Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Detecção de Ataque DDoS em SDN Utilizando Entropia e Machine Learning

Marcos José dos Santos Neto

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Orientador
Prof. Dr. Edison Ishikawa

Coorientador
Prof. Dr. Jacir Luiz Bordim

Brasília
2021

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

Santos Neto, Marcos José dos
SS237d Detecção de Ataque DDoS em SDN Utilizando Entropia e Machine Learning / Marcos José dos Santos Neto; orientador Edison Ishikawa; co-orientador Jacir Luiz Bordim. -- Brasília, 2021.
108 p.

Dissertação (Mestrado - Mestrado Profissional em Computação Aplicada) -- Universidade de Brasília, 2021.

1. SDN. 2. Entropia. 3. Machine Learning. 4. DDoS. 5. SVR. I. Ishikawa, Edison , orient. II. Bordim, Jacir Luiz , co-orient. III. Título.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Detecção de Ataque DDoS em SDN Utilizando Entropia e Machine Learning

Marcos José dos Santos Neto

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Prof. Dr. Edison Ishikawa (Orientador)
CIC/UnB

Prof. Dr. Marcelo Antônio Marotta
CIC/UNB

Prof. Dr. Luciano Paschoal Gaspary
INF/UFRGS

Prof. Dr. Marcelo Ladeira
Coordenador do Programa de Pós-graduação em Computação Aplicada

Brasília, 13 de janeiro de 2021

Dedicatória

Dedico este trabalho à minha esposa Thábata e aos meus filhos Felipe, Lucas, Heitor e Miguel que sempre me apoiaram e souberam entender minhas ausências durante os longos momentos de estudos e dedicação as pesquisas para conclusão do mestrado. Obrigado!

Agradecimentos

Agradeço em primeiro lugar à Deus, por me guiar e me dar forças durante toda a trajetória.

Agradeço aos meus orientadores Prof. Dr. Edison Ishikawa, Prof. Dr. Jacir Luiz Bordim e Prof. Dr. Eduardo Adílio Pelinson Alchieri, por todo direcionamento, conselhos, ensinamentos e confiança generosamente dados a mim durante a realização deste trabalho.

Agradeço à minha esposa Thábata e meus filhos Felipe, Lucas, Heitor e Miguel por todo amor, apoio e compreensão. Aos meus pais Dionízio e Argentina (*In memoriam*) por me ensinarem todos os valores que carrego.

Agradeço ao amigo Leonardo dos Santos Dourado, que contribuiu com o trabalho de disciplina que serviu de ideia base para esta dissertação.

Agradeço aos amigos do COMNET, por toda a ajuda técnica, companheirismo e compartilhamento de todos os momentos característicos da vida de pesquisa.

Agradeço aos colegas da empresa em que a solução foi implementada, por todo apoio e disponibilidade na implementação deste trabalho de pesquisa no ambiente corporativo.

Agradeço a todos os professores, amigos e colegas do PPCA, com os quais, durante a realização das disciplinas, pude aprender lições de pesquisa e de vida, além de todas as conversas motivadoras e conselhos.

Agradeço a todos os meus irmãos e amigos por me apoiarem e sempre torcerem por mim.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

Redes Definidas por Software (do inglês - *Software-Defined Networking, SDN*) traz um novo conceito em termos de arquitetura de rede de dados, em resposta às limitações das redes tradicionais, mas ao mesmo tempo são introduzidos novos desafios de segurança, para uma rede programável com visão global. Muitos artigos na literatura têm usado técnicas estatísticas, Entropia e Aprendizagem de Máquina (do inglês - *Machine Learning, ML*), isoladamente para detectar ataques DDoS em SDN. A principal questão abordada por estes trabalhos é estabelecer um limite que melhor separe o tráfego normal do espúrio, além da ausência de um limiar dinâmico que melhore a eficiência dos algoritmos baseados em entropia. Este trabalho explora a utilização de ML para o cálculo dinâmico dos limiares de algoritmos que utilizam entropia e apresenta um método híbrido utilizando entropia e ML, aqui denominado ML-Entropia, para diferenciá-lo da entropia tradicional e também três alternativas híbridas de ML (SVR híbrido, SVC híbrido e *Random Forest* híbrido), melhorando assim seus resultados na detecção com redução das taxas de erro em relação a entropia tradicional. Como contribuições são apresentados: Um método que melhor distinga o tráfego legítimo do espúrio em ML-Entropia e mais três técnicas alternativas viáveis; Usamos o ambiente Real para validar o experimento e o método; Uso de tráfego real ao invés de apenas o sintético na validação; Uso de uma técnica na coleta de dados que não induz sobrecarga ao controlador e finalmente apresentando um método híbrido que extrai o melhor de ambas as técnicas utilizadas, reduzindo a taxa de erro e habilitando uma técnica que permite um Limiar Dinâmico. Os resultados, em ambiente real e emulado, das quatro técnicas são todos comparados com entropia e entre si e também com três trabalhos na literatura, onde nossos resultados foram superiores em relação ao estado da arte com taxas de acerto acima de 99%, taxas de erro abaixo de 1% e além disso a ML-Entropia melhorou em até 3.621% os resultados da Entropia tradicional, incluindo Tráfego Real, em diferentes tamanhos de janelas e demonstra os ganhos da técnica proposta.

Palavras-chave: SDN, Entropia, Aprendizado de Máquina, DDoS, SVR, SVC, Random Forest.

Abstract

Software Defined Networking (SDN) brings a new concept in terms of data network architecture, in response to the limitations of traditional networks, but at the same time new security challenges are introduced for a programmable network with global vision. Many articles in the literature have used statistical techniques, Entropy and Machine Learning, alone to detect DDoS attacks in SDN. The main issue addressed by these papers is to establish a threshold that best separates normal traffic from spurious traffic, and the absence of a dynamic threshold that improves the efficiency of entropy-based algorithms. This work explores the use of ML for the dynamic calculation of algorithm thresholds using entropy and presents a hybrid method using entropy and ML, here called ML-Entropy, to differentiate it from traditional entropy and also three hybrid ML alternatives (hybrid SVR, hybrid SVC and hybrid Random Forest), thus improving its results in detection with reduced error rates compared to traditional entropy. As contributions are presented: A method that best distinguishes legitimate traffic from spurious traffic in ML-Entropy and three more viable alternative techniques; We use the Real environment to validate the experiment and the method; Use of real traffic instead of just synthetic traffic in the validation; Use of a technique in data collection that does not overload the controller and finally presenting a hybrid method that extracts the best of both techniques used, reducing the error rate and enabling a technique that allows a Dynamic Threshold. The results, in real and emulated environment, of the four techniques are all compared with entropy and among themselves and also with three works in the literature, where our results were superior in relation to the state of the art with hit rates above 99%, error rates below 1% and in addition ML-Entropy improved by up to 3.621% the results of traditional Entropy, including Real Traffic, in different window sizes and demonstrates the gains of the proposed technique.

Keywords: SDN, Entropy, Machine Learning, DDoS, SVR, SVC, Random Forest.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AV Antivírus.

DDoS *Distributed Denial of Service.*

DMZ *Demilitarized Zone.*

DoS *Denial of Service.*

DSR *Design Science Research.*

FN Falso Negativo.

FP Falso Positivo.

IC Intervalo de Confiança.

IDS *Intrusion Detection System.*

IETF *Internet Engineering Task Force.*

IoT *Internet of Things.*

IPS *Intrusion Prevention System.*

IRTF *Internet Research Task Force.*

KDD *Data Mining ou Knowledge Data Discovery.*

KNN *K Nearest Neighbors.*

LS-SVM *Least Squares Support Vector Machine.*

MC Matriz de Confusão.

ML *Machine Learning.*

MLP *Multi Layer Perceptron.*

MLR *Multinomial Logistic Regression.*

NS-3 *Network Simulator 3.*

ONRC *OpenFlow Network Research Center.*

OVS *Open Virtual Swicth.*

PC *Personal Computer.*

RBF *Radial Basis Function.*

SDN *Software-Defined Networking.*

SOM *Self-Organizing Maps.*

SV *Support Vector.*

SVC *Support Vector Classifier.*

SVM *Support Vector Machine.*

SVR *Support Vector Regression.*

UTM *Unified Thread Managemen.*

VoIP *Voice over IP.*

VPN *Virtual Private Network.*

Capítulo 1

Introdução

A Rede Definida por *Software* (do inglês - *Software-Defined Networking*, SDN) surgiu em resposta à necessidade de adaptações rápidas das redes tradicionais, abstraindo a rede em Plano de Dados e Plano de Controle, permitindo um controle centralizado e visão global da rede [9]. A sua utilização comercial já é uma realidade, implementada em muitas grandes empresas, mas traz alguns desafios relacionados com a segurança da informação [10]. Por exemplo, um ataque dirigido ao controlador pode parar toda a rede [11]. Os ataques DDoS têm crescido a taxas anuais elevadas [11], atingindo 1.3 Tbps, tal como reportado em [12], acometendo redes tradicionais e SDN. A atenuação deste tipo de ataque é necessária para manter a disponibilidade das redes. A utilização de técnicas estatísticas e *Machine Learning* ML para mitigar este problema é apresentada em [13–15]. De fato, diferenciar o tráfego legítimo do espúrio para lidar com ataques DDoS ainda é um grande desafio [16].

