados de treino e teste não passaram por pré-processamento, ou seja, foram utilizados todas as características do conjunto de dados. A Figura 13 demonstra o processamento realizado pelo módulo ao se utilizar apenas um classificador em cada UPCD.

Os três métodos de classificação de dados utilizados foram as Árvores de Decisão, o Naive Bayes e o Multilayer Perceptron, obtendo-se resultados mostrados nas Tabela 5 (classificador de dados e NSL-KDD), Tabela 6 (classificador de dados e CICIDS2017), Tabela 7 (classificador de fluxo de dados e NSL-KDD) e Tabela 8 (classificador de fluxo de dados e CICIDS2017).

Os resultados obtidos servem como base para comparação e análise da eficiência da plataforma proposta.

Figura 13 – As amostras são recebidas do agente de mensagens e encaminhadas diretamente ao classificador.



Fonte: Autoria própria..

Tabela 5 – Resultados obtidos pelo sistema ao utilizar um classificador de dados e a base de dados NSL-KDD.

Classificador	Corretas (%)	Taxa VP	Taxa FP	Precisão	Recall	F-Measure	Tempo (ms)
J48	81,5339	0,815	0,146	0,858	0,815	0,836	198
Naive Bayes	76,1222	0,761	0,197	0,809	0,761	0,784	413
M. Perceptron	75,8738	0,759	0,199	0,808	0,759	0,783	949

Fonte: Autoria própria..

Tabela 6 – Resultados obtidos pelo sistema ao utilizar um classificador de dados e a base de dados CICIDS2017.

Classificador	Corretas (%)	Taxa VP	Taxa FP	Precisão	Recall	F-Measure	Tempo (ms)
J48 Botnet	99,9817	1,000	0,005	1,000	1,000	1,000	492
J48 PortScan	99,9878	1,000	0,016	1,000	1,000	1,000	441
J48 DDoS	99,9911	1,000	0,000	1,000	1,000	1,000	384
Naive Bayes Botnet	96,5635	0,966	0,003	0,992	0,966	0,976	564
Naive Bayes PortScan	99,6876	0,997	0,403	0,997	0,997	0,997	779
Naive Bayes DDoS	99,9003	0,999	0,001	0,999	0,999	0,999	615
M. Perceptron Botnet	99,6493	0,996	0,363	0,997	0,996	0,996	872
M. Perceptron PortScan	99,7347	0,997	0,323	0,997	0,997	0,997	1.255
M. Perceptron DDoS	99,1783	0,992	0,011	0,992	0,992	0,992	1002

Fonte: Autoria própria..

Tabela 7 – Resultados obtidos pelo sistema ao utilizar um classificador de fluxo de dados e a base de dados NSL-KDD.

Classificador	Corretas (%)	Kappa (%)	Kappa Temporal (%)	Kappa M (%)	RAM/Horas	Tempo (s)
J48	97,0094	93,9896	94,0239	93,6975	7,0460	1,64
Naive Bayes	86,8094	73,5441	73,7052	72,2689	9,2902	1,33
Perceptron	99,2957	98,5966	98,6056	98,5294	5,3190	219,89

Fonte: Autoria própria..

Tabela 8 – Resultados obtidos pelo sistema ao utilizar um classificador de fluxo de dados e a base de dados CICIDS2017.

Classificador	Corretas (%)	Kappa (%)	Kappa Temporal (%)	Kappa M (%)	RAM/Horas	Tempo (s)
J48 Botnet	99,6739	83,9788	78,2168	68,3113	2,7437	4,21
J48 PortScan	99,9658	99,9308	98,0423	99,9231	5,0184	5,19
J48 DDoS	99,8419	99,6780	-711,3636	99,6347	6,8211	4,66
Naive Bayes Botnet	98,3329	92,8752	-679,3007	98,6724	3,3399	3,55
Naive Bayes PortScan	98,4225	96,8057	9,7283	96,4567	7,0937	4,95
Naive Bayes DDoS	99,0932	98,1502	-4.552,2727	97,9052	6,3552	4,44
Perceptron Botnet	99,4173	72,1631	61,0839	43,3875	2,7211	6,20
Perceptron PortScan	99,8481	99,6926	91,3104	99,6589	2,9283	7,09
Perceptron DDoS	99,9588	99,9160	-111,3636	99,9048	2,1320	5,09

Fonte: Autoria própria..

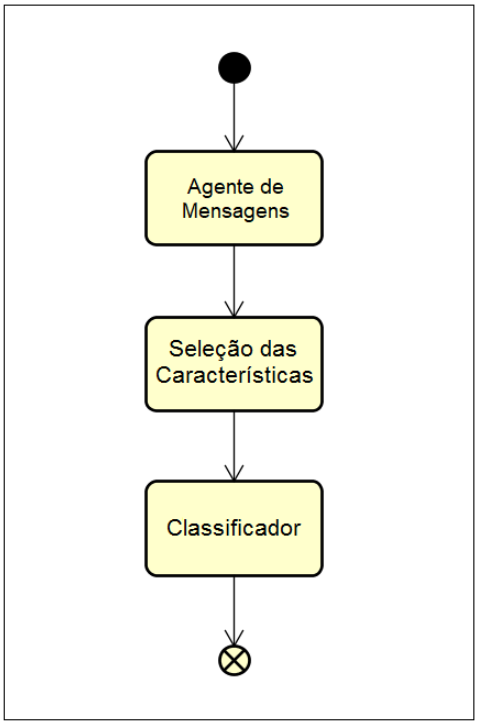
5.4.4 Coleta de Dados com uma Única Unidade Classificadora e Seleção de Características

Nesta segunda etapa de coleta de dados, realizamos um pré-processamento na base de dados de teste antes de passarmos as amostras aos classificadores dentro da unidade AID. Utilizando o avaliador de característica IG com o método de pesquisa *Ranker*, a Tabela 9 apresenta as características ordenadas em ordem de relevância para a base de dados NSL-KDD. Conforme apresentado no trabalho de (CHAE *et al.*, 2013), utilizaremos 22 características para realizar o treinamento e as classificações.

Utilizando a base de dados CICIDS2017, as melhores características foram apresentadas para detecção dos ataques Botnet, PortScan e DDoS, conforme mostradas na Tabela 10. Assim, utilizaremos quatro características para realizar o treinamento e as classificações dos ataques do tipo Botnet e DDoS, e três características para o ataque do tipo PortScan.

Esta etapa de classificação é demonstrada através da Figura 14. As amostras são recebidas do agente de mensagens, escolhidas as características mais relevantes e alimentadas aos classificadores. Novamente, utilizamos três classificadores diferentes neste processo, cada classificador dentro de uma UPCD própria.

Figura 14 – As amostras são recebidas do agente de mensagens e passam por um processo de seleção de características. A seguir, são alimentadas a um classificador já treinado somente com as características selecionadas.



Fonte: Autoria própria..

Tabela 9 – Classificação das características através do avaliador IG com o método *Ranker* sobre o *dataset* NSL-KDD.

Ordem	Relevância	Número da Característica	Descrição da Característica
1	0.81620015	5	src_bytes
2	0.6715649	3	service
3	0.63304893	6	dst_bytes
4	0.51938822	4	flag
5	0.51868903	30	diff_srv_rate
6	0.50984907	29	same_srv_rate
7	0.4759185	33	dst_host_srv_count
8	0.43821385	34	dst_host_same_srv_rate
9	0.41091066	35	dst_host_diff_srv_rate
10	0.40596111	38	dst_host_serror_rate
11	0.40475154	12	logged_in
12	0.39806695	39	dst_host_srv_serror_rate
13	0.39273998	25	serror_rate
14	0.38358863	23	count
15	0.37912493	26	srv_serror_rate
16	0.27083688	37	dst_host_srv_diff_host_rate
17	0.19803814	32	dst_host_count
18	0.18887561	36	dst_host_same_src_port_rate
19	0.14155352	31	srv_diff_host_rate
20	0.09424551	24	srv_count
21	0.08826684	41	dst_host_srv_rerror_rate
22	0.06263911	2	protocol_type
23	0.05674069	27	rerror_rate
24	0.05217739	40	dst_host_rerror_rate
25	0.05156684	28	srv_rerror_rate
26	0.03672469	1	duration
27	0.01155446	10	hot
28	0.00960996	8	wrong_fragment
29	0.00650408	13	num_compromised
30	0.00371697	16	num_root
31	0.00215482	19	num_access_files
32	0.00116837	22	is_guest_login
33	0.00091826	17	num_file_creations
34	0.00051404	15	su_attempted
35	0.00032406	14	root_shell
36	0.00010426	18	num_shells
37	0.00006037	11	num_failed_logins
38	0.00003811	7	land
39	0.00000717	21	is_host_login
40	0	20	num_outbound_cmds
41	0	9	urgent

Fonte: Autoria própria..

Tabela 10 – Melhores características para detecção dos três tipos de ataque (*Botnet*, *PortScan* e *DDos*), segundo (SHARAFALDIN; LASHKARI; GHORBANI, 2018).

Tipo de Ataque	Características
Botnet	Subflow F. Bytes Total Len F. Packets F. Packet Len Mean B. Packets/s
PortScan	Init Win F. Bytes B. Packets/s PSH Flag Count
DDoS	B. Packet Len Std Avg Packet Size Flow Duration Flow IAT Std

Fonte: Autoria própria..

Utilizando as 22 primeiras características (iniciando em *src_bytes* e terminando em *protocol_type*), a Tabela 11 apresenta os resultados obtidos pelo sistema ao utilizar um classificador de dados, com redução no espaço das características e a base de dados NSL-KDD.

Tabela 11 – Resultados obtidos pelo sistema ao utilizar um classificador, com redução das características e a base de dados NSL-KDD.

Classificador	Corretas (%)	Taxa VP	Taxa FP	Precisão	Recall	F-Measure	Tempo (ms)
J48	78,1627	0,782	0,172	0,839	0,782	0,809	88
Naive Bayes	74,3524	0,744	0,209	0,802	0,744	0,772	228
M. Perceptron	76,3174	0,763	0,197	0,808	0,763	0,785	612

Fonte: Autoria própria..

Ao compararmos com os resultados obtidos apresentados na Tabela 5, podemos observar um decréscimo na acurácia para os classificadores J48 e Naive Bayes em aproximadamente 4,13% e 2,32%, respectivamente, enquanto o classificador Perceptron teve um ganho de aproximadamente 0,58%. A redução nas acurácias é explicado através da remoção de algumas características menos relevantes junto com as irrelevantes, tornando a detecção de algumas ameaças menos precisa. Entretanto notamos, para todos os classificadores, uma redução no tempo de processamento dos fluxos de amostras. O classificador J48 teve uma redução no tempo de aproximadamente 55,56%, enquanto os classificadores Naive Bayes e Perceptron tiveram reduções de aproximadamente 44,79% e 35,51%, respectivamente. Estas reduções ocorrem devido ao fato de cada classificador possuir menos características para computar, resultando em um aumento na quantidade de registros que o classificador consegue processar, dentro de um mesmo período de tempo.

Utilizando as características Subflow F. Bytes, Total Len F. Packets, F. Packet Len Mean e B. Packets/s para ataques do tipo Botnet; Init Win F. Bytes, B. Packets/s e PSH Flag Count para ataques do tipo PortScan; e B. Packet Len Std, Avg Packet Size, Flow Duration e Flow IAT Std para ataques do tipo DDoS, a Tabela 12 apresenta os resultados obtidos pelo sistema ao utilizar um classificador, com redução no espaço das características utilizadas e base de dados CICIDS2017.