A entropia tem sido utilizada para detectar ataques DDoS com técnicas de ajuste do limiar como desvio padrão e outros [15]. Caraka et al [17] utilizaram algoritmo de regressão SVR onde o peso de cada *kernel* é optimizado durante o treino noutro contexto. Em [6–8], os autores propuseram um método de detecção híbrido no qual os resultados da Entropia são entregues aos algoritmos ML para detectar o ataque. Os trabalhos anteriormente referidos compartilham o mesmo objetivo, que é o de vencer o desafio do melhor limiar que separa o tráfego legítimo do espúrio, além da ausência de um limiar dinâmico que melhore a eficiência dos algoritmos baseados em entropia. No entanto, estão limitados ao que a entropia por si só oferece de ajuste ou simplesmente entregam ao ML este trabalho sem considerar o custo computacional que pode ter impacto no desempenho geral da rede.

Este trabalho explora a utilização de ML para o cálculo dinâmico dos limiares de algoritmos que utilizam entropia e apresenta um método híbrido utilizando entropia e ML, aqui denominado ML-Entropia, para diferenciá-lo da entropia tradicional e também três alternativas híbridas de ML (SVR híbrido, SVC híbrido e *Random Forest* híbrido).

ML-Entropia difere dos trabalhos anteriores pois utiliza o ML para encontrar o melhor limiar e aprimorar a detecção de ataque. A abordagem resultante reduz a taxa de erro e permite um limiar dinâmico, de acordo com o estado da rede. Foram realizadas experiências tanto em ambiente real como emulado, onde os resultados foram superiores que a literatura aqui comparada e mostram que a abordagem proposta neste trabalho atinge taxas de acerto de detecção superiores a 99%, taxas de erro inferiores a 1% e, além disso, o ML-Entropia melhorou em até 3.621% as taxas de erro em relação à Entropia tradicional. Este projeto de pesquisa se apoia na SDN para aplicação do experimento, mas a solução poderá ser aplicada também em Redes Tradicionais. Contudo ressalta-se que o diferencial deste trabalho é que a solução já estará pronta e habilitada para explorar todo o potencial existente nas Redes SDN, em termos de mitigação e implementação em qualquer de seus planos, ao contrário de soluções que só visam Redes Tradicionais ou só Redes SDN.

1.1 Motivação

O ambiente corporativo real que se beneficia da implementação da solução proposta é uma empresa que tem a competência para desapropriar, desocupar, recuperar, isolar, proteger e conservar áreas de preservação de mananciais utilizados ou reservados para fins de abastecimento à população da região, bem como para controlar as ações poluidoras de suas águas, inclusive além dos limites de sua concessão. A empresa possui atividades ininterruptas com funcionamento 24x7, pois seus serviços tem impacto direto sobre a manutenção da vida humana na região em que atua como em hospitais, presídios e do meio ambiente entre outros.

Nesse sentido a empresa tem investido em infraestrutura de rede a fim de garantir a disponibilidade das informações em tempo integral e no ano de 2017 adquiriu e implementou uma infraestrutura de virtualização de rede, também conhecida no meio acadêmico como SDN, aqui já definido, para atender a camada *CORE* (núcleo da rede onde estão localizados os principais servidores de rede, segurança e aplicações, além dos ativos de rede de núcleo) e Rede Desmilitarizada (do inglês - *Demilitarized Zone*, DMZ), buscando garantir a continuidade dos negócios através da geodistribuição dos dados e serviços com balanceamento de carga entre dois *data centers*, tornando a rede de dados corporativa híbrida com parte dela no modelo tradicional baseada em dispositivos físicos e outra parte virtualizada e integrada ao legado.

Este modelo de infraestrutura híbrida SDN é, em termos de tecnologia hoje disponível no mercado, inovador e promissor, devido a programabilidade da rede, contudo no meio acadêmico, muito ainda se discute a respeito de seu desempenho, segurança e

o impacto desses quando integrado a redes Múlti-inquilinos (do inglês - *Multi-Tenant*), como em [18–21]. Considerando a relevância que a infraestrutura tem na continuidade dos negócios e o alto impacto socioeconômico da sua indisponibilidade, garantir que essa infraestrutura SDN está adequada é importante. Portanto detectar e informar vulnerabilidades é fundamental para garantir a maior disponibilidade e segurança, direcionando, em tempo, ações de melhoria e mitigação de riscos para não comprometer o negócio e nem vidas humanas.

1.2 Justificativa

Alguns desafios e lacunas em SDN foram abordados e apresentados por [16], que realizaram uma extensa pesquisa do estado da arte em SDN, aqui apresentamos de forma sintética os de relevância para este trabalho, tais como: Implementar soluções que melhor diferenciem o tráfego legítimo do espúrio; Ausência de pesquisas que implementem em ambiente real as abordagens propostas; Uso de tráfego real ao invés de sintético nas pesquisas; Soluções de segurança que retirem a sobrecarga do Plano de Controle e Uso de técnicas eficientes que utilizem coletas de estatísticas fora do Plano de Controle retirando a carga do controlador.

Os Ataques do tipo DDoS têm crescido anualmente com taxas elevadas e representam um risco para todas as redes [11, 22, 23]. Instituições financeiras e outras corporações têm sido vítimas com percentuais significativos de perdas financeiras [24]. Os dispositivos tradicionais de segurança não foram capazes de impedir nem o maior ataque desse tipo registrado em fevereiro de 2018 na Akamai, um importante provedor de serviços na Internet, que hospeda o GitHub [12]. Vários trabalhos de pesquisa foram realizados para demonstrar as vulnerabilidades e prover um melhor sistema de detecção de ataques DDoS, como em [11, 13, 15], que usaram abordagens estatísticas ou que usaram técnicas de *Machine Learning* (ML) isoladamente para esse fim [14, 25, 26]. A questão em comum a estes trabalhos é a definição do melhor limiar e como encontrá-lo em técnicas como em Entropia ou mesmo em ML para melhor determinar a possibilidade de tráfego específico ser considerado espúrio ou não, que é um dos desafios a ser vencido, segundo [16]. Outra questão abordada por [8] é a necessidade, não apenas de um melhor limiar, mas também dele poder ser dinâmico e acompanhar as constantes mudanças que ocorrem na rede.

Este trabalho de pesquisa propõe um mecanismo híbrido que possibilite o uso de duas técnicas (Entropia e *Machine Learning*) de forma conjunta e eficiente, onde um melhor coeficiente de previsão do limiar utilizado no cálculo da entropia é encontrado pelo ML e devolvido à entropia, que denominaremos como "ML-Entropia", para diferenciá-la da entropia tradicional, aqui usada a Entropia de Shannon, melhorando assim seus resultados na detecção ML-Entropia em comparação com entropia tradicional. O trabalho não se

limita a esta técnica apenas e também apresenta mais três outras alternativas, todas na abordagem híbrida, usando algoritmos de ML, de Regressão e de Classificação (SVR Híbrido, SVC Híbrido e *Random Forest* Híbrido), recebendo os cálculos da entropia como entrada de dados, calculando o melhor limiar de separação dos dados de ataque e legítimo na detecção do ataque, seguida da detecção do ataque por eles, gerando seus resultados individuais para comparação entre as diversas técnicas do experimento. Esta abordagem também permitirá, em todas as técnicas, que o limiar seja dinâmico e ajustável ao estado atual da rede. Pretende-se ainda melhorar as métricas de desempenho na detecção de ataques do tipo DoS/DDoS volumétrico e reduzir as taxas de erros, em relação à entropia tradicional, evitando que serviços importantes da rede fiquem indisponíveis devido a este tipo de ataque. Reduzir as taxas de erro implica em evitar alarmar e bloquear falsamente um usuário legítimo ou ignorar e permitir um atacante real, nos dois casos a situação é prejudicial. Outro fator importante é que quanto maior a taxa de erro maior a chance dos administradores de rede passarem a ignorar os alarmes por descrédito no mecanismo de segurança, o que também representa um risco, pois o alarme ignorado pode ser verdadeiro.

As principais contribuições deste trabalho de pesquisa podem ser resumidas, de acordo com os desafios e lacunas de pesquisa discutidos em [16], da seguinte forma:

- Um método híbrido que pode distinguir melhor o tráfego legítimo do espúrio, com valor do limiar que é o coeficiente de previsão fornecido pelo ML para ML-Entropia, reduzindo a taxa de erro em comparação a Entropia de Shannon;
- Usar o ambiente Real para validar experimentos emulados (Mininet) aplicando o método proposto sob condições reais para validar o método;
- Uso do tráfego real ao invés de usar apenas o sintético na validação do método;
- Uso de uma técnica alternativa com os protocolos Netflow e IPFix na coleta de fluxos de dados, evitando o uso do Plano de Controle, que induz a um *overhead* ao controle centralizado, de acordo com [27];
- Um método com técnica de ML que permite que o Limiar seja Dinâmico e acompanhe o estado atual da rede;
- Um método que já é habilitado para explorar todo o potencial existente em SDN, mas que também é aplicável às redes tradicionais.

Os resultados em ambiente real e emulado são comparados com três trabalhos semelhantes na literatura e os resultados das alternativas de técnicas deste experimento entre si, onde temos a Entropia tradicional, ML-Entropia, SVR híbrido, SVC híbrido e *Random Forest* híbrido, todos os híbridos recebendo como dados de entrada os cálculos da Entro-

pia. As Métricas de Acurácia, Precisão, Taxa de Erro, *Recall e F1-Score* são utilizadas para aferir o desempenho e determinar a efetividade da técnica híbrida aqui proposta.

1.3 Objetivo Geral

Desenvolver um mecanismo de detecção de ataque DDoS em SDN, utilizando Entropia e *Machine Learning* que melhor separe o tráfego legítimo do espúrio, reduzindo a taxa de erro em comparação a Entropia de Shannon, com limiar dinâmico que possa ser ajustável ao estado atual da rede com uso de ML.

1.3.1 Objetivos Específicos

- Desenvolver um mecanismo híbrido de detecção de ataques DDoS em SDN que melhore o desempenho da detecção comparado à Entropia tradicional (de Shannon);
- Desenvolver um mecanismo híbrido de detecção de ataques DDoS em SDN que melhore o desempenho da detecção comparado ao Algoritmo de Regressão (SVR);
- Apresentar alternativas de técnicas dentro dessa abordagem que permita ao administrador da rede escolher a mais viável para sua realidade;
- Permitir que o mecanismo híbrido desenvolvido tenha um limiar que possa ser ajustável ao estado atual da rede;
- Implementar em ambiente sintético para comprovação de viabilidade da proposta;
- Implementar em ambiente real para validação e comprovação da efetividade do mecanismo desenvolvido;
- Comparar as técnicas híbridas do mecanismo híbrido proposto entre si e com Entropia tradicional, comparando também com soluções semelhante na literatura,;
- Utilizar técnica que permita a coleta do fluxo de dados e análise para detecção do ataque que não sobrecarregue o Plano de Controle;
- Apresentar um método que já é habilitado para explorar todo o potencial existente em SDN, mas que também é aplicável às Redes tradicionais.

1.4 Estrutura do Documento

Este documento está estruturado da seguinte forma:

- **Capítulo 2:** A fundamentação teórica é apresentada para contextualizar o leitor sobre os principais conceitos aqui abordados e também para melhor entendimento quanto as principais escolhas feitas na condução do experimento;
- **Capítulo 3:** Uma revisão da literatura é realizada apontando os desafios e abordagens em SDN, enfatizando entropia, *Machine Learning* e as diversas técnicas utilizadas para encontrar o melhor limiar;
- **Capítulo 4:** A proposta de trabalho é detalhada, bem como os caminhos a serem seguidos para atingir os objetivos;
- **Capítulo 5:** A metodologia adotada neste trabalho de pesquisa segue o *Design Science Research (DSR)* e suas etapas são definidas neste capítulo, além da metodologia na condução do experimento, para melhor avaliação de desempenho da solução desenvolvida, detalhando e justificando as definições;
- **Capítulo 6:** Os ambientes emulado e real do experimento são detalhados e os resultados obtidos são apresentados, comentados, analisados e comparados com uma conclusão a respeito da viabilidade da proposta;
- **Capítulo 7:** São feitas as considerações finais dos objetivos alcançados com base nos resultados obtidos e onde este projeto pode evoluir contribuindo com propostas de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Os principais conceitos deste trabalho foram abordados neste capítulo para um melhor entendimento da proposta e soluções adotadas na implementação do mecanismo de segurança. Na Seção 2.1 foi feita uma explanação do que é uma SDN e como ela se diferencia de uma rede tradicional, seu funcionamento e protocolos, depois na Seção 2.2 as ferramentas de emulação e simulação de redes SDN foram apresentadas e como elas possibilitaram a montagem do ambiente de teste para produção de um Modelo com os resultados sintéticos, importantes para testar sua viabilidade antes de seguir para uma abordagem de aplicação da solução em ambiente real. Na sequência, a Seção 2.3 aborda os principais mecanismos de segurança e porque eles não eliminam a necessidade de mecanismos de segurança complementares como este aqui proposto no trabalho, seguindo com a Seção 2.4 que explica por que os ataques DoS/DDoS são relevantes em termos de risco a serem mitigados em redes SDN ou Tradicional, além de uma explanação de seu funcionamento e classificação, posteriormente a Seção 2.5 trata a respeito da teoria da entropia e como ela é usada na detecção de ataques, e por fim a Seção 2.6 explica o *Machine Learning*, falando brevemente dos principais algorítimos e técnicas, sua aplicabilidade e como ele foi útil nessa abordagem de pesquisa devido a suas características de aprendizado.

2.1 Redes SDN

Conforme ilustrado na Figura 2.1 nas redes tradicionais a tomada de decisão (controle) e o encaminhamento dos dados estão localizados nos dispositivos físicos de rede e fortemente acoplados um ao outro, onde esta característica somada à baixa capacidade de programação dos dispositivos limita o suporte à disponibilização de novas funcionalidades [28]. A Rede Definida por *Software* (SDN) é a virtualização da rede e surgiu em resposta às limitações da rede tradicional, sendo ainda muito discutida na academia e na indústria [18]. A literatura nos têm apresentado como grandes vantagens da SDN a simplificação de imple-

mentação, a flexibilidade de adaptação às constantes mudanças requeridas pelo negócio, a programabilidade da rede e a visão global da rede e seus aplicativos em execução [19]. Ela desacopla o controle do dispositivo, abstraindo a rede em camadas definidas como Plano de Dados, Plano de Controle e Plano de Aplicação.

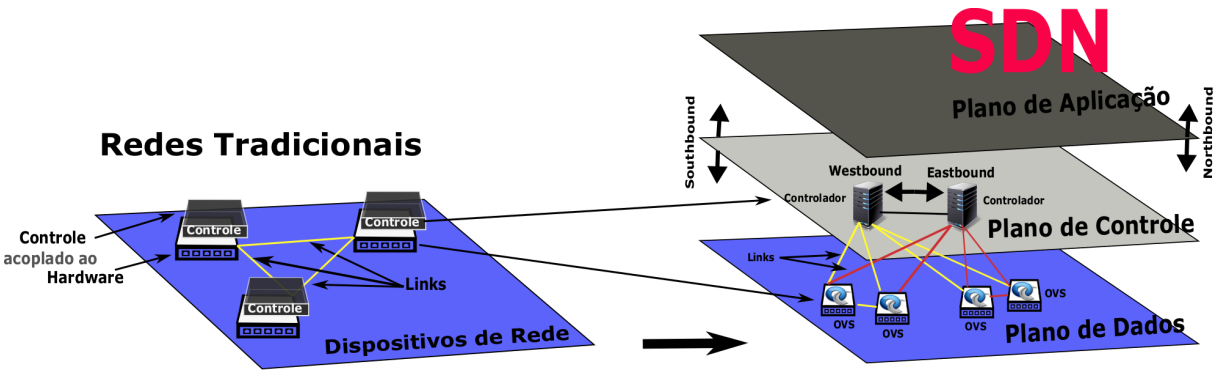

Figura 2.1: Rede Tradicional versus Rede SDN.

Definida como uma arquitetura de rede que possui quatro pilares por [28] da seguinte forma:

1. Os planos de controle e dados são desacoplados, o que resulta em dispositivos de rede se tornando elementos de simples encaminhamento de pacotes, regidos por um controlador;

2. As decisões de encaminhamento são baseadas em fluxo. No contexto SDN-OpenFlow, um fluxo é uma sequência de pacotes entre uma origem e um destino, onde todos recebem políticas de serviço idênticas nos dispositivos de encaminhamento;

3. A lógica de controle é gerenciada por um controlador.

4. A rede é programável por meio de aplicativos de software executados sobre o controlador que interage com os dispositivos do plano de dados subjacentes.

Na abstração do paradigma SDN, conforme Figura 2.2, os serviços e aplicações estão no Plano de Aplicação (plano mais alto), enquanto a inteligência ou o controle é desacoplado do *hardware* e está localizado no Plano de Controle (plano intermediário), comparado a um sistema operacional, com controle centralizado, o que pode representar um ponto único de falha, com controlador único ou múltiplos controladores distribuídos, necessitando neste caso de mecanismo de sincronismo entre eles para convergência do conhecimento único do estado geral da rede, disponibilizando ainda uma interface programável aberta que permite a implementação de tarefas de gerenciamento e oferece novas funcionalidades [19].

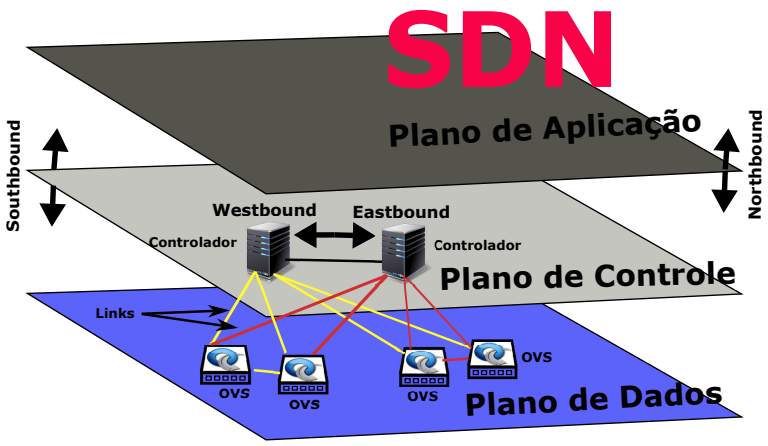

Figura 2.2: Infraestrutura SDN e Comunicação das Interfaces *Northbound e Southbound*.