Comparando os resultados obtidos com os resultados anteriores apresentados na Tabela 6, sem redução de características, podemos notar uma pequena redução na acurácia para quase todos os classificadores de aproximadamente 0,80%, enquanto obteve-se uma redução média no tempo de processamento dos dados de 92,71% $\pm$ 5%, sendo a maior redução no classificador Multilayer Perceptron Botnet (96,67%) e a menor redução no classificador J48 DDoS (88,02%). A Tabela 13 apresenta as variações nos tempos de processamento dos diversos classificadores, sem e com redução das características.

Também podemos observar que o classificador Multilayer Perceptron teve dificuldades nas detecções de ameaças do tipo Botnet e PortScan. Nos ataques do tipo Botnet, embora a acurácia foi de 99,0630%, obtivemos uma taxa de falsos-positivos de 0,966. A Tabela 14

Tabela 12 – Resultados obtidos pelo sistema ao utilizar um classificador, com redução das características e a base de dados CICIDS2017.

Classificador	Corretas (%)	Taxa VP	Taxa FP	Precisão	Recall	F-Measure	Tempo (ms)
J48 Botnet	99,6519	0,997	0,338	0,996	0,997	0,996	28
J48 PortScan	99,8045	0,998	0,184	0,998	0,998	0,998	50
J48 DDoS	99,3754	0,994	0,005	0,994	0,994	0,994	46
Naive Bayes Botnet	95,0376	0,950	0,057	0,991	0,950	0,968	29
Naive Bayes PortScan	99,6457	0,996	0,110	0,997	0,996	0,997	58
Naive Bayes DDoS	99,0498	0,990	0,009	0,991	0,990	0,991	61
M. Perceptron Botnet	99,0630	0,991	0,966	0,990	0,991	0,986	29
M. Perceptron PortScan	77,1229	0,771	0,063	0,993	0,771	0,865	61
M. Perceptron DDoS	96,9435	0,969	0,028	0,970	0,969	0,969	59

Fonte: Autoria própria..

apresenta a Matriz de Confusão para o classificador Multilayer Perceptron e ameaças do tipo Botnet. Quanto aos ataques tipo PortScan, obteve-se uma redução na acurácia de 22,67%. Esses resultados podem ser amenizados através da escolha de outras características diferentes das sugeridas por (SHARAFALDIN; LASHKARI; GHORBANI, 2018). Entretanto, para podermos comparar os resultados obtidos com outros autores, optou-se por utilizar as mesmas características sugeridas por Sharafaldin et al.

Assim como os resultados obtidos ao utilizar a base de dados NSL-KDD, os classificadores apresentaram um decréscimo na acurácia porém com ganho no tempo de processamento. Novamente, estes resultados ocorrem devido ao fato de cada classificador possuir menos características para computar, resultando em um aumento na quantidade de registros que o classificador consegue processar, dentro de um mesmo período de tempo.

Utilizando as 22 primeiras características (iniciando em *src_bytes* e terminando em *protocol_type*), a Tabela 15 apresenta os resultados obtidos pelo sistema ao utilizar um classificador de fluxo de dados, com redução no espaço das características utilizadas e base de dados NSL-KDD.

Ao compararmos com os resultados obtidos apresentados na Tabela 7, podemos observar um decréscimo na acurácia para os três classificadores (J48, Naive Bayes e Perceptron) em aproximadamente 0,06%, 2,97% e 0,13%, respectivamente. A redução nas acurácias é explicado através da remoção de algumas características menos relevantes junto com as irrelevantes, tornando a detecção de algumas ameaças menos precisa. Porém, para todos os

Tabela 13 – Redução no tempo de processamento de cada classificador, utilizando redução de características e a base de dados CICIDS2017, para detecção dos três tipos de ataque (*Botnet*, *PortScan* e *DDos*).

Classificador	Tempo sem Redução de Características (ms)	Tempo com Redução de Características (ms)	Redução no Tempo de Classificação (%)
J48 Botnet	492	28	94,31
J48 PortScan	441	50	88,66
J48 DDoS	384	46	88,02
Naive Bayes Botnet	564	29	94,86
Naive Bayes PortScan	779	58	92,55
Naive Bayes DDoS	615	61	90,08
M. Perceptron Botnet	872	29	96,67
M. Perceptron PortScan	1.255	61	95,14
M. Perceptron DDoS	1.002	59	94,11

Fonte: Autoria própria..

Tabela 14 – Matriz de Confusão para o classificador Multilayer Perceptron e ameaças do tipo Botnet.

BENIGN	Bot	Classificado como
37.840	1	BENIGN
357	9	Bot

Fonte: Autoria própria..

Tabela 15 – Resultados obtidos pelo sistema ao utilizar um classificador de fluxo de dados, com redução das características e a base de dados NSL-KDD.

Classificador	Corretas (%)	Kappa (%)	Kappa Temporal (%)	Kappa M (%)	RAM/Horas	Tempo (s)
J48	96,9485	93,8868	93,9021	93,6582	6,5444	1,52
Naive Bayes	84,2309	71,3211	71,4817	70,5408	3,7893	0,84
Perceptron	99,1684	98,3341	98,3382	98,2718	7,7055	208,56

Fonte: Autoria própria..

classificadores, observa-se uma redução no tempo de processamento dos fluxos de amostras. O classificador J48 teve uma redução no tempo de aproximadamente 7,32%, enquanto os classificadores Naive Bayes e Perceptron tiveram reduções de aproximadamente 36,84% e 5,15%, respectivamente. Estas reduções ocorrem devido ao fato de cada classificador possuir menos

características para computar, resultando em um aumento na quantidade de registros que o classificador consegue processar, dentro de um mesmo período de tempo.

Utilizando as características Subflow F. Bytes, Total Len F. Packets, F. Packet Len Mean e B. Packets/s para ataques do tipo Botnet; Init Win F. Bytes, B. Packets/s e PSH Flag Count para ataques do tipo PortScan; e B. Packet Len Std, Avg Packet Size, Flow Duration e Flow IAT Std para ataques do tipo DDoS, a Tabela 16 apresenta os resultados obtidos pelo sistema ao utilizar um classificador, com redução no espaço das características e a base de dados CICIDS2017.

Tabela 16 – Resultados obtidos pelo sistema ao utilizar um classificador de fluxo, com redução das características e a base de dados CICIDS2017.

Classificador	Corretas (%)	Kappa (%)	Kappa Temporal (%)	Kappa M (%)	RAM/Horas	Tempo (s)
J48 Botnet	99,5189	72,3073	67,8671	53,2553	9,0500	0,57
J48 PortScan	80,5478	-0,7852	-1.199,3007	-1.790,1322	5,7722	0,32
J48 DDoS	99,7588	99,7160	-111,1616	99,7048	1,6635	0,56
Naive Bayes Botnet	97,8551	97,7067	89,7099	97,6746	6,4251	0,45
Naive Bayes PortScan	94,0356	87,9015	-241,3104	86,6031	7,9712	0,45
Naive Bayes DDoS	98,6805	98,3534	80,7219	98,2825	3,3138	1,01
Perceptron Botnet	99,2119	98,3970	-3.943,1818	98,1794	1,0473	0,48
Perceptron PortScan	74,3604	49,9299	-131.445,4545	40,7677	7,6834	0,42
Perceptron DDoS	87,9599	76,0650	-61.672,7272	72,1850	4,4868	0,48

Fonte: Autoria própria..

Comparando os resultados obtidos com os resultados anteriores apresentados na Tabela 8, sem redução de características, podemos notar novamente uma redução na acurácia para quase todos os classificadores menor que 1%. Para os ataques do tipo Portscan, entretanto, a redução na acurácia foi, no pior caso, de 25,53%, para o classificador Perceptron. Porém, obteve-se uma redução média no tempo de processamento dos dados acima de 75%, sendo a maior redução no classificador Perceptron Portscan (94,08%) e a menor redução no classificador Naive Bayes DDoS (77,25%). A Tabela 17 apresenta as variações nos tempos de processamento dos diversos classificadores, sem e com redução das características.

Podemos observar que os classificadores J48 e Perceptron tiveram dificuldades nas detecções de ameaças do tipo PortScan. O classificador J48 apresentou um Kappa de $-0,7852\%$, o que representa uma confiança no classificador abaixo do classificador persistente. Entretanto, o classificador Perceptron, embora tenha resultado em uma acurácia menor, apresentou um

Tabela 17 – Redução no tempo de processamento de cada classificador de fluxo de dados, utilizando redução de características e a base de dados CICIDS2017, para detecção dos três tipos de ataque (*Botnet*, *PortScan* e *DDos*).

Classificador	Tempo sem Redução de Características (ms)	Tempo com Redução de Características (ms)	Redução no Tempo de Classificação (%)
J48 Botnet	4,21	0,57	86,46
J48 PortScan	5,19	0,32	93,83
J48 DDoS	4,66	0,56	87,98
Naive Bayes Botnet	3,55	0,45	87,32
Naive Bayes PortScan	4,95	0,45	90,91
Naive Bayes DDoS	4,44	1,01	77,25
M. Perceptron Botnet	6,20	0,48	92,26
M. Perceptron PortScan	7,09	0,42	94,08
M. Perceptron DDoS	5,09	0,56	89,00

Fonte: Autoria própria..

Kappa de $49{,}9299\%$, ou seja, uma confiabilidade acima do classificador persistente. Semelhante aos resultados obtidos utilizando classificadores tradicionais, esses resultados podem ser amenizados através da escolha de outras características diferentes das sugeridas por (SHARAFALDIN; LASHKARI; GHORBANI, 2018). Entretanto, para podermos comparar os resultados obtidos com outros autores, optou-se por utilizar as mesmas características sugeridas por Sharafaldin et al.

Assim como os resultados obtidos ao utilizar a base de dados NSL-KDD, os classificadores apresentaram um decréscimo na acurácia porém com ganho no tempo de processamento e redução no consumo de memória (RAM/Hora). Novamente, estes resultados ocorrem devido ao fato de cada classificador possuir menos características para computar, resultando em um aumento na quantidade de registros que o classificador consegue processar, dentro de um mesmo período de tempo.

5.4.5 Coleta de Dados com Duas Unidades Classificadoras e Seleção de Características

O próximo passo é introduzir no sistema duas unidades de classificação dos dados: a AID e a APD. A AID é responsável pela análise inicial das amostras. Este processo utiliza as características mais relevantes para realizar a classificação da amostra, em tempo reduzido de processamento.

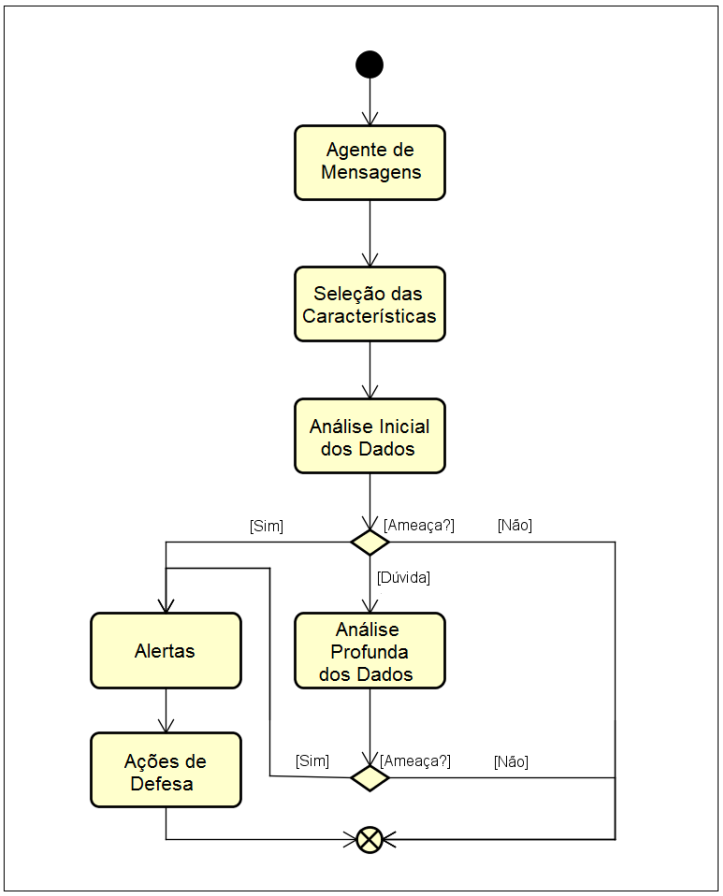
Para os classificadores de dados tradicionais, cada classificador possui, inicialmente uma taxa de acertos definida pelo seu treinamento inicial. Cada amostra recebida é classificada e feito um registro da acurácia (número de acertos em relação ao número total de amostras já classificadas). A cada amostra classificada, uma nova taxa de acerto é calculada. Esse cálculo pode ser feito facilmente quando se utilizam amostras rotuladas. Caso contrário, é necessário comparar o resultado de pelo menos três UPCDs independentes para se obter o resultado da classificação através de uma votação simples. Se esta nova taxa de acerto ficar menor que um valor pré-determinado, as novas amostras passarão a ser classificadas na APD. A APD utiliza todas as características do *dataset*, aumentando a acurácia na detecção das ameaças, porém consumindo mais tempo para realizar a tarefa. Novamente, a taxa de acerto é ajustada após cada amostra ser classificada e, caso a taxa de acerto seja maior que o valor pré-determinado, a classificação volta a ocorrer novamente na AID. Novamente, ao se utilizar amostras não rotuladas, é necessário no mínimo três UPCDs independentes para decidirem o resultado da classificação da amostra. Durante este período de classificação pela APD, o classificador na AID pode ser re-treinado utilizando-se a base de dados formada pelas amostras já classificadas.

Para os classificadores de fluxo de dados, o classificador possui inicialmente uma taxa de acertos definida pelo seu treinamento inicial. Após cada amostra ser classificada, esta é utilizada para continuamente treinar o sistema, gerando uma nova taxa de acerto. Para amostras não rotuladas, utiliza-se o mesmo sistema de votação através de pelo menos três UPCDs independentes para estimar o valor da amostra. Se esta nova taxa de acerto ficar menor que um valor pré-determinado, as novas amostras passarão a ser classificadas na APD. A APD utiliza todas as características do *dataset*, aumentando a acurácia na detecção das ameaças, porém consumindo mais tempo para realizar a tarefa. Novamente, a taxa de acerto é ajustada após cada registro ser classificado e, caso a taxa de acerto seja maior que o valor pré-determinado, a classificação passa a ocorrer novamente na AID. De maneira diferente dos classificadores tradicionais, a re-treinagem é feita dinamicamente a cada amostra avaliada, ao contrário do re-treino em lote realizado em intervalos determinados.

A Figura 15 representa o sistema de classificação com duas camadas de classificadores: AID e APD.

A Tabela 18 apresenta os resultados obtidos pelo sistema ao utilizar um classificador de dados em duas camadas (utilizando redução no espaço das características e com o conjunto completo de características) e a base de dados NSL-KDD. O valor pré-estabelecido utilizado

Figura 15 – As amostras são recebidas através do Agente de Mensagens e depois possuem seu número de características reduzidas para serem classificadas na AID. Caso necessário, serão encaminhadas para a APD, onde serão classificadas utilizando todas as características.



Fonte: Autoria própria..

para decisão nas mudanças de camada de classificação foi de 80%. Outros valores foram testados, com resultados mistos de acurácia e tempo de execução.

Tabela 18 – Resultados obtidos pelo sistema ao utilizar um classificador de dados em duas camadas, com e sem redução de características, e com a base de dados NSL-KDD.

Classificador	Corretas (%)	Taxa VP	Taxa FP	Precisão	Recall	F-Measure	Tempo (ms)
J48	80,7448	0,809	0,155	0,851	0,809	0,829	114
Naive Bayes	75,8778	0,755	0,201	0,806	0,755	0,780	257
M. Perceptron	76,3174	0,763	0,197	0,808	0,763	0,785	612

Fonte: Autoria própria..

Comparando os resultados obtidos com os anteriores, os classificadores J48 e Naive Bayes apresentaram resultados intermediários. Enquanto as acurácias ficaram mais próximas dos valores obtidos ao utilizar todas as 41 características, o tempo total de execução ficou mais próximo dos resultados obtidos ao se utilizar apenas 22 características. Este comportamento

se deve ao fato da alternância entre as camadas de decisão. Parte da classificação é realizada na camada rápida porém com menor acurácia e a outra parte da classificação é realizada na camada de maior acerto, porém que exige maior tempo de processamento. O valor de corte permite estabelecer quanto tempo o classificador permanece em cada camada: valores menores de corte fazem o sistema permanecer na camada AID por mais tempo, enquanto valores maiores fazem o sistema passar mais tempo na camada APD. Assim, o sistema demonstra que é possível realizar uma relação diretamente proporcional entre acurácia e tempo de processamento ajustando o valor pré-determinado de corte entre as duas camadas AID e APD. Pode-se, com isso, escolher uma maior acurácia porém utilizando maior tempo de processamento, ou escolher reduzir o tempo necessário para as classificações ao custo de uma redução na taxa de acerto. Para o classificador Multilayer Perceptron, obteve-se o mesmo resultado comparado ao uso de 22 características, utilizando um valor de corte de 80% na taxa de acerto.

A Tabela 19 apresenta os resultados obtidos pelo sistema ao utilizar um classificador de dados, em duas camadas (utilizando redução no espaço das características e com o conjunto completo de características) e a base de dados CICIDS2017. O valor pré-estabelecido utilizado para decisão nas mudanças de camada de classificação foi de 80%.

Tabela 19 – Resultados obtidos pelo sistema ao utilizar um classificador de dados em duas camadas, com e sem redução das características, e a base de dados CICIDS2017.

Classificador	Corretas (%)	Taxa VP	Taxa FP	Precisão	Recall	F-Measure	Tempo (ms)
J48 Botnet	99,8536	0,999	0,120	0,998	0,998	0,998	64
J48 PortScan	99,9559	0,999	0,052	0,999	0,999	0,999	97
J48 DDoS	99,7812	0,998	0,002	0,998	0,998	0,998	88
Naive Bayes Botnet	95,9771	0,961	0,012	0,984	0,961	0,970	55
Naive Bayes PortScan	99,6603	0,997	0,172	0,997	0,996	0,997	121
Naive Bayes DDoS	99,6691	0,998	0,003	0,998	0,998	0,998	149
M. Perceptron Botnet	99,4811	0,994	0,418	0,995	0,994	0,994	48
M. Perceptron PortScan	96,0018	0,968	0,387	0,968	0,968	0,968	131
M. Perceptron DDoS	98,0212	0,989	0,016	0,989	0,989	0,989	114

Fonte: Autoria própria..

Comparando os resultados obtidos com os anteriores, todos os classificadores apresentaram resultados intermediários. Enquanto as acurácias ficaram mais próximas dos valores obtidos ao utilizar todas as 78 características, o tempo total de execução ficou mais próximo dos resultados obtidos ao se utilizar apenas 3 ou 4 características. Este comportamento se deve ao

fato da alternância entre as camadas de decisão. Parte da classificação é realizada na camada rápida porém com menor acurácia e a outra parte da classificação é realizada na camada de maior acerto, porém que exige maior tempo de processamento. Ainda, o valor de corte permite estabelecer quanto tempo o classificador permanece em cada camada: valores menores de corte fazem o sistema permanecer na camada AID por mais tempo, enquanto valores maiores fazem o sistema passar mais tempo na camada APD. Assim, igualmente demonstrado com a base de dados NSL-KDD, o sistema demonstra que é possível realizar uma relação diretamente proporcional entre acurácia e tempo de processamento ajustando o valor pré-determinado de corte entre as duas camadas AID e APD. Pode-se, com isso, escolher uma maior acurácia porém utilizando maior tempo de processamento, ou escolher reduzir o tempo necessário para as classificações ao custo de uma redução na taxa de acerto.

A Tabela 20 apresenta os resultados obtidos pelo sistema ao utilizar um classificador de fluxo de dados, com duas camadas (utilizando redução no espaço das características e com o conjunto completo de características) e a base de dados NSL-KDD. O valor pré-estabelecido utilizado para decisão nas mudanças de camada de classificação foi de 80%. Outros valores foram testados, com resultados mistos de acurácia e tempo de execução.

Tabela 20 – Resultados obtidos pelo sistema ao utilizar um classificador de fluxo de dados em duas camadas, com e sem redução de características, e com a base de dados NSL-KDD.

Classificador	**Corretas (%)**	**Kappa (%)**	**Kappa Temporal** (%)	**Kappa M (%)**	**RAM/Horas**	**Tempo (s)**
J48	96,9989	93,9278	93,9774	93,6701	6,8864	1,55
Naive Bayes	86,4419	73,2009	73,2297	71,9088	4,2559	0,91
Perceptron	99,2514	98,5478	98,5771	98,4930	5,9634	212,97

Fonte: Autoria própria..

Comparando os resultados obtidos com os anteriores, todos os classificadores apresentaram resultados intermediários. Enquanto as acurácias ficaram mais próximas dos valores obtidos ao utilizar todas as 41 características, o tempo total de execução ficou mais próximo dos resultados obtidos ao se utilizar apenas 22 características. Este comportamento se deve ao fato da alternância entre as camadas de decisão. Parte da classificação é realizada na camada rápida porém com menor acurácia e a outra parte da classificação é realizada na camada de maior acerto, porém que exige maior tempo de processamento. O valor de corte permite estabelecer quanto tempo o classificador permanece em cada camada: valores menores de corte fazem o sistema permanecer na camada AID por mais tempo, enquanto valores maiores fazem o sistema passar mais tempo na camada APD. O sistema demonstra que é possível realizar uma relação diretamente proporcional entre acurácia e tempo de processamento ajustando o valor pré-determinado de corte entre as duas camadas AID e APD. Pode-se, com isso, escolher uma maior acurácia porém utilizando maior tempo de processamento, ou escolher reduzir o tempo necessário para as classificações ao custo de uma redução na taxa de acerto.

A Tabela 21 apresenta os resultados obtidos pelo sistema ao utilizar um classificador de fluxo de dados, com duas camadas (utilizando redução no espaço das características e com o conjunto completo de características) e a base de dados CICIDS2017. O valor pré-estabelecido utilizado para decisão nas mudanças de camada de classificação foi de 80%.

Tabela 21 – Resultados obtidos pelo sistema ao utilizar um classificador de fluxo em duas camadas, com e sem redução das características, e utilizando a base de dados CICIDS2017.