O Plano de Controle é a camada em que todos os cálculos são feitos e as decisões são tomadas. O controlador desempenhará o papel mais importante na rede, responsável por coletar e gerenciar todas as informações de status da rede, o que oferece a possibilidade de gerenciamento automático da rede [19]. Os usuários podem modificar o protocolo interno do roteador programando para obter melhor desempenho de troca de dados e aprimorar a flexibilidade para suportar o rápido crescimento das necessidades de negócios da rede [10]. Após a tomada de decisão ele informa os elementos de encaminhamento do Plano de Dados na camada abaixo, atualizando suas tabelas de fluxo [19]. Isso é feito usando a interface segura de comunicação sul chamada de *SouthBound* que fica entre a camada de controle e de dados, usando um protocolo como o *OpenFlow* [29], que implementa a lógica de controle que será traduzida em comandos a serem instalados no plano de dados, através da tabela de fluxos, armazenada na memória do *switch*, ditando o comportamento dos dispositivos de encaminhamento [28]. Existe outra interface de comunicação norte entre a camada de Controle e a camada de Aplicação que é chamada de *NorthBound*, também é definida na SDN uma interface de comunicação de sentido leste e de sentido oeste para prover a troca de informações entre os controladores, denominadas *WestBound* e *EastBound* [29].

O Plano de Dados (plano mais baixo) é composto pelos dispositivos de encaminhamento que são responsáveis pelo encaminhamento de fluxo dos dados, tais como *Switches, Open Virtual Switches (OVS)*, Roteadores e *Wireless Acess Point* [19]. Eles não tomam nenhuma decisão e estão inteiramente dependentes das regras de fluxos que são definidas no plano de controle e inseridas em suas tabelas de fluxos, que são armazenadas em memória, com tempo de validade definidos por regra [28]. Sempre que um caminho desconhecido aparecer uma nova regra de encaminhamento será requisitada ao Controlador e adicionada à tabela de fluxo [28]. O protocolo *OpenFlow* é o responsável por toda a comunicação no plano de dados e deste com o plano de controle através da interface *southbound*, ele foi padronizado em 31 de dezembro de 2009 com a publicação de suas especificações pelo

Open Networking Foundation (ONF), uma organização dedicada à promoção e adoção de Redes Definidas por Software (SDN) [29]. Ele tem evoluído ao longo dos anos desde sua primeira versão até os dias atuais com a contribuição da industria e da academia que muito tem investido em sua pesquisa e desenvolvimento, conforme *Roadmap* da ONF [1] na Figura 2.3, que traz uma breve descrição de suas versões, listando suas principais características [1]. Do lado acadêmico, o *OpenFlow Network Research Center* (ONRC) foi criado com foco na pesquisa em SDN. Também houve esforços de padronização da SDN na IETF e IRTF e em outras organizações produtoras de padrões [19].

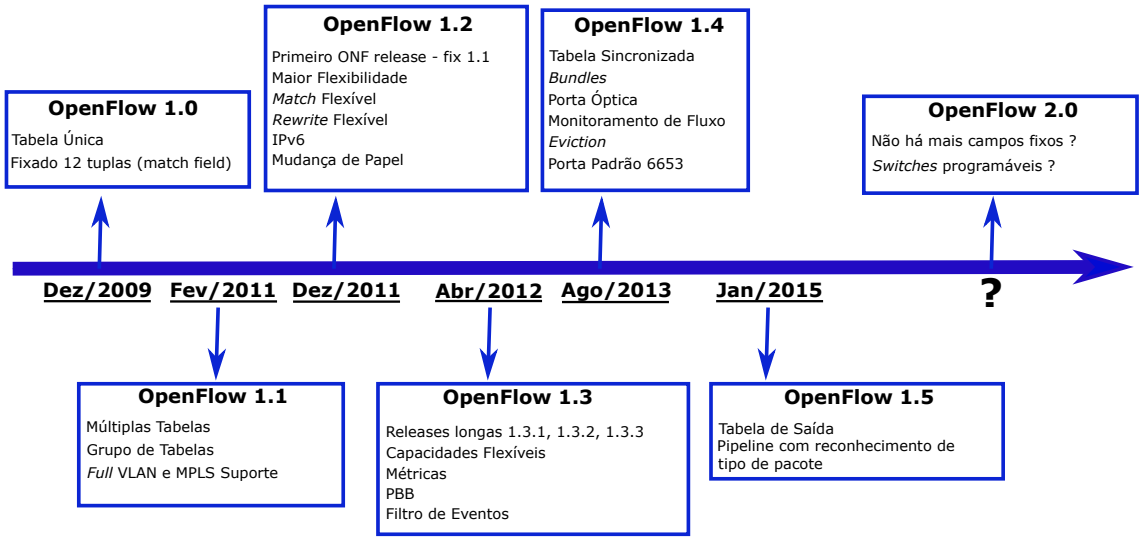

Figura 2.3: Linha do tempo das versões *OpenFlow*, baseado em [1].

O sucesso da SDN não está restrito ao campo de pesquisa e academia, mas também se aplica com sucesso no mundo real em empresas como Google que implementou a rede B4 e alcançou uma utilização de rede sem precedentes de 95%, bem como a Microsoft que implementou o controlador *OpenDaylight* para aprimorar a experiência de vídeo dos usuários do Skype [10]. Também a rede de lojas GAP e a Viptela colaboraram para aplicar a tecnologia SD-WAN em 1.350 lojas nos EUA [10]. De acordo com os dados divulgados pela *Research and Markets* na Irlanda, o volume de negócios da SDN crescerá a uma taxa anual de 42,3% nos próximos seis anos [10]. Até 2023, espera-se que os lucros da SDN no mercado global de operadoras atinjam US $ 9,5 bilhões, sendo a SDN denominada pelo MIT como “uma das dez principais tecnologias inovadoras para mudar o mundo“ [10].

A industria resolveu unir forças com o ONF e no dia 12 de março de 2018, foi anunciada a formação do projeto ONF Stratum com o objetivo criar a próxima geração da SDN (NG-SDN) baseada na tecnologia utilizada pelo Google em seus próprios ambientes SDN [30]. Fazem parte do projeto ONF Stratum: Google, Tencent, China Unicom, NTT, Turk Telekom, Big Switch Networks, VMware, Broadcom, Cavium, Mellanox e Xilinx

[30]. Este Projeto promete levar a SDN para um próximo nível, com funcionalidade de controle de *pipeline* habilitada por meio do uso da linguagem P4 que ajuda a definir um *pipeline* de plano de dados, permitindo programar dinamicamente as tabelas de *pipelines* de encaminhamento em um *switch* [30].

2.2 Ferramentas de Simulação e Emulação

É de fundamental importância que antes de se implementar uma solução em ambiente real seja feito um teste de prova ou prototipação da solução para não comprometer o ambiente de produção, o que pode resultar em perdas financeiras. Algumas ferramentas de simulação e emulação em SDN estão disponíveis para apoiar nesse sentido, sendo preferível um emulador por se aproximar mais do ambiente real que o simulador, são elas:

- EstiNet: É um software próprietário que requer licença de uso, trata-se de um simulador e emulador de redes SDN que permite a utilização de controladores externos com uso do *OpenFlow* nas versões 1.0 e 1.1 [31]. Segundo Wang et al. ele é um emulador confiável mesmo utilizando grandes cargas de dados e comparado com o Mininet funcionou bem no geral, no entanto, demostrou algumas inconsistências dependendo do cenário do teste [32].
- NS-3: É um software de código aberto, escrito em C++, direcionado principalmente para pesquisa e uso educacional. Está limitado ao *OpenFlow* 0.8.9, vale ressaltar que o *OpenFlow* está na versão 1.5 [33].
- Mininet: É um software de código aberto desenvolvido por pesquisadores da Universidade de Stanford, com o objetivo de auxiliar no desenvolvimento de redes SDN. Trata-se de um emulador capaz de criar *hosts, switches* tradicionais e *Open Virtual Swicth* (OVS), controladores, *links* e roteadores. Com Mininet é possível criar redes com grande quantidade de nós por linha de comando ou interface gráfica, como também é possível conectar com outras redes e outros computadores, permitindo o uso do *OpenFlow* nas versões 1.0 até 1.3 [34]. O código desenvolvido no Mininet pode ser movido para uma rede de produção real [11].

2.3 Dispositivos de Segurança

As corporações investem em dispositivos de segurança da informação, para suas redes traicionais, dos mais variados tipos como *Firewall*, Sistemas de Detecção de Intrusão (do inglês - *Intrusion Detection System*, IDS), Sistemas de Prevenção de Intrusão (do inglês - *Intrusion Prevention System*, IPS) e outros, pois cada um tem seu propósito

específico e é esse conjunto de especialidades em perímetros bem definidos na rede que vão mitigar os ríscos. Na tentativa de um gerenciamento reduzido, menores custos de suporte e complexidade de uso, surgiram os Gerenciadores Unificados de Ameaças (do inglês - *Unified Thread Management*, UTM), que vieram com a promessa de substituir vários mecanismos de uma só vez por um único *appliance* [35], contudo se observou com esses dispositivos, já consolidados no mercado, que mesmo assim, isso não descartou a necessidade de mecanismos especializados. Vejamos as razões.