Classificador	Corretas (%)	Kappa (%)	Kappa Temporal (%)	Kappa M (%)	RAM/Horas	Tempo (s)
J48 Botnet	99,6421	81,0219	76,9330	65,4739	4,8806	1,22
J48 PortScan	95,6558	96,0119	92,9001	92,9937	5,2903	0,78
J48 DDoS	99,8377	99,6955	-248,4937	99,6855	2,1873	1,09
Naive Bayes Botnet	98,0228	95,9980	64,0019	98,0993	4,9637	0,89
Naive Bayes PortScan	97,1190	92,3348	1,6596	89,9338	7,2337	0,99
Naive Bayes DDoS	98,9099	98,3008	48,6279	98,1089	4,8873	2,38
Perceptron Botnet	99,2884	92,9552	39,5591	91,0029	1,8490	1,93
Perceptron PortScan	95,5663	92,0338	78,6449	92,9938	4,1967	2,98
Perceptron DDoS	95,2937	93,8633	-87.224,1389	93,9008	3,0098	1,22

Fonte: Autoria própria..

Os classificadores apresentaram resultados intermediários. As acurácias, o tempo total de execução e o consumo de memória ficaram próximos aos melhores valores obtidos, tanto ao utilizar todas as 78 características ou com redução para 3 e 4 características. Novamente, este comportamento se deve ao fato da alternância entre as camadas de decisão. A alternância nas camadas de processamento permitiu ao sistema aumentar as acurácias dos classificadores J48 PortScan, Perceptron PortScan e Perceptron DDoS, enquanto reduzindo o tempo de processamento. Novamente, parte da classificação é realizada na camada rápida porém com menor acurácia, consumo de tempo e memória, e a outra parte da classificação é realizada na camada de maior acerto, porém que exige maior tempo de processamento e consumo de memória. O valor de corte permite estabelecer quanto tempo o classificador permanece em cada camada: valores menores de corte fazem o sistema permanecer na camada AID por mais tempo, enquanto valores maiores fazem o sistema passar mais tempo na camada APD. Assim, igualmente demonstrado com a base de dados NSL-KDD, o sistema demonstra que é possível realizar uma relação entre acurácia, tempo de processamento e consumo de memória ajustando o valor pré-determinado de corte entre as duas camadas AID e APD. Assim, pode-se escolher

uma maior acurácia porém utilizando maior tempo de processamento, ou escolher reduzir o tempo necessário para as classificações, ao custo de uma redução na taxa de acerto.

5.4.5.1 Coleta de Dados com Três Classif., Seleção de Caract. e Sistema de Votação

A partir dos resultados anteriores, observa-se que é possível aumentar a taxa de acerto nas classificações ao custo de tempo de processamento. Nota-se também que cada classificador possui uma taxa de acerto diferente, pois cada método de aprendizagem de máquina utiliza técnicas diferentes para detecção de ameaças.

Assim, construímos na arquitetura o componente AFD. Os três módulos UPCDs contendo diferentes classificadores estarão operando independentemente e em paralelo. O sistema inicialmente fará a classificação no AID. Caso exista incerteza na classificação da amostra, esta será encaminhada ao APD. Novamente, existindo baixa acurácia na classificação, então a amostra será encaminhada ao terceiro nível de classificação, a AFD. A amostra será enviada para diversos classificadores independentes, treinados em métodos diferentes de detecção de ameaças. Neste trabalho, optamos por utilizar o mesmo *dataset* para treinar cada classificador da AFD, trabalhando com todas as características do conjunto de dados. Após o classificador realizar a classificação da amostra, cada um deles transmite ao Decisor o resultado obtido e a sua taxa de acerto atual. Baseado nesses dados, o Decisor realiza a tarefa de classificação final através do processo de votação, obtendo assim uma classificação definitiva da amostra.

Podemos observar o módulo de classificação AFD, com três classificadores e um Decisor final, na Figura 16. Nesta etapa, utilizou-se a arquitetura completa da plataforma proposta.

A Tabela 22 apresenta os resultados obtidos pelo sistema ao utilizar três classificadores de dados, com os módulos AID, APD e AFD, um Decisor final (pertencente ao AFD) e a base de dados NSL-KDD. O valor pré-estabelecido utilizado para decisão nas mudanças de camada de classificação foi de 80%.

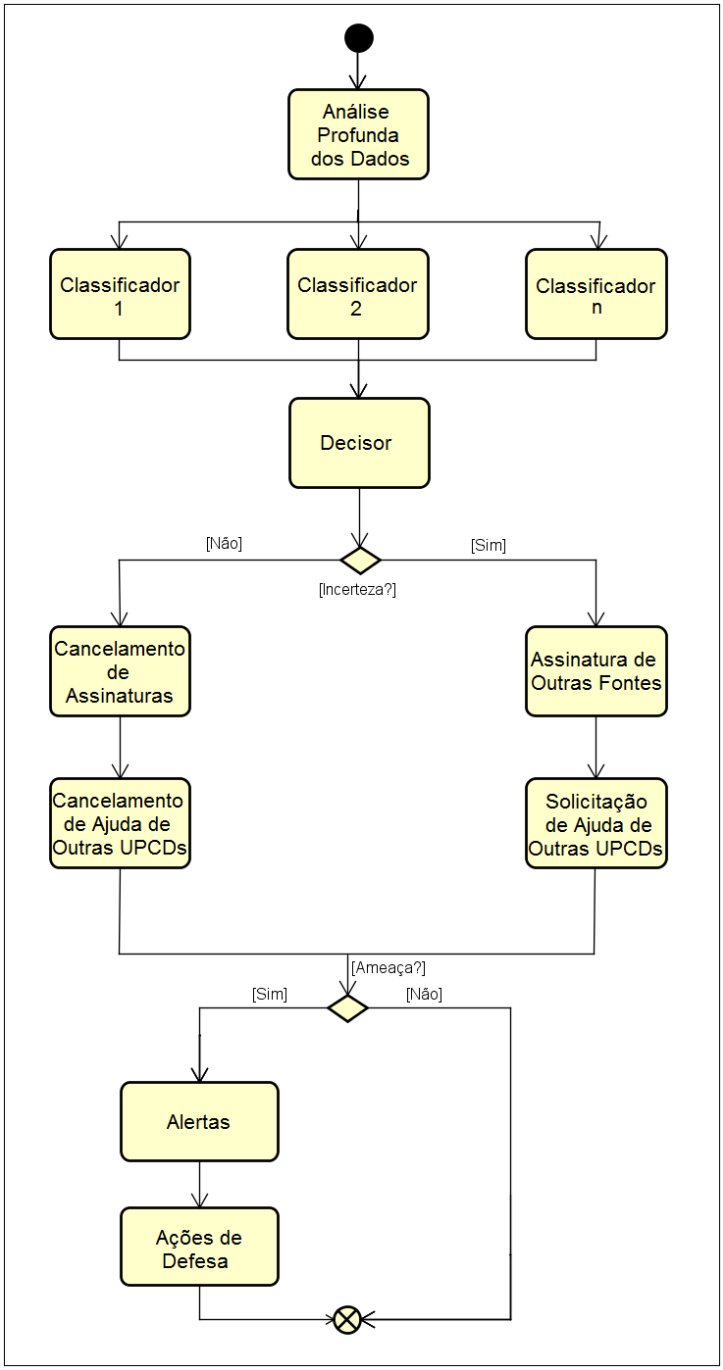
Tabela 22 – Resultados obtidos pelo sistema ao utilizar três classificadores de dados, com os módulos AID, APD e AFD e um Decisor, com a base de dados NSL-KDD.

Classificador	Corretas (%)	Taxa VP	Taxa FP	Precisão	Recall	F-Measure	Tempo (ms)
J48	80,7448	0,809	0,155	0,851	0,809	0,829	114
Naive Bayes	75,8778	0,755	0,201	0,806	0,755	0,780	257
M. Perceptron	76,3174	0,763	0,197	0,808	0,763	0,785	612
Decisor	89,9265	0,887	0,068	0,929	0,892	0,910	612

Fonte: Autoria própria..

Os resultados acima mostram que a taxa de acerto do decisor apresenta melhoras de 11,37%, 18,51% e 17,83% em comparação aos classificadores J48, Naive Bayes e Multilayer Perceptron, respectivamente. Isso ocorre devido ao sistema de votação, permitindo o decisor descartar uma classificação incorreta que tenho acontecido por um classificador isolado. En-

Figura 16 – As amostras recebidas da APD são inicialmente processados em classificadores separadamente e depois é realizada uma votação no Decisor para obter a classificação final da amostra.



Fonte: Autoria própria..

tretanto, devido ao Decisor necessitar das três classificações para obter a classificação final, o tempo de processamento é igual ao maior tempo de processamento entre os três classificadores, neste caso, do Multilayer Perceptron.

As Tabela 23, Tabela 24 e Tabela 25 apresentam os resultados obtidos pelo sistema ao utilizar três classificadores de dados, com os módulos AID, APD e AFD, um Decisor final (pertencente ao AFD), a base de dados CICIDS2017 e os tipos de ataques Botnet, PortScan e DDoS, respectivamente. O valor pré-estabelecido utilizado para decisão nas mudanças de camada de classificação foi de 80%.

Tabela 23 – Resultados obtidos pelo sistema ao utilizar três classificadores de dados, com os módulos AID, APD e AFD e um Decisor, com a base de dados CICIDS2017 e ataques do tipo Botnet.

Classificador	Corretas (%)	Taxa VP	Taxa FP	Precisão	Recall	F-Measure	Tempo (ms)
J48 Botnet	99,8536	0,999	0,120	0,998	0,998	0,998	64
Naive Bayes Botnet	95,9771	0,961	0,012	0,984	0,961	0,970	55
M. Perceptron Botnet	99,4811	0,994	0,418	0,995	0,994	0,994	48
Decisor Botnet	99,9912	1,000	0,002	1,000	1,000	1,000	64

Fonte: Autoria própria..

Tabela 24 – Resultados obtidos pelo sistema ao utilizar três classificadores de dados, com os módulos AID, APD e AFD e um Decisor, com a base de dados CICIDS2017 e ataques do tipo PortScan.

Classificador	Corretas (%)	Taxa VP	Taxa FP	Precisão	Recall	F-Measure	Tempo (ms)
J48 PortScan	99,9559	0,999	0,052	0,999	0,999	0,999	97
Naive Bayes PortScan	99,6603	0,997	0,172	0,997	0,996	0,997	121
M. Perceptron PortScan	96,0018	0,968	0,387	0,968	0,968	0,968	131
Decisor PortScan	99,9973	1,000	0,008	0,999	1,000	1,000	131

Fonte: Autoria própria..

Os resultados acima mostram que a taxa de acerto do decisor apresenta melhoras de 0,14%, 4,18% e 0,51% em comparação aos classificadores J48, Naive Bayes e Multilayer Perceptron, respectivamente, para ataques do tipo Botnet. Para ataques do tipo PortScan, as melhorias foram de 0,04%, 3,38% e 4,16% em comparação aos classificadores J48, Naive Bayes e Multilayer Perceptron, respectivamente. Por último, para ataques do tipo DDOS, obteve-se melhorias na acurácia de 0,21%, 0,32% e 2,00% em comparação aos classificadores J48, Naive

Tabela 25 – Resultados obtidos pelo sistema ao utilizar três classificadores de dados, com os módulos AID, APD e AFD e um Decisor, com a base de dados CICIDS2017 e ataques do tipo DDoS.

Classificador	Corretas (%)	Taxa VP	Taxa FP	Precisão	Recall	F-Measure	Tempo (ms)
J48 DDoS	99,7812	0,998	0,002	0,998	0,998	0,998	88
Naive Bayes DDoS	99,6691	0,998	0,003	0,998	0,998	0,998	149
M. Perceptron DDoS	98,0212	0,989	0,016	0,989	0,989	0,989	114
Decisor DDoS	99,9864	0,999	0,001	0,999	0,999	0,999	149

Fonte: Autoria própria..