Os sistemas IDS geralmente usam métodos baseados em assinaturas ou em anomalias para detectar ataques, pacotes e tráfego maliciosos [36]. A diferença básica entre um IDS e um IPS é que o primeiro apenas alerta e o segundo executa ações de contramedidas predefinidas. Baseados em assinaturas são eficientes, mas estão limitados aos modelos já conhecidos em sua base, sendo ineficazes aos desconhecidos. Os baseados em anomalias buscam e classificam anomalias no tráfego da rede normal [36], contudo eles apresentam uma alta taxa de falsos alarmes [35].

Os *Firewalls* são muito eficazes no controle de fronteiras de dentro para fora e vice-versa, contudo são vulneráveis a *backdoors* de aplicações. Os UTMs que são caixas ou *Appliances* que tentam exercer a função de tudo em um, como *Firewall*, IDS/IPS, anti-spam de *gateway*, antivírus de *gateway* (AV), filtro de conteúdo, segurança de camada de aplicação, balanceamento de carga, prevenção de vazamento de dados, segurança VPN e VoIP [35]. Cada módulo de função pode ser ativado independente de outro, contudo quando vários ou todos os dispositivos estão ativos problemas de manipulações de processamento de pacotes repetidas em diferentes pontos de verificação podem ocorrer, resultando em problemas redundantes de classificação de pacotes [35].

Cada dispositivo tem sua vantagem e desvantagem, sendo que o uso de um não descarta a necessidade do outro, pois o conjunto de soluções de segurança implementados em uma infraestrutura melhora o nível de segurança, apoiando não apenas em termos de especialidades, mas também de abrangência perimetral. Outro ponto é que são todos concebidos para redes tradicionais e com controle acoplado ao *hardware*, requerendo ajustes para uso em SDN e com perda da capacidade de programabilidade das redes SDN. Por fim na SDN, um invasor pode ignorar os dispositivos de segurança, resultando na falha das medidas de segurança pré-implantação [10].

2.4 Ataques de Negação de Serviço

Ataques de negação de serviço (do inglês - *Denial of Service*, DoS) são hoje o maior desafio em segurança de redes, pois com pouco esforço é possível alcançar mais de 5000 ataques com o tipo de ataque de negação de serviço distribuído (do inglês - *Distributed Denial*

of Service, DDoS) [13]. O ataque DoS/DDoS é caracterizado pela tentativa maliciosa de esgotar os recursos de um computador ou de uma rede enviando tráfego pesado para o alvo, exaurindo os recursos do dispositivo ou do link [11]. Segundo Mansfield, 2014 [24], que mediu a incidência de ataques de negação de serviço, com fluxos à época registrados entre 100 a 300 Gbps, distribuindo suas taxas de incidência na industria segundo a atividade da seguinte forma: Financeira 37%, Varejo 24%, Serviços 20%, Manufatura 20%. O autor finaliza considerando que esse número chegaria a ordem de Terabits em pouco tempo.

Os ataques DDoS podem ser organizados em dois tipos principais: volumétricos e não volumétricos [37]. Os ataques volumétricos, também conhecidos como ataques de saturação de recurso, geram tráfego excessivo em bits por segundo (bps) até exaurir um recurso de rede, utilizando técnicas de amplificação e reflexão para lançar o ataque, como exemplo temos *UDP/TCP floods, ICMP flood*, etc [22]. Os ataques não volumétricos, também conhecidos como ataques de baixo volume e lentos, exploram características específicas do protocolo ou detalhes de implementação, como uma *string* incompleta, segurando a conexão com a vítima por tempo indefinido, que por sua vez deixa de responder usuários legítimos, tal como *Slow* HTTP *(HyperText Transfer Protocol) headers* [37]. Quanto aos recursos explorados, segundo Mirkovic e Reiher [38] os ataques podem ser caracterizados como de:

- Aplicação: Uma aplicação é direcionada a um *host* para negar uso legítimo;
- *Host*: Torna um *host* inacessível;
- Recurso: Sobrecarrega um servidor para mantê-lo vinculado ao fluxo contínuo de solicitações falsas;
- Rede: Ataque volumétrico de tráfego a uma rede até esgotar o link;
- Infraestrutura: Tem como alvo o servidor de DNS, como exemplo.

Comparando o primeiro trimestre de 2013 ao último trimestre de 2012 o índice de ataque DDoS cresceu 718%, sendo que nem todos os ataques podem ser detectados ou documentados [11], mais de 100% de aumento foi observado em 2017 em relação a 2016 [22] e em 2020 570% em relação a 2019 [23]. Às 17h28 GMT, 28 de fevereiro de 2018, a Akamai sofreu um ataque DDoS de 1.3 Tbps contra um de seus clientes, o GitHub, conduzido por uma técnica chamada *Memcached Reflection* baseado em tráfego UDP, este foi um ataque de reflexão amplificada cuja descoberta havia sido anunciada pouco tempo antes pela própria Akamai e outras empresas de segurança de rede [12].

O ataque utilizado neste projeto é volumétrico, dessa forma seguiremos descrevendo um ataque desse tipo. O ataque de negação de serviço distribuído (DDoS) é o DoS com dois ou mais envolvidos no ataque de forma coordenada a um mesmo alvo. Segundo

Alomari et al. 2012 [39] no DDoS o “Atacante“ implanta o código malicioso, denominado de “*Botnet*“ usados principalmente para ataque à camada de aplicação, no(s) PC(s), que são denominados de “Zumbis“, após transformá-los em Zumbis, no momento do ataque o *Botnet* é executado e um fluxo de tráfego, partindo do(s) Zumbi(s), é direcionado para a vítima (Alvo), potencializando o ataque. Em um método mais sofisticado o criminoso pode usar uma camada intermediária de PCs, aqui denominados de “Manipuladores“, para orquestrar o ataque, onde cada Manipulador possui um número específico de Zumbis sob seu comando, reduzindo o esforço anterior. Este método é um dos métodos mais comuns usados na sobrecarga e interrupção de serviços de rede.

O DDoS pode assumir uma arquitetura complexa, conforme Figura 2.4, esta arquitetura contém todos os elementos descritos anteriormente para o tipo volumétrico de ataques de negação de serviços já caracterizados, contudo uma evolução deste tipo de ataque transformou a vítima Alvo em “Alvo Espelho“ para um ataque maior ainda, utilizando uma solicitação de serviço a um servidor legítimo (“Alvo Espelho“), que representa um pedido de tamanho pequeno, mas alterando o IP de origem para outro servidor legítimo, a segunda vítima (“Alvo da Reflexão Amplificada“), que receberá as respostas que possuem tamanhos maiores em comparação aos pedidos feitos, ou seja, um menor esforço com maior impacto, que é denominado de ataque DDoS de Reflexão Amplificada [37].

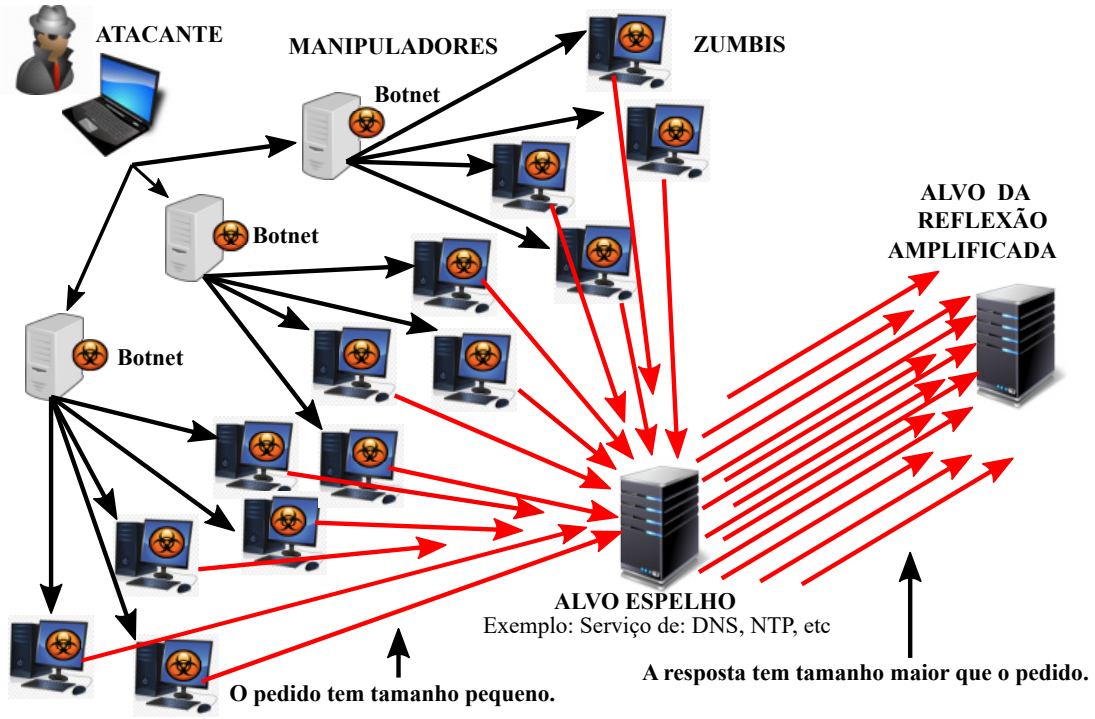

Figura 2.4: Ilustração da Arquitetura do Ataque DDoS de Reflexão Amplificada.

Os ataques de DoS/DDoS são o principal modo de ataque para reduzir a disponibilidade de uma rede [10]. Diversas abordagens tem sido utilizadas para mitigar ataques

DDoS como métodos de entropia, aprendizado de máquinas ou desenvolvimento de aplicações que rodam no plano de aplicação para este fim, este assunto será detalhado no Capítulo 3 na Seção 3.1 e para dar continuidade aos fundamentos teóricos abordar-se-á apenas os conceitos de entropia e aprendizado de máquinas para um melhor entendimento das duas técnicas em separado.

2.5 Entropia

O conceito de entropia tem origem na mecânica estatística e foi transformado em uma medida de quantidade de informação no desenvolvimento da teoria da informação de Claude Shannon [40]. Em síntese, Shannon inicia a Teoria Matemática da Comunicação afirmando que a informação de uma mensagem pode ser mensurada pela quantidade do que ele chamou de "entropia", e que é relacionada com a frequência dos símbolos transmitidos (os símbolos fazem parte de um repertório finito, que define a forma de codificação das mensagens). Ele adverte que a quantidade de informação de uma mensagem é independente de seu significado. Como consequência da abordagem estatística adotada, demonstra que a entropia zero é obtida quando existe a certeza da transmissão de um único símbolo e, no caso oposto, a entropia máxima é obtida quando a frequência dos símbolos é equiprovável [40]. Em outras palavras Shannon observa que a entropia máxima ocorre quando as possibilidades são equiprováveis, assim, quanto maior for a diferença entre as probabilidades, menor será a entropia, e esta é a situação de maior incerteza.

Na fórmula extrai-se o logaritmo de uma probabilidade (que por definição, é um número fracionário entre 0 e 1). Sabe-se que o logaritmo de um número fracionário é negativo. O sinal negativo na fórmula de Shannon é decorrência da demonstração algébrica do teorema da entropia e que serve para tornar seu valor positivo [40]. Shannon definiu o cálculo da Entropia na Equação 2.1, onde H é a entropia, x é um evento no conjunto, N é o número de elementos, P_i é a probabilidade da ocorrência de i.

$$H(x) = -\sum_{i=0}^{N-1} P_i \log_2 P_i \tag{2.1}$$

Entropia tem sido utilizada na detecção de ataques em SDN, demonstrando ser uma boa abordagem que é leve computacionalmente por [11, 13, 15, 41], apenas para citar alguns trabalhos que aplicaram o cálculo de entropia nos dados estatísticos gerados pela rede SDN na busca por tráfego espúrio. Os resultados tem se demonstrado viáveis e promissores [16]. Em condições de ataque DDoS, o comportamento esperado dos dados será distribuído para endereços de origem falsificados e concentrado para endereços de destino, onde a entropia captura este comportamento distribuído e concentrado [42].

Na prática, para entendimento da aplicação de entropia na detecção de ataques, como exemplo, imagine que, em uma comparação par a par, os atributos Entropia Média do IP de Origem ($\overline{E}\_IPOrig$) e Entropia Mínima do IP de Destino ($E_{Min}IPDest$) são selecionados para definir os limiares de comparação (Λ) em um peso (T_1) para $\overline{E}\_IPOrig$ e um peso (T_2) para $E_{Min}IPDest$, onde T_1 e T_2, no exemplo, podem ser duas vezes o desvio padrão da Entropia do IP de Origem/Destino, respectivamente. Assim sendo temos o seguinte exemplo: $\Lambda =$ **Se** $(\overline{E}\_IPOrig < T_1)$ **&** $(E_{Min}IPDest > T_2)$ **Então** *Ataque* **Senão** $Normal$. O desafio é justamente definir um limiar com um peso que melhor separe o tráfego espúrio do legítimo na comparação.

2.6 Machine Learning

O *Machine Learning* (ML) é parte do *Data Mining ou Knowledge Data Discovery* (KDD), proposto por Faayad [2] segundo ele o valor de armazenar volumes de dados dependente da nossa capacidade de extrair relatórios úteis, identificando eventos e tendências interessantes, apoiando decisões e políticas com base em análise estatística, inferência e exploração dos dados para atingir metas comerciais, operacionais ou científicas. O processo é descrito na Figura 2.5, que é uma coleção de técnicas que buscam conhecimento através de padrões estruturais nos dados, com capacidade de aprender automaticamente com a experiência e melhorar sua base de conhecimentos [43]. Em 1983, Herbert Alexander Simon observou que aprender denota mudanças no sistema que são adaptativas no sentido de que elas permitem que o sistema faça a mesma tarefa ou tarefas extraídas da mesma população de forma mais eficiente e eficaz na próxima vez [44].

As técnicas de KDD estão divididas em diversos métodos como Classificação, Modelos de Relacionamento entre Variáveis, Analise de Agrupamento, Sumarização, Modelo de Dependência, Regras de Associação e Analise de Séries Temporais, conforme definido por [2]. Segundo Ribeiro et al. 2013 [45] classificação é o aprendizado de uma função a ser usada para mapear dados em uma de várias classes discretas definidas previamente. A análise discriminante permite que dois ou mais grupos possam ser comparados com o objetivo de determinar se diferem uns dos outros e, também, a natureza da diferença, de forma que, com base em um conjunto de variáveis independentes, seja possível classificar indivíduos ou objetos em duas ou mais categorias mutuamente exclusivas.

Ainda segundo os autores [45] dentre os métodos de classificação podemos citar:

- **Distância mínima euclidiana:** para a definição de classes por proximidade no gráfico com o protótipo;
- **Vizinhos mais próximos:** Esse algoritmo de classificação é baseado na técnica do vizinho mais próximo para reconhecer padrões. Aqui o conjunto de dados mais comum

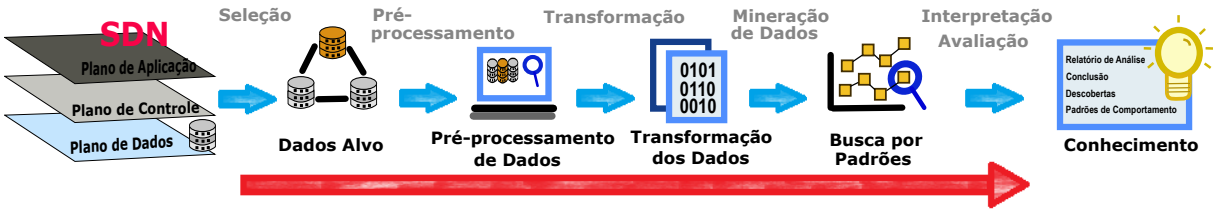

Figura 2.5: Ilustração dos passos do processo de *Machine Learning* aplicado na SDN, de acordo com o KDD proposto por Fayyad. [2]

é mantido como modelo a ser comparado com novos dados, comparando se uma instância de classe desconhecida está a uma distância pequena de uma classe conhecida, então as classes devem ser as mesmas. Nesse método, não são criados protótipos ou assinaturas, usando-se as próprias instâncias.

- **Agrupamento ou Clusterização:** Busca agrupar os dados com características semelhantes em um mesmo grupo e deve garantir que tenham características diferentes entre si e entre os grupos.
- **Sumarização:** Descrição compacta do que caracteriza um conjunto de informações. As medidas de posição e variabilidade são exemplos simples de sumarização.
- **Séries temporais:** São uma forma de organizar no tempo as informações quantitativas e podem apresentar tendência crescente, decrescente ou estacionária, e até tendências diferentes em trechos sequenciais, este segundo [46].
- **Detecção de desvios ou *Outliers*:** Busca a detecção de desvios discrepantes do padrão ou do comportamento esperado, também chamado de detecção de anomalias, detecção de ruído ou ainda mineração de exceções.
- **Associação:** Tem o intuito de identificar elementos que ocorrem em comum dentro de uma base de dados que, de alguma maneira, estão ou devem estar relacionados, seguindo uma lógica básica de associação que é encontrar elementos que implicam na presença de outro(s) numa mesma transação. Como exemplos de algoritmos que implementam associação temos os padrões sequenciais e as regras de associação.
- **Padrões sequenciais:** Procura por elementos ou eventos que ocorrem em uma sequencia através do tempo. Por exemplo, uma loja pode descobrir que consumidores que compram o produto “A“ tendem também a levar o produto “B“ em 70% dos casos. Isto pode indicar a necessidade de uma mudança na disposição dos produtos na loja e assim facilitar aos consumidores que podem ser alvo de promoções de venda do produto “B“, próximo do produto “A“, influenciando num aumento do consumo do produto “B“.
- **Regras de associação:** Busca agrupar os dados que apresentam ocorrência entre si, onde a derivação dessas correlações permite subsidiar a tomada de decisões. Exemplos: Se {A} –> {B} (Se ocorrer A então B) ou Se {A,B} –> {C} (Se ocorrer A e B então C).

- **Redes neurais:** É uma técnica de construção de um modelo matemático, originalmente concebida com base no estudo do cérebro humano. Tem capacidade para aprendizado, generalização, associação e abstração.
- **Regressão ou Predição:** Aprendizado de uma função usada para mapear os valores associados aos dados, baseado no método dos mínimos quadrados ordinários com propriedades estatísticas relevantes e apropriadas. A regressão pode ser linear e não linear.
- **Árvores de decisão:** É a representação simples do conhecimento em estrutura de árvore, particionando recursivamente um conjunto de treinamento até que um subconjunto contenha casos de uma única classe, tendo este resultado final da árvore obtida de maneira compacta para reutilização em novos casos. Na árvore de decisão os nodos representam os atributos, os arcos indicam os valores dos atributos e as folhas que designam a classificação. A árvore pode ser lida a partir do teste encontrado, normalmente chamado nó raiz da árvore. Como exemplo de algoritmos que implementam a árvore de decisão, tem-se o ID3, o C4.5 e o C5.0. O *Random Forest* é um dos algoritmos de classificação desse tipo mais utilizados e serviu neste experimento como uma alternativa de técnica híbrida que é comparada ao final.