Bayes e Multilayer Perceptron, respectivamente. Embora o sistema de votação apresente um aumento na taxa de acerto em todos os casos, observa-se que esse aumento não é tão acentuado ao se utilizar a base de dados CICIDS2017. Isso se deve ao fato da base de dados CICIDS2017 possuir ataques de um único tipo (Botnet, PortScan ou DDoS), resultando em um treinamento mais preciso para cada classificador quando comparado a base de dados NSL-KDD, que requer cada classificador detectar 39 tipos de ataques. Ainda, igualmente aos resultados obtidos ao usar a base de dados NSL-KDD, devido ao decisor necessitar das três classificações para obter a classificação final, o tempo de processamento é igual ao maior tempo de processamento entre os três classificadores, visto que cada classificador opera em paralelo com os outros classificadores, de forma distribuída.

A Tabela 26 apresenta os resultados obtidos pelo sistema ao utilizar três classificadores de fluxo de dados, com os módulos AID, APD e AFD, um Decisor final (pertencente ao AFD) e a base de dados NSL-KDD. O valor pré-estabelecido utilizado para decisão nas mudanças de camada de classificação foi de 80%.

Tabela 26 – Resultados obtidos pelo sistema ao utilizar três classificadores de fluxo de dados, com os módulos AID, APD e AFD e um Decisor, com a base de dados NSL-KDD.

Classificador	Corretas (%)	Kappa (%)	Kappa Temporal (%)	Kappa M (%)	RAM/Horas	Tempo (s)
J48	96,9989	93,9278	93,9774	93,6701	6,8864	1,55
Naive Bayes	86,4419	73,2009	73,2297	71,9088	4,2559	0,91
Perceptron	99,2514	98,5478	98,5771	98,4930	5,9634	212,97
Decisor	99,8852	99,8361	99,8749	99,8408	17,1057	212,97

Fonte: Autoria própria..

Os resultados acima mostram que a taxa de acerto do decisor apresenta melhoras de 2,98%, 15,55% e 0,63% em comparação aos classificadores J48, Naive Bayes e Perceptron, respectivamente. Isso ocorre devido ao sistema de votação, permitindo o decisor descartar uma

classificação incorreta que tenho acontecido por um classificador isolado. Entretanto, devido ao decisor necessitar das três classificadores em paralelo para obter a classificação final, o tempo de processamento é igual ao maior tempo de processamento entre os três classificadores, neste caso, do Perceptron. Ainda, o consumo de memória (RAM/Hora) apresentou um acréscimo significativo, devido a necessidade de construir três modelos de classificação ao mesmo tempo, em memória.

As Tabela 27, Tabela 28 e Tabela 29 apresentam os resultados obtidos pelo sistema ao utilizar três classificadores de fluxo de dados, com os módulos AID, APD e AFD, um Decisor final (pertencente ao AFD), a base de dados CICIDS2017 e os tipos de ataques Botnet, PortScan e DDoS, respectivamente. O valor pré-estabelecido utilizado para decisão nas mudanças de camada de classificação foi de 80%.

Tabela 27 – Resultados obtidos pelo sistema ao utilizar três classificadores de fluxo de dados, com os módulos AID, APD e AFD e um Decisor, com a base de dados CICIDS2017 e ataques do tipo Botnet.

Classificador	**Corretas (%)**	**Kappa (%)**	**Kappa Temporal** (%)	**Kappa M (%)**	**RAM/Horas**	**Tempo (s)**
J48 Botnet	99,6421	81,0219	76,9330	65,4739	4,8806	1,22
Naive Bayes Botnet	98,0228	95,9980	64,0019	98,0993	4,9637	0,89
Perceptron Botnet	99,2884	92,9552	39,5591	91,0029	1,8490	1,93
Decisor Botnet	99,8817	98,5568	88,9771	98,6189	11,6933	1,93

Fonte: Autoria própria..

Tabela 28 – Resultados obtidos pelo sistema ao utilizar três classificadores de fluxo de dados, com os módulos AID, APD e AFD e um Decisor, com a base de dados CICIDS2017 e ataques do tipo PortScan.

Classificador	**Corretas (%)**	**Kappa (%)**	**Kappa Temporal** (%)	**Kappa M (%)**	**RAM/Horas**	**Tempo (s)**
J48 PortScan	95,6558	96,0119	92,9001	92,9937	5,2903	0,78
Naive Bayes PortScan	97,1190	92,3348	1,6596	89,9338	7,2337	0,99
Perceptron PortScan	95,5663	92,0338	78,6449	92,9938	4,1967	2,98
Decisor PortScan	99,1873	98,9913	96,8014	97,0077	16,7207	2,98

Fonte: Autoria própria..

Tabela 29 – Resultados obtidos pelo sistema ao utilizar três classificadores de fluxo de dados, com os módulos AID, APD e AFD e um Decisor, com a base de dados CICIDS2017 e ataques do tipo DDoS.

Classificador	Corretas (%)	Kappa (%)	Kappa Temporal (%)	Kappa M (%)	RAM/Horas	Tempo (s)
J48 DDoS	99,8377	99,6955	-248,4937	99,6855	2,1873	1,09
Naive Bayes DDoS	98,9099	98,3008	48,6279	98,1089	4,8873	2,38
Perceptron DDoS	95,2937	93,8633	-87.224,1389	93,9008	3,0098	1,22
Decisor DDoS	99,9428	99,9087	84,2198	99,9007	10,0844	2,38

Fonte: Autoria própria..

Os resultados acima mostram que a taxa de acerto do decisor apresenta melhoras de até 0,24%, 1,90% e 0,60% em comparação aos classificadores J48, Naive Bayes e Perceptron, respectivamente, para ataques do tipo Botnet. Para ataques do tipo PortScan, as melhorias foram de 3,69%, 2,13% e 3,79% em comparação aos classificadores J48, Naive Bayes e Perceptron, respectivamente. Por último, para ataques do tipo DDOS, obteve-se melhorias na acurácia de 0,11%, 1,04% e 4,88% em comparação aos classificadores J48, Naive Bayes e Perceptron, respectivamente. Embora o sistema de votação apresente um aumento na taxa de acerto em todos os casos, observa-se que esse aumento não é tão acentuado ao se utilizar a base de dados CICIDS2017. Isso se deve ao fato da base de dados CICIDS2017 possuir ataques de um único tipo (Botnet, PortScan ou DDoS), resultando em um treinamento mais preciso para cada classificador quando comparado a base de dados NSL-KDD, que requer cada classificador detectar 39 tipos de ataques. Novamente, devido ao decisor necessitar das três classificações para obter a classificação final, o tempo de processamento é igual ao maior tempo de processamento entre os três classificadores. Finalmente, o consumo de memória ao utilizar três classificadores em paralelo resultou em um consumo maior de memória (RAM/Hora) por ser necessário construir três modelos de classificação simultâneos em memória.

5.5 Resultados Finais

Após obter os resultados para o sistema, utilizando duas bases de dados diferentes e dois tipos de classificadores, apresentamos o resumo dos resultados.

As Tabela 30, Tabela 31, Tabela 32 e Tabela 33 apresentam os resultados para os classificadores de dados e base de dados NSL-KDD; classificadores de dados e base de dados CICIDS2017; classificadores de fluxo de dados e base de dados NSL-KDD; e classificadores de fluxo de dados e base de dados CICIDS2017, respectivamente.

Tabela 30 – Resultados obtidos ao utilizar um classificador de dados, comparado ao resultado obtido pelo sistema proposto, utilizando a base de dados NSL-KDD.

Classificador	Corretas (%)	Taxa VP	Taxa FP	Precisão	Recall	F-Measure	Tempo (ms)
J48 Sem Redução	81,5339	0,815	0,146	0,858	0,815	0,836	198
Naive Bayes Sem Redução	76,1222	0,761	0,197	0,809	0,761	0,784	413
M. Perceptron Sem Redução	75,8738	0,759	0,199	0,808	0,759	0,783	949
J48 Com Redução	78,1627	0,782	0,172	0,839	0,782	0,809	88
Naive Bayes Com Redução	74,3524	0,744	0,209	0,802	0,744	0,772	228
M. Perceptron Com Redução	76,3174	0,763	0,197	0,808	0,763	0,785	612
J48 Duas Camadas	80,7448	0,809	0,155	0,851	0,809	0,829	114
Naive Bayes Duas Camadas	75,8778	0,755	0,201	0,806	0,755	0,780	257
M. Perceptron Duas Camadas	76,3174	0,763	0,197	0,808	0,763	0,785	612
Decisor	89,9265	0,887	0,068	0,929	0,892	0,910	612

Fonte: Autoria própria..

Conforme podemos observar nas tabelas anteriores, obtivemos um acréscimo na acurácia nos testes realizados ao utilizar o sistema proposto, ao custo de um acréscimo no tempo de processamento e requisitos de memória, para ambas as bases de dados NSL-KDD e CICIDS2017. Notamos também que é possível realizar um compromisso entre acurácia e tempo e memória, ao variarmos o coeficiente de corte entre as camadas de classificação de dados AID e APD. A quantidade de características também possui um forte impacto na acurácia, tempo de processamento e consumo de recursos, podendo ela ser ajustada na etapa de publicação das amostras e permitindo os classificadores sub-escreverem nos fluxos providos pelo agente de mensagens, escolhendo, assim, quantas características desejam utilizar nos classificadores.

Tabela 31 – Resultados obtidos ao utilizar um classificador de dados, comparado ao resultado obtido pelo sistema proposto, utilizando a base de dados CICIDS2017.

Classificador	Corretas (%)	Taxa VP	Taxa FP	Precisão	Recall	F-Measure	Tempo (ms)
J48 Botnet Sem	99,9817	1,000	0,005	1,000	1,000	1,000	492
J48 PortScan Sem	99,9878	1,000	0,016	1,000	1,000	1,000	441
J48 DDoS Sem	99,9911	1,000	0,000	1,000	1,000	1,000	384
Naive Bayes Botnet Sem	96,5635	0,966	0,003	0,992	0,966	0,976	564
Naive Bayes PortScan Sem	99,6876	0,997	0,403	0,997	0,997	0,997	779
Naive Bayes DDoS Sem	99,9003	0,999	0,001	0,999	0,999	0,999	615
M. Perceptron Botnet Sem	99,6493	0,996	0,363	0,997	0,996	0,996	872
M. Perceptron PortScan Sem	99,7347	0,997	0,323	0,997	0,997	0,997	1.255
M. Perceptron DDoS Sem	99,1783	0,992	0,011	0,992	0,992	0,992	1002
J48 Botnet Com	99,6519	0,997	0,338	0,996	0,997	0,996	28
J48 PortScan Com	99,8045	0,998	0,184	0,998	0,998	0,998	50
J48 DDoS Com	99,3754	0,994	0,005	0,994	0,994	0,994	46
Naive Bayes Botnet Com	95,0376	0,950	0,057	0,991	0,950	0,968	29
Naive Bayes PortScan Com	99,6457	0,996	0,110	0,997	0,996	0,997	58
Naive Bayes DDoS Com	99,0498	0,990	0,009	0,991	0,990	0,991	61
M. Perceptron Botnet Com	99,0630	0,991	0,966	0,990	0,991	0,986	29
M. Perceptron PortScan Com	77,1229	0,771	0,063	0,993	0,771	0,865	61
M. Perceptron DDoS Com	96,9435	0,969	0,028	0,970	0,969	0,969	59
J48 Botnet Duas	99,8536	0,999	0,120	0,998	0,998	0,998	64
J48 PortScan Duas	99,9559	0,999	0,052	0,999	0,999	0,999	97
J48 DDoS Duas	99,7812	0,998	0,002	0,998	0,998	0,998	88
Naive Bayes Botnet Duas	95,9771	0,961	0,012	0,984	0,961	0,970	55
Naive Bayes PortScan Duas	99,6603	0,997	0,172	0,997	0,996	0,997	121
Naive Bayes DDoS Duas	99,6691	0,998	0,003	0,998	0,998	0,998	149
M. Perceptron Botnet Duas	99,4811	0,994	0,418	0,995	0,994	0,994	48
M. Perceptron PortScan Duas	96,0018	0,968	0,387	0,968	0,968	0,968	131
M. Perceptron DDoS Duas	98,0212	0,989	0,016	0,989	0,989	0,989	114
Decisor Botnet	99,9912	1,000	0,002	1,000	1,000	1,000	64

Fonte: Autoria própria..

Tabela 32 – Resultados obtidos ao utilizar um classificador de fluxo de dados, comparado ao resultado obtido pelo sistema proposto, utilizando a base de dados NSL-KDD.

Classificador	**Corretas (%)**	**Kappa (%)**	**Kappa Temporal** (%)	**Kappa M (%)**	**RAM/Horas**	**Tempo (s)**
J48 Sem Redução	97,0094	93,9896	94,0239	93,6975	7,0460	1,64
Naive Bayes Sem Redução	86,8094	73,5441	73,7052	72,2689	9,2902	1,33
Perceptron Sem Redução	99,2957	98,5966	98,6056	98,5294	5,3190	219,89
J48 Com Redução	96,9485	93,8868	93,9021	93,6582	6,5444	1,52
Naive Bayes Com Redução	84,2309	71,3211	71,4817	70,5408	3,7893	0,84
Perceptron Com Redução	99,1684	98,3341	98,3382	98,2718	7,7055	208,56
J48 Duas Camadas	96,9989	93,9278	93,9774	93,6701	6,8864	1,55
Naive Bayes Duas Camadas	86,4419	73,2009	73,2297	71,9088	4,2559	0,91
M. Perceptron Duas Camadas	99,2514	98,5478	98,5771	98,4930	5,9634	212,97
Decisor	99,8852	99,8361	99,8749	99,8408	17,1057	212,97

Fonte: Autoria própria..

Tabela 33 – Resultados obtidos ao utilizar um classificador de fluxo de dados, comparado ao resultado obtido pelo sistema proposto, utilizando a base de dados CICIDS2017.

Classificador	Corretas (%)	Kappa (%)	Kappa Temporal (%)	Kappa M (%)	RAM/Horas	Tempo (s)
J48 Botnet Sem	99,6739	83,9788	78,2168	68,3113	2,7437	4,21
J48 PortScan Sem	99,9658	99,9308	98,0423	99,9231	5,0184	5,19
J48 DDoS Sem	99,8419	99,6780	-711,3636	99,6347	6,8211	4,66
Naive Bayes Botnet Sem	98,3329	92,8752	-679,3007	98,6724	3,3399	3,55
Naive Bayes PortScan Sem	98,4225	96,8057	9,7283	96,4567	7,0937	4,95
Naive Bayes DDoS Sem	99,0932	98,1502	-4.552,2727	97,9052	6,3552	4,44
Perceptron Botnet Sem	99,4173	72,1631	61,0839	43,3875	2,7211	6,20
Perceptron PortScan Sem	99,8481	99,6926	91,3104	99,6589	2,9283	7,09
Perceptron DDoS Sem	99,9588	99,9160	-111,3636	99,9048	2,1320	5,09
J48 Botnet Com	99,5189	72,3073	67,8671	53,2553	9,0500	0,57
J48 PortScan Com	80,5478	-0,7852	-1.199,3007	-1.790,1322	5,7722	0,32
J48 DDoS Com	99,7588	99,7160	-111,1616	99,7048	1,6635	0,56
Naive Bayes Botnet Com	97,8551	97,7067	89,7099	97,6746	6,4251	0,45
Naive Bayes PortScan Com	94,0356	87,9015	-241,3104	86,6031	7,9712	0,45
Naive Bayes DDoS Com	98,6805	98,3534	80,7219	98,2825	3,3138	1,01
Perceptron Botnet Com	99,2119	98,3970	-3.943,1818	98,1794	1,0473	0,48
Perceptron PortScan Com	74,3604	49,9299	-131.445,4545	40,7677	7,6834	0,42
Perceptron DDoS Com	87,9599	76,0650	-61.672,7272	72,1850	4,4868	0,48
J48 Botnet Duas	99,6421	81,0219	76,9330	65,4739	4,8806	1,22
J48 PortScan Duas	95,6558	96,0119	92,9001	92,9937	5,2903	0,78
J48 DDoS Duas	99,8377	99,6955	-248,4937	99,6855	2,1873	1,09
Naive Bayes Botnet Duas	98,0228	95,9980	64,0019	98,0993	4,9637	0,89
Naive Bayes PortScan Duas	97,1190	92,3348	1,6596	89,9338	7,2337	0,99
Naive Bayes DDoS Duas	98,9099	98,3008	48,6279	98,1089	4,8873	2,38
Perceptron Botnet Duas	99,2884	92,9552	39,5591	91,0029	1,8490	1,93
Perceptron PortScan Duas	95,5663	92,0338	78,6449	92,9938	4,1967	2,98
Perceptron DDoS Duas	95,2937	93,8633	-87.224,1389	93,9008	3,0098	1,22
Decisor Botnet	99,8817	98,5568	88,9771	98,6189	11,6933	1,93

Fonte: Autoria própria..

6 CONCLUSÕES E PERSPECTIVAS

Este trabalho propôs a criação de uma arquitetura de detecção de intrusão distribuída utilizando diversas camadas de classificação, seleção de características e classificadores de fluxo de dados.

Para analisarmos os resultados finais da proposta, coletamos dados durante diferentes estágios de implementação da arquitetura. Para obtermos uma sequência de amostras, utilizamos duas bases de dados bem conhecidas na comunidade: NSL-KDD e CICIDS2017. A partir dos dados contidos nesses arquivos, foi possível criar um fluxo de amostras contendo 41 e 84 características, respectivamente. A seguir, cada base de dados foi utilizada separadamente para validarmos a arquitetura com *datasets* diferentes. O fluxo de dados foi submetido ao pré-processamento dos dados coletados, onde ocorreram normalizações nas características necessárias. então, esse fluxo foi publicado através do agente de mensagens (publicador). O módulo de análise e classificação dos dados recebe o fluxo de amostras através do seu agente de mensagens (assinante) e a encaminha para as unidades de processamento e classificação de dados (UPCD).

Inicialmente montamos apenas uma camada de classificação e utilizamos apenas um classificador utilizando todas as características da amostra, obtendo uma acurácia que serve de base para comparações. Esses resultados representam o uso de um classificador comum (ou seja, não um classificador de fluxo de dados) para classificar a amostra em normal ou ataque. Uma vez coletados os resultados de classificação para ambas as bases de dados, modificamos a UPCD. No segundo momento, passamos a realizar a seleção das características das amostras, através de uma lista pré-processada durante o treinamento, contendo em ordem decrescente de relevância, todas as características. Assim, realizamos a classificação do fluxo de amostras utilizando um número menor de *features* para ambas as bases. A próxima modificação na UPCD consiste em utilizar duas camadas de classificação. A primeira camada possui um classificador e realiza operações com seleção de características. A segunda camada contém o mesmo classificador, porém trabalha utilizando todas as características da amostra. Novamente, coletamos os resultados para ambas as bases de dados. Por último, modificamos a UPCD para representar a arquitetura proposta, com três camadas de classificação. A terceira camada é composta por três classificadores diferentes, trabalhando em paralelo e as três classificações resultantes são enviadas para um decisor, que irá determinar a classificação da amostra através de um sistema de votação. Após todos os resultados serem coletados, para ambas as bases, repetimos todo o processo porém agora utilizando classificadores de fluxo de dados.

Comparando os resultados obtidos pela arquitetura proposta e os demais métodos de classificação não de fluxo, para a base de dados NSL-KDD, obtivemos respectivamente ganhos na precisão e no *recall* de até 15,84% e 19,89% no melhor caso (uma camada de classificação, Naive Bayes e com seleção de características), e 8,28% e 9,45% no pior caso (uma camada

de classificação, J48 e sem redução de características). Entretanto, tivemos aumento no tempo necessário para classificação das amostras em 595,45% no pior caso (uma camada de classificação, J48 e sem redução de características), enquanto no melhor caso obtivemos um ganho de velocidade de 35,51% (uma camada de classificação, M. Perceptron e sem redução de características).

Para a base de dados CICIDS2017, obtivemos respectivamente ganhos na precisão e no *recall* de até 3,31% (duas camadas de classificação, M. Perceptron e PortScan) e 29,70% (uma camada de classificação, M. Perceptron, PortScan e com seleção de características) no melhor caso, e 0% (nenhum ganho em precisão ou *recall*) no pior caso (uma camada de classificação, J48, todos os tipos de ataques e sem seleção de características). Também tivemos aumento no tempo necessário para classificação das amostras em até 128,57% no pior caso (uma camada de classificação, J48, Botnet e com redução de características), enquanto no melhor caso obtivemos um ganho de velocidade de até 1.860,94% (uma camada de classificação, M. Perceptron, PortScan e sem redução de características).

Trocando os classificadores para classificadores de fluxo de dados, para a base de dados NSL-KDD, obtivemos melhorias no Kappa em até 39,98% (um classificador, Naive Bayes com redução das características) no melhor caso, e de até 1,26% (um classificador, Perceptron, sem redução das características) no pior caso. O aumento no consumo de RAM/Horas no pior caso foi de 351,42% (um classificador, Naive Bayes, com redução das características) e no melhor caso foi de 84,17% (um classificador, Naive Bayes, sem redução das características). Por fim, houve um aumento no tempo de classificação de até 25.253,57% (um classificador, Naive Bayes, com redução das características) no pior caso, porém um ganho de até 3,25% (um classificador, Perceptron, sem redução das características) no melhor caso.

Para a base de dados CICIDS2017, obtivemos melhorias no Kappa em até 97,39% (um classificador, Perceptron, PortScan, com Redução das características) no melhor caso, e redução de até 1,39% (um classificador, J48, PortScan, sem redução das características) no pior caso. O aumento no consumo de RAM/Horas no pior caso foi de 1.016,52% (um classificador, Perceptron, Botnet, com redução das características) e no melhor caso foi de 29,21% (um classificador, J48, Botnet, com redução das características). Por fim, houve um aumento no tempo de classificação de até 503,13% (um classificador, J48, PortScan, com redução das características) no pior caso, porém um ganho de até 267,36% (um classificador, Perceptron, PortScan, sem redução das características) no melhor caso.

Com base nos resultados obtidos, a arquitetura proposta apresenta ganhos na acurácia em relação aos demais métodos propostos na literatura. Existe um aumento nos recursos de memória detectado, porém a arquitetura é distribuída, podendo esta sobrecarga de recursos ser compartilhada entre diversos equipamentos. Existe, em muitos casos, um aumento no tempo de classificação das amostras. Entretanto, novamente, pode-se criar (e desativar) novos módulos de análise e classificação de dados *on-the-fly*, pois a arquitetura permite elevado grau de escalabilidade. Por fim, é possível ajustar diversos parâmetros na arquitetura para obter um melhor

relacionamento entre acurácia e tempo de classificação, permitindo a proposta alcançar uma flexibilização ainda maior.