Nos algoritmos de ML, diferentemente do método tradicional, onde um conjunto de regras é introduzido para gerar uma resposta a partir do processamento dos dados introduzidos, no ML as regras são criadas a partir dos dados que serão analisados e as respostas (ou resultados) que são esperados a partir dessa análise que constitui o Modelo [3]. O aprendizado pode ser **Supervisionado**, quando o modelo tem uma referência ao que está certo e ao que está errado, ou **Não Supervisionado**, quando não há resultados predefinidos para o modelo usar como referência para o aprendizado [3]. Há dois tipos principais de problemas de ML, chamados de Classificação e Regressão. O problema será de **Classificação** quando o objetivo for prever uma determinada classe, que é uma escolha a partir de uma lista de possibilidades predefinidas. O problema será de **Regressão**, quando o objetivo for prever um número contínuo ou um número de ponto flutuante em termos de programação [3]. Originado do *Support Vector Machine* (SVM) e baseado no método de classificação temos o *Support Vector Classifier* (SVC) [3] que servirá para receber os cálculos de entropia, também na abordagem híbrida proposta e realizar a detecção do ataque na comparação dos resultados.

A utilização de ML para o cálculo dinâmico dos limiares de algoritmos que utilizam entropia, explorando a capacidade dos algoritmos de regressão de retornar um coeficiente, como já mencionado aqui, é viável para solucionar o problema da ausência de um limiar dinâmico que melhore a eficiência dos algoritmos baseados em entropia. O *Support Vector Regression* (SVR) é um tipo de regressão, linear e não linear, também originado do *Support Vector Machine* (SVM) e é baseado no cálculo de uma regressão linear em

um espaço de recurso multidimensional que pode ser caracterizado por uma função de *kernel* como *Radial Basis Function* (RBF) [47]. Ele serviu aqui como técnica híbrida baseada em regressão e também para fornecer o coeficiente de previsão à ML-Entropia, assim sendo dedicamos um pouco mais de explicação sobre seu funcionamento para melhor entendimento da proposta.

Ko e Lee [48] afirmam que o SVR cria uma aproximação com uma função desconhecida, mapeando os dados de entrada em um espaço multidimensional através do mapeamento não linear da função e, em seguida, o problema linear é construído de acordo com suas características no espaço. Esta característica é importante neste trabalho na busca de uma boa separação do tráfego espúrio do legítimo, uma vez que a função busca a linearidade respeitando as características do espaço multidimensional. Segundo Filgueiras [4], a separação máxima entre as classes é denominada margem e as amostras que limitam a separação entre as classes são denominadas vetores de suporte. Essa propriedade é explorada no trabalho, além disso o SVR requer poucos parâmetros definidos pelo usuário para se ajustar ao *kernel* do modelo. A seleção desses parâmetros determina a complexidade da previsão [49]. O SVR pode ser usado com uma função de *kernel* como RBF, também conhecido como LS-SVM *(Least Squares Support Vector Machine)*, para atingir um limite mais apropriado na implementação da entropia, aumentando a eficiência da detecção de ataques e seu principal benefício é ser computacionalmente mais eficiente que o método SVM comum [50].

Segundo Caraka et al. [17], que utilizaram SVR noutro contexto, uma boa separação dos dados é obtida pelo SVR por um hiperplano. O SV *(Support Vector)* determina automaticamente o centro, peso e limite, afirmando ainda que o método SVR obtém a função de regressão de acordo com a Equação 2.2. Centro, peso e limite, ou seja limiar, este é exatamente o objetivo deste trabalho encontrar o melhor limiar que possa definir o limite nos dados separando o tráfego espúrio do legítimo na detecção do ataque, com isso pode-se ter uma melhor precisão e uma consequente redução dos Falsos Negativos (FN) e Falsos Positivos (FP). A redução das taxas de erro evita causar o bloqueio de usuários indevidamente, impactando em suas rotinas de trabalho e no negócio, bem como uma situação ainda pior que é o FN que permite aos atacantes continuarem a ter acesso a rede na tentativa de atingir seus objetivos espúrios. Na Equação 2.2 segundo Filgueiras, 2014 [4], ω é um vetor de pesos, que multiplicado pelo vetor amostral $\phi(x)$ e somado à constante b define o hiperplano que separa as classes e permite identificar a classe ao qual a amostra pertence.

$$f(x) = \omega^T \phi(x) + b \tag{2.2}$$

Ainda de acordo com ele [4], a máxima separação entre as classes é denominada margem e as amostras que limitam a separação entre as classes são chamadas de vetores de

suporte conforme exibido na Figura 2.6, os vetores de suporte são as amostras nos quais $\omega^T\phi$(x)+b=+1 para amostras pertencentes à classe +1 e $\omega^T\phi$(x)+b=-1 para amostras pertencentes à classe -1. Devido a não possibilidade de sempre separar duas classes de amostras por um hiperplano, para contornar esse problema, Corte e Vapnik [5] introduziram o conceito de variável de folga ξ_i, no qual admitem que algumas amostras possam ter um erro associado à sua classificação. Este erro é proporcional a distância da amostra ao hiperplano definido pelos vetores de suporte da classe à qual pertencem.

De acordo com a Equação 2.3 a margem para esta nova situação pode assumir um valor muito alto, assim, os erros de classificação são ponderados por uma *(Support Vector Machine)* (SVM) constante C adicionada ao problema de otimização [4]. O *(Support Vector Regression)* (SVR) necessita de poucos parâmetros definidos pelo usuário para o ajustar o *kernel* do modelo. Necessita do valor da constante C e da quantidade de erros e em uma área ξ_i, conforme a Equação 2.4. A seleção desses parâmetros ditam a complexidade da predição [49].

$$minimize : \frac{1}{2}||\omega^T||^2 + C\sum_{i=1}^{N}\xi_i \tag{2.3}$$

sujeito a:

$$\begin{cases} \omega^T\phi(x) + b \geq +1 - \xi_i \\ \omega^T\phi(x) + b \leq -1 + \xi_i \\ \xi_i \geq 0 \end{cases} \tag{2.4}$$

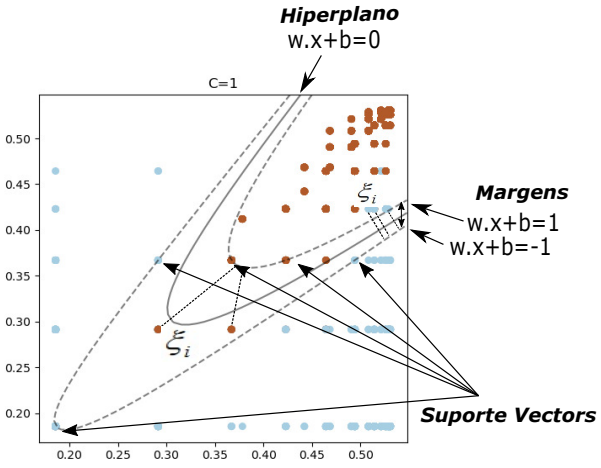

Figura 2.6: Hiperplanos de separação e margem, baseado em [3–5]

O SVR possui desempenho de predição comparável com as redes neurais do tipo MLP *(Multi Layer Perceptron)* e RBF, só que mais leves computacionalmente [51]. A Função de Base Radial (RBF) é conhecida como LS-SVM *(Least Squares Support Vector Machine)*

e seu principal benefício é ser computacionalmente mais eficiente do que o método SVM comum [50]. Carvalho et al. 2018 [25] apresentam o *kernel* RBF, utilizando na detecção de anomalias em redes SDN, com uma combinação linear de funções radialmente simétricas. Matematicamente, o modelo pode ser expresso como na Equação 2.5.

$$\hat{y} = exp(-\gamma \left\| X_i - X_j \right\|^2) \tag{2.5}$$

Dois termos muito utilizados e relevantes em ML, estão relacionados a capacidade de generalização do Modelo, ou seja, capacidade de fazer previsões precisas sobre dados não vistos [3] e que não podem faltar nesta explanação são o *Overfitting* e o *Underfitting*. Eles representam problemas que devem ser evitados em qualquer projeto de pesquisa que envolva aprendizado de máquinas que têm o seguinte significado:

Overfitting: Ocorre quando se encaixa um modelo muito próximo às particularidades do conjunto de dados de treino e o modelo tem um desempenho excelente, mas quando utilizado no conjunto de dados de teste ele não é capaz de generalizar e o resultado é ruim. É como se o modelo ao invés de aprender estivesse apenas decorando uma regra do que fazer, e ao receber um novo conjunto de dados o modelo tenta aplicar estas regras, resultando em um desempenho ruim [3].

Underfitting: Ocorre quando o modelo é muito simples ou contém excesso de atributos, nesse caso ele pode não ser capaz de capturar todos os aspectos e a variabilidade dos dados, então se sairá mal mesmo no conjunto de dados de treino. Este modelo já pode ser descartado, sem aplicá-lo ao conjunto de dados de teste, pois não terá utilidade [3]. A questão é colocar a quantidade certa de atributos, adicionando os mais relevantes, sem excesso para que o algoritmo não se perca na tentativa de encontrar relação e nem que ela seja de menos, para não causar o efeito de ausência de características relevantes que precisam ser aprendidas.