Para trabalhos futuros, propõe-se utilizar outras bases de dados para validação, além da criação de uma base de dados própria, contendo ataques mais recentes. Outra proposta é utilizar mais classificadores de fluxo de dados além dos três apresentados neste trabalho, além de utilizar mais do que três classificadores na terceira camada de classificação. Outra proposta é permitir uma unidade de análise e classificação de dados se comunicar com outras unidades, podendo, assim, trocar informações sobre potenciais ataques que estão surgindo na rede, ou solicitar ajudar na classificação, quando necessário. Também propomos trabalhar com detecção de ataques por anomalias e não apenas por assinaturas. Por fim, espera-se trabalhar com variações na quantidade de características selecionadas, permitindo um controle maior sobre a relação acurácia por tempo de classificação.

Assim, ao término deste trabalho foi possível:

a) Desenvolver uma arquitetura distribuída capaz de analisar em tempo real o tráfego de rede, utilizando ferramentas de código aberto e gratuitas;

b) Coletar, processar, analisar e caracterizar grandes volumes de dados (*big data*);

c) Construir um fluxo de amostras a partir da extração de características dos pacotes que trafegam em uma rede;

d) Realizar a seleção das características mais importantes para detecção de ameaças;

e) Classificar as amostras através de diferentes métodos de aprendizagem de máquina através do método supervisionado, de forma paralela;

f) Obter uma relação entre acurácia e tempo de classificação das amostras através de diversas camadas de classificação;

g) Obter uma maior acurácia através de um sistema de votação;

h) Comparar os resultados obtidos em várias etapas para demostrar os ganhos obtidos ao utilizar a arquitetura proposta;

i) Demonstrar o desempenho da arquitetura proposta através de ferramentas, técnicas e procedimentos de forma integrada.

REFERÊNCIAS

AGGARWAL, C. C. **Data Classification: Algorithms and Applications**. 1st. ed. [*S.l.*]: Chapman & Hall/CRC, 2014. ISBN 1466586745, 9781466586741.

AHMAD, I.; ABDULLAH, A. B.; ALGHAMDI, A. S. Applying neural network to u2r attacks. *In*: **2010 IEEE Symposium on Industrial Electronics and Applications (ISIEA)**. [*S.l.*: *s.n.*], 2010. p. 295–299.

ALGHUSHAIRY, O.; ALSINI, R.; MA, X. An efficient local outlier factor for data stream processing: A case study. *In*: **2020 International Conference on Computational Science and Computational Intelligence (CSCI)**. [*S.l.*: *s.n.*], 2020. p. 1525–1528.

Alhaj, T. A. *et al.* Feature selection using information gain for improved structural-based alert correlation. **Public Library of Science**, v. 11, n. 11, p. e0166017, November 2016. ISSN 1932-6203.

ALOM, M. Z.; TAHA, T. M. Network intrusion detection for cyber security using unsupervised deep learning approaches. *In*: **2017 IEEE National Aerospace and Electronics Conference (NAECON)**. [*S.l.*: *s.n.*], 2017. p. 63–69.

ALSAFI, H. A review of intrusion detection system schemes in wireless sensor network. 10 2013.

ANDREONI, M. *et al.* **CATRACA: uma Ferramenta para Classificação e Análise de Tráfego Escalável Baseada em Processamento por Fluxo**. 2017.

ANSARI, S.; RAJEEV, S.; CHANDRASHEKAR, H. Packet sniffing: a brief introduction. **IEEE Potentials**, v. 21, n. 5, p. 17–19, 2003.

Apache Software Foundation. **Apache Hadoop**. 2006. Acessado em: 27-03-2017. Disponível em: https://hadoop.apache.org/.

Apache Software Foundation. **Apache Zookeeper**. 2010. Acessado em: 27-03-2017. Disponível em: https://zookeeper.apache.org.

Apache Software Foundation. **Apache Spark**. 2013. Acessado em: 27-03-2017. Disponível em: https://spark.apache.org/.

Apache Software Foundation. **Apache Flink**. 2014. Acessado em: 27-03-2017. Disponível em: https://flink.apache.org/.

Apache Software Foundation. **Apache Storm**. 2015. Acessado em: 27-03-2017. Disponível em: https://storm.apache.org.

Apache Software Foundation. **Apache Kafka**. 2016. Acessado em: 27-03-2017. Disponível em: https://kafka.apache.org.

BALUPARI, R. *et al.* Real-time system security. *In*: TJADEN, B.; WELCH, L. R. (Ed.). Commack, NY, USA: Nova Science Publishers, Inc., 2003. cap. Real-time Network-based Anomaly Intrusion Detection, p. 1–19. ISBN 1-59033-586-4. Disponível em: http://dl.acm.org/citation.cfm?id=903866.903869.

BEREZIÇNSKI, P.; JASIUL, B.; SZPYRKA, M. An entropy-based network anomaly detection method. **Entropy**, v. 17, n. 4, p. 2367–2408, 2015. ISSN 1099-4300. Disponível em: https://www.mdpi.com/1099-4300/17/4/2367.

BHARGAVA, N. *et al.* Decision tree analysis on j48 algorithm for data mining. **Proceedings of International Journal of Advanced Research in Computer Science and Software Engineering**, v. 3, n. 6, 2013.

BIFET, A. *et al.* Fast perceptron decision tree learning from evolving data streams. *In*: . [*S.l.*: *s.n.*], 1970. p. 299–310. ISBN 978-3-642-13671-9.

BIFET, A. *et al.* Moa: A real-time analytics open source framework. *In*: **Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part III**. Berlin, Heidelberg: Springer-Verlag, 2011. (ECML PKDD'11), p. 617–620. ISBN 978-3-642-23807-9. Disponível em: http://dl.acm.org/citation.cfm?id=2034161.2034204.

BIFET, A.; KIRKBY, R. Data stream mining a practical approach. *In*: . [*S.l.*: *s.n.*], 2009.

BIFET, A. *et al.* Efficient online evaluation of big data stream classifiers. *In*: **Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: Association for Computing Machinery, 2015. (KDD '15), p. 59–68. ISBN 9781450336642. Disponível em: https://doi.org/10.1145/2783258.2783372.

BIFET, A. *et al.* Streaming data mining with massive online analytics (moa). *In*: ____. [*S.l.*: *s.n.*], 2018. p. 1–25. ISBN 978-981-322-803-0.

CHAE, H.-s. *et al.* Feature selection for intrusion detection using nsl-kdd. **Recent advances in computer science**, v. 20132, p. 184–187, 2013.

CHEN, F. *et al.* Data mining for the internet of things: Literature review and challenges. **International Journal of Distributed Sensor Networks**, v. 11, n. 8, p. 431047, 2015. Disponível em: https://doi.org/10.1155/2015/431047.

CHOUVATUT, V.; JINDALUANG, W.; BOONCHIENG, E. Training set size reduction in large dataset problems. *In*: **2015 International Computer Science and Engineering Conference (ICSEC)**. [*S.l.*: *s.n.*], 2015. p. 1–5.

CICFLOWMETER. **CICFlowMeter - A network traffic Biflow generator and analyzer**. 2017. Acessado em: 15-08-2018. Disponível em: http://www.netflowmeter.ca/index.html.

CLAY, P. A modern threat response framework. **Network Security**, v. 2015, n. 4, p. 5 – 10, 2015. ISSN 1353-4858. Disponível em: http://www.sciencedirect.com/science/article/pii/S135348581530026X.

CORRÊA, D. G.; ENEMBRECK, F.; SILLA, C. N. An investigation of the hoeffding adaptive tree for the problem of network intrusion detection. *In*: **2017 International Joint Conference on Neural Networks (IJCNN)**. [*S.l.*: *s.n.*], 2017. p. 4065–4072. ISSN 2161-4407.

CROWDSTRIKE. **Global Threat Report**. 2021. Disponível em: https://go.crowdstrike.com/rs/281-OBQ-266/images/Report2021GTR.pdf.

DESALE, K. S.; KUMATHEKAR, C. N.; CHAVAN, A. P. Efficient intrusion detection system using stream data mining classification technique. *In*: **2015 International Conference on Computing Communication Control and Automation**. [*S.l.*: *s.n.*], 2015. p. 469–473.

DEVIKRISHNA, K. S.; RAMAKRISHNA, B. An artificial neural network based intrusion detection system and classification of attacks. **International Journal of Engineering Research and Applications**, v. 3, n. 4, p. 1959–1964, Jul-Aug 2013.

DOMINGOS, P.; HULTEN, G. Mining high-speed data streams. *In*: **Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: Association for Computing Machinery, 2000. (KDD '00), p. 71–80. ISBN 1581132336. Disponível em: https://doi.org/10.1145/347090.347107.

DURAISAMY, B.; CHAKRABARTI, A.; MIDHUNCHAKKARAVARTHY, D. Smart devices threats, vulnerabilities and malware detection approaches: A survey. **European Journal of Engineering Research and Science**, v. 3, p. 7, 02 2018.

ELMOMEN, A. A.; DIN, A. B. E.; WAHDAN, A. Detecting abnormal network traffic in the secure event management systems. **International Conference on Aerospace Sciences and Aviation Technology**, The Military Technical College, v. 14, n. AEROSPACE SCIENCES; AVIATION TECHNOLOGY, ASAT - 14; May 24 - 26, 2011, p. 1–15, 2011. ISSN 2090-0678.

FABRIS, F.; MAGALHAES, J. P. de; FREITAS, A. A review of supervised machine learning applied to ageing research. **Biogerontology**, v. 18, 04 2017.

GADAL, S. *et al.* Machine learning-based anomaly detection using k-mean array and sequential minimal optimization. **Electronics**, v. 11, n. 14, 2022. ISSN 2079-9292. Disponível em: https://www.mdpi.com/2079-9292/11/14/2158.

GARTNER. **Gartner Says By 2020, More Than Half of Major New Business Processes and Systems Will Incorporate Some Element of the Internet of Things**. 2016. Acessado em: 07-11-2018. Disponível em: https://www.gartner.com/newsroom/id/3185623.

GLUHAK, A. *et al.* A survey on facilities for experimental internet of things research. **IEEE Communications Magazine**, v. 49, n. 11, p. 58–67, November 2011. ISSN 0163-6804.

GOMES, H. M. *et al.* Adaptive random forests for evolving data stream classification. **Machine Learning**, v. 106, p. 1–27, 10 2017.

HALL, M. *et al.* The WEKA data mining software: an update. **SIGKDD Explorations**, v. 11, n. 1, p. 10–18, 2009.

HANSEN, L.; SALAMON, P. Neural network ensembles. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 12, n. 10, p. 993–1001, 1990.

HARBI, N.; BAHRI, E. Real detection intrusion using supervised and unsupervised learning. *In*: **2013 International Conference on Soft Computing and Pattern Recognition (SoCPaR)**. [*S.l.*: *s.n.*], 2013. p. 321–326.

HASAN, M. A. *et al.* Performance evaluation of different kernels for support vector machine used in intrusion detection system. **International journal of Computer Networks and Communications**, v. 8, p. 39–53, 11 2016.

HASTIE, T. *et al.* The elements of statistical learning: Data mining, inference, and prediction. **Math. Intell.**, v. 27, p. 83–85, 11 2004.

HEBA, F. E. *et al.* Principle components analysis and support vector machine based intrusion detection system. *In*: **2010 10th International Conference on Intelligent Systems Design and Applications**. [*S.l.*: *s.n.*], 2010. p. 363–367.

HOLTZ, M. D.; DAVID, B. M.; JÚNIOR, R. T. de S. Building scalable distributed intrusion detection systems based on the mapreduce framework. **Revista Telecommun**, v. 13, n. 2, p. 22, 2011.

HSU, C.-H.; WANG, S.-D. An embedded nids with multi-core aware packet capture. *In*: **2013 IEEE 16th International Conference on Computational Science and Engineering**. [*S.l.*: *s.n.*], 2013. p. 778–785.

III, H. D. **A Course in Machine Learning**. 2020. Acessado em: 12-11-2021. Disponível em: http://ciml.info/dl/v0_99/ciml-v0_99-all.pdf.

Information and Computer Science University of California. **KDD Cup 1999 Data**. 1999. Acessado em: 27-03-2017. Disponível em: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

JASWAL, K.; KUMAR, P.; RAWAT, S. Design and development of a prototype application for intrusion detection using data mining. *In*: **2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions)**. [*S.l.*: *s.n.*], 2015. p. 1–6.

JOHN, G. H.; LANGLEY, P. Estimating continuous distributions in bayesian classifiers. *In*: MORGAN KAUFMANN PUBLISHERS INC. **Proceedings of the Eleventh conference on Uncertainty in artificial intelligence**. [*S.l.*], 1995. p. 338–345.

KARIMI, Z.; KASHANI, M. R.; HAROUNABADI, A. Feature ranking in intrusion detection dataset using combination of filtering methods. **International Journal of Computer Applications**, v. 78, p. 21–27, 09 2013.

Kezih, M.; Taibi, M. Evaluation effectiveness of intrusion detection system with reduced dimension using data mining classification tools. *In*: **2nd International Conference on Systems and Computer Science**. [*S.l.*: *s.n.*], 2013. p. 205–209.

KHALID, S.; KHALIL, T.; NASREEN, S. A survey of feature selection and feature extraction techniques in machine learning. *In*: **2014 Science and Information Conference**. [*S.l.*: *s.n.*], 2014. p. 372–378.

KIM, K.; AMINANTO, M. E. Deep learning in intrusion detection perspective: Overview and further challenges. *In*: **2017 International Workshop on Big Data and Information Security (IWBIS)**. [*S.l.*: *s.n.*], 2017. p. 5–10.

KOHAVI, R.; JOHN, G. H. Wrappers for feature subset selection. **Artificial Intelligence**, v. 97, n. 1, p. 273–324, 1997. ISSN 0004-3702. Relevance. Disponível em: https://www.sciencedirect.com/science/article/pii/S000437029700043X.

KUL, S.; SAYAR, A. A survey of publish/subscribe middleware systems for microservice communication. *In*: **2021 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)**. [*S.l.*: *s.n.*], 2021. p. 781–785.

Kurniabudi *et al.* Cicids-2017 dataset feature analysis with information gain for anomaly detection. **IEEE Access**, v. 8, p. 132911–132921, 2020.

LABS, F. **The Hunt for IoT: The Rise of Thingbots**. 2017. Acessado em: 07-11-2018. Disponível em: https://www.f5.com/labs/articles/threat-intelligence/the-hunt-for-iot-the-rise-of-thingbots.

LASHKARI, A. H. *et al.* **CICFlowMeter - A network traffic Biflow generator and analyzer**. 2017. Acessado em: 15-08-2018. Disponível em: http://www.netflowmeter.ca/index.html.

LAZAREVIC, A.; KUMAR, V.; SRIVASTAVA, J. Intrusion detection: A survey. *In*: ____. [*S.l.*: *s.n.*], 2005. v. 5, p. 19–78.

LEU, F. Y. *et al.* An internal intrusion detection and protection system by using data mining and forensic techniques. **IEEE Systems Journal**, PP, n. 99, p. 1–12, 2015. ISSN 1932-8184.

LOBATO, A. G. P.; ANDREONI, M.; DUARTE, O. C. M. B. Um sistema acurado de detecção de ameaças em tempo real por processamento de fluxos. *In*: . [*S.l.*: *s.n.*], 2016.

LOPEZ, M. A.; LOBATO, A. G. P.; DUARTE, O. C. M. B. A performance comparison of open-source stream processing platforms. *In*: **2016 IEEE Global Communications Conference (GLOBECOM)**. [*S.l.*: *s.n.*], 2016. p. 1–6.

LOPEZ, M. A. *et al.* Design and performance evaluation of a virtualized network function for real-time threat detection using stream processing. *In*: . [*S.l.*: *s.n.*], 2016.

LOPEZ, M. A. *et al.* An evaluation of a virtual network function for real-time threat detection using stream processing. *In*: **2018 Fourth International Conference on Mobile and Secure Services (MobiSecServ)**. [*S.l.*: *s.n.*], 2018. p. 1–5.

LOPEZ, M. A.; MATTOS, D. M. F.; DUARTE, O. C. M. B. An elastic intrusion detection system for software networks. **Annals of Telecommunications**, v. 71, n. 11, p. 595–605, Dec 2016. ISSN 1958-9395. Disponível em: https://doi.org/10.1007/s12243-016-0506-y.

LOPEZ, M. A. *et al.* Collecting and characterizing a real broadband access network traffic dataset. *In*: **2017 1st Cyber Security in Networking Conference (CSNet)**. [*S.l.*: *s.n.*], 2017. p. 1–8.

MAHFOUZ, A. M.; VENUGOPAL, D.; SHIVA, S. G. Comparative analysis of ml classifiers for network intrusion detection. *In*: YANG, X.-S. *et al.* (Ed.). **Fourth International Congress on Information and Communication Technology**. Singapore: Springer Singapore, 2020. p. 193–207. ISBN 978-981-32-9343-4.

MARZ NATHAN; WARREN, J. **Big Data: Principles and best practices of scalable realtime data systems**. [*S.l.*]: Manning Publications, 2013.

MASUD, M. M. *et al.* Flow-based identification of botnet traffic by mining multiple log files. *In*: **2008 First International Conference on Distributed Framework and Applications**. [*S.l.*: *s.n.*], 2008. p. 200–206.

MCHUGH, M. Interrater reliability: The kappa statistic. **Biochemia medica : časopis Hrvatskoga društva medicinskih biokemičara / HDMB**, v. 22, p. 276–82, 10 2012.

PATEL, H.; PATEL, H.; BHATT, N. Empirical analysis on stream classification and clustering with concept drift in moa. **International Journal of Computer Sciences and Engineering**, v. 6, p. 341–345, 10 2018.

PECHT, M. G.; KANG, M. Machine learning: Fundamentals. *In*: ____. **Prognostics and Health Management of Electronics: Fundamentals, Machine Learning, and the Internet of Things**. IEEE, 2019. ISBN 9781119515326. Disponível em: https://ieeexplore.ieee.org/document/8471050.

QUINLAN, J. R. **C4. 5: programs for machine learning**. [*S.l.*]: Elsevier, 2014.

RATHORE, M. M. *et al.* Hadoop based real-time intrusion detection for high-speed networks. *In*: **2016 IEEE Global Communications Conference (GLOBECOM)**. [*S.l.*: *s.n.*], 2016. p. 1–6.

RYU, J. W.; KANTARDZIC, M. M.; WALGAMPAYA, C. Ensemble classifier based on misclassified streaming data. *In*: . [*S.l.*: *s.n.*], 2010.

SCHUARTZ, F. C.; FONSECA, M.; MUNARETTO, A. Sistema distribuído para detecção de ameaças em tempo real utilizando big data. *In*: **XXXV Simpósio Brasileiro de Telecomunicações e Processamento de Sinais - SBrT 2017**. [*S.l.*: *s.n.*], 2017.

SCHUARTZ, F. C.; FONSECA, M. S. P.; FONSECA, A. M. Sistema distribuído para detecção de ameaças em tempo real utilizando big data. *In*: **XXXV Simpósio Brasileirp de Telecomunicações e Processamento de Sinais - SBrT 2017**. [*S.l.*: *s.n.*], 2017. p. 472–476.

SCHUARTZ, F. C.; FONSECA, M. S. P.; MUNARETTO, A. Distributed system for threat detection in networks using machine learning. *In*: **1st Blockchain, Robotics and AI for Networking Security Conference - BRAINS 2019**. [*S.l.*: *s.n.*], 2019.

SETH, S.; SINGH, G.; CHAHAL, K. K. Drift-based approach for evolving data stream classification in intrusion detection system. *In*: . [*S.l.*: *s.n.*], 2021.

SHARAFALDIN, I.; LASHKARI, A. H.; GHORBANI, A. A. **CICIDS2017**. 2017. Acessado em: 15-08-2018. Disponível em: https://www.unb.ca/cic/datasets/ids-2017.html.

SHARAFALDIN, I.; LASHKARI, A. H.; GHORBANI, A. A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *In*: INSTICC. **Proceedings of the 4th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,**. [*S.l.*]: SciTePress, 2018. p. 108–116. ISBN 978-989-758-282-0.

SHONE, N. *et al.* A deep learning approach to network intrusion detection. **IEEE Transactions on Emerging Topics in Computational Intelligence**, v. 2, n. 1, p. 41–50, Feb 2018.

SI, W.; LI, J.-H.; HUANG, X.-J. Features extraction based on deep analysis of network packets in industrial control systems. *In*: XU, Y. *et al.* (Ed.). **Nuclear Power Plants: Innovative Technologies for Instrumentation and Control Systems**. Singapore: Springer Singapore, 2020. p. 524–529. ISBN 978-981-15-1876-8.

SYMANTEC. **Internet Security Threat Report**. 2019. Disponível em: https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf.

TAN, Z. *et al.* Enhancing big data security with collaborative intrusion detection. **IEEE Cloud Computing**, v. 1, n. 3, p. 27–33, Sept 2014. ISSN 2325-6095.

TAVALLAEE, M. *et al.* A detailed analysis of the kdd cup 99 data set. *In*: **2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications**. [*S.l.*: *s.n.*], 2009. p. 1–6.

TSYMBAL, A. The problem of concept drift: Definitions and related work. 05 2004.

VAN, N. T.; THINH, T. N.; SACH, L. T. An anomaly-based network intrusion detection system using deep learning. *In*: **2017 International Conference on System Science and Engineering (ICSSE)**. [*S.l.*: *s.n.*], 2017. p. 210–214.

Vinayakumar, R. *et al.* Deep learning approach for intelligent intrusion detection system. **IEEE Access**, v. 7, p. 41525–41550, 2019.

WANG, F.; WANG, H.; XUE, L. Research on data security in big data cloud computing environment. *In*: **2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)**. [*S.l.*: *s.n.*], 2021. v. 5, p. 1446–1450.

WIDMER, G.; KUBAT, M. Learning in the presence of concept drift and hidden contexts. **Machine Learning**, v. 23, p. 69–101, 04 1996.

WU, K. *et al.* Rs-forest: A rapid density estimator for streaming anomaly detection. *In*: **2014 IEEE International Conference on Data Mining**. [*S.l.*: *s.n.*], 2014. p. 600–609. ISSN 1550-4786.

YADAV, S.; SHUKLA, S. Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification. *In*: **2016 IEEE 6th International Conference on Advanced Computing (IACC)**. [*S.l.*: *s.n.*], 2016. p. 78–83.

YANG, K.; KPOTUFE, S.; FEAMSTER, N. Feature extraction for novelty detection in network traffic. **arXiv preprint arXiv:2006.16993**, 2020.

ŽLIOBAITé, I.; BIFET, A.; READ, J. e. a. Evaluation methods and decision theory for classification of streaming data with temporal dependence. **Machine Learning**, v. 98, p. 455–482, 04 2014.