Capítulo 3

Revisão da Literatura

A revisão do estado da arte foi realizada com foco em trabalhos com seguintes critérios: Relacionados à segurança em SDN; Voltados para ataques DDoS; Últimos cinco anos, com exceção de trabalhos mais antigos que se tornaram referência em determinado tema e que não podem deixar de ser citados dada sua relevância; Contendo pelo menos duas ou mais das palavras chaves (DDoS, SDN, *Machine Learning*, Algoritmo de Regressão e Classificação). Como limitador de escopo dos trabalhos apresentados também foram utilizados os critérios: Do plano em que a solução é implementada; Distinção da técnica aplicada e melhores resultados; Similaridade com a abordagem aqui aplicada para melhor comparação dos resultados obtidos.

Este capítulo visa mostrar os desafios e abordagens que estão sendo utilizadas na literatura para tratar o tema segurança, classificando as abordagens de implementação de solução por camada da SDN, seja na camada de aplicação, camada de controle ou camada de dados. Na Seção 3.1 os desafios em SDN foram apresentados de forma sintética e as abordagens na literatura foram também agrupadas de acordo com a camada da SDN em que são implementadas as soluções propostas, sendo subdividida em três subseções. Na Subseção 3.1.1 foram apresentados trabalhos com abordagens no plano de aplicação, seguida pela Subseção 3.1.2 apresentando trabalhos com abordagens de soluções implementadas no plano de controle e por fim na Subseção 3.1.3 foram apresentados os trabalhos que utilizaram o plano de dados para implementar suas propostas de soluções. Ao final na Seção 3.2 foi feita uma discussão dos trabalhos comentados na seção anterior e em que este trabalho de pesquisa se diferencia dos demais.

3.1 Desafios e Abordagens em SDN

Singh e Behal, 2020 [16] realizaram uma revisão na literatura de soluções de detecção e mitigação de ataques em SDN e apontaram que muitas soluções de detecção de ataques

DDoS são baseadas na comparação dos valores paramétricos, usando valores de limiares predefinidos, afirmando que diferenciar o tráfego legítimo do espúrio para lidar com ataques DDoS ainda é um grande desafio. Alguns autores apresentam o problema da ausência de uma técnica para definir um limiar que seja dinâmico para melhorar a detecção dos ataques por entropia e da dificuldade de se conseguir um bom ajuste [8]. Enquanto outros empregam a técnica de entropia como primeiro filtro e aplicam ML como segundo filtro na tentativa de melhor separar os fluxos espúrios dos legítimos [6]. Por fim, nessa mesma linha, só que de forma inversa usando ML como primeiro filtro e entropia, com limiares predefinidos, como segundo filtro, também empregam esforços na tentativa de melhorar os resultados na detecção final [7].

Os trabalhos citados compartilham o mesmo objetivo, que é o de vencer o desafio do melhor limiar que separa o tráfego legítimo do espúrio pela ausência de um limiar dinâmico que melhore a eficiência dos algoritmos baseados em entropia. No entanto, estão limitados ao que a entropia por si só oferece de ajuste ou simplesmente entregam ao ML este trabalho sem considerar o custo computacional que pode ter impacto no desempenho global da rede. Muitas abordagens de pesquisa foram empregadas com diferentes técnicas de definição de limiares, elas foram aqui classificadas de acordo com a camada da SDN em que elas são implementadas seja no plano de aplicação, plano de controle ou plano de dados, apesar de poderem ser agrupadas também através do método, interface de comunicação como sul, norte ou leste/oeste e outros. Este trabalho classificou de acordo com a camada SDN de implementação da solução proposta.

3.1.1 Abordagens na Camada de Aplicação

As abordagens de implementação de segurança na camada de aplicação utilizam a interface de comunicação norte para entregar ao plano de controle as regras e políticas de segurança que serão repassadas ao plano de dados, seguindo esta estratégia de desenvolvimento é apresentado por Shin et al. 2013 [52] o FRESCO, esta solução é um *Framework* de desenvolvimento composta de módulos (Python, NOX) especificamente projetada para fornecer um interpretador e APIs para definição de políticas de aplicativos de segurança via linguagem de scripting, através de uma abstração do *OpenFlow*, se tornando um *kernel* de aplicação de segurança (SEK) de imposição de restrições de fluxo aplicáveis que possam defender a rede contra as ameaças detectadas. Os módulos são estendidos pelas APIs do FRESCO para fornecer duas funções principais de desenvolvedor: (*i*) um ambiente de desenvolvimento FRESCO, composta pelos módulos, instâncias e base de dados; (*ii*) um controlador de recursos, que fornece aos desenvolvedores de aplicativos FRESCO acesso e controle dos recursos de rede. Trabalham com valores de limiares predefinidos baseados em confiança, como por exemplo um valor limite para um algoritmo de detecção

de varredura definido em 5, ou seja, se um IP de origem gerar cinco conexões TCP com falha é considerado como um invasor.

Carvalho et al. 2018 [25] propuseram a detecção de anomalias em um ecossistema baseado em SDN, monitorando o tráfego da rede e detectando proativamente essas anomalias na rede. Para isso usaram *Multinomial Logistic Regression* (MLR) e *Support Vector Machine* (SVM) com uso do *kernel* RBF. A proposta permite a automação no monitoramento da rede e na detecção de eventos de tráfego anômalos. O Design é modular, permitindo que tarefas como coleta de estatísticas de tráfego, detecção de anomalias, mitigação de ataques e geração de relatórios sejam complementares e dissociadas, assim modificações em uma tarefa não causam interferência em outra. O controlador lida com a tabela de encaminhamento de equipamentos de rede, instalando regras de fluxo de acordo com a classificação atribuída a cada fluxo. A solução implementa duas fases de monitoramento: A primeira fase consiste em monitorar, calcular os limites, baseado em dados estatísticos e probabilidades a fim de comparar o comportamento do tráfego em relação ao esperado, ou seja, seu perfil normal. Se for detectado um desvio comportamental, com base nos limites esperados, a segunda fase inicia um monitoramento ostensivo realizado em várias janelas de tempo para identificar o atacante ou os alvos sob ataque. Os resultados obtidos para MLR foram de 95.1% de acurácia e para o SVM de 89%.

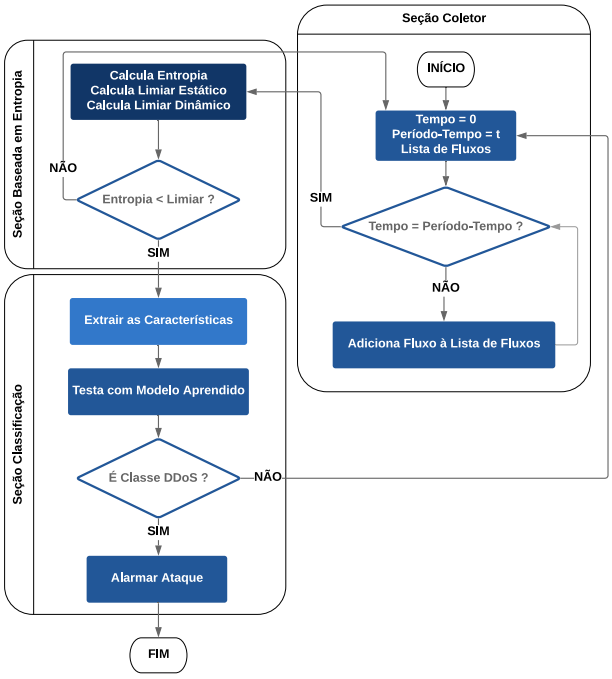

Figura 3.1: Baseado no Fluxograma da solução proposta por Dehkordi et al. 2020 [6]

No trabalho de Dehkordi et. al. [6] foi proposta a solução conforme ilustrado pela Figura 3.1, o uso de entropia e *Machine Learning*, com limiares estáticos definidos e dinâmico baseado em uma técnica em entropia, onde a entropia realiza a primeira etapa do

processo com detecção dos suspeitos que se positivo são entregues para o ML, recebendo os dados da entropia desses suspeitos, realizar a segunda etapa do processo de detecção e fim do processo. Foram utilizados 04 *Datasets* sintéticos distintos com ataque e 01 sem ataque, sendo eles ISCX-SlowDDos-2016, ISCXIDS-2012, CTU-10, CTU-11 e ISOT(dados sem ataque). Para ataques DDoS de alta e baixa taxas e os limiares são definidos estaticamente e com uma técnica dinâmica para comparação. São utilizados algorítimos de Classificação: BayesNet, J48, RandomTree, logistic regression e REPTree. Os autores compararam com trabalhos na literatura e foram melhores, apresentando apenas o melhor resultados obtido na comparação que foi de 99.85% acurácia e 0.1% Taxa de Falso Positivo (do inglês *Falso Positive Rate* - FPR). Este trabalho servirá de comparação com os resultados obtidos neste trabalho de pesquisa. A abordagem dos autores se assemelha mais à proposta de [8] com a entrega dos cálculos de entropia para ML realizar a detecção, contudo ela se difere, daquela e desta, também no ponto em que na detecção do ataque entropia faz o primeiro filtro e somente os suspeitos são submetidos à ML em segundo filtro como fase de confirmação, como pode ser percebido pelo fluxograma citado.

3.1.2 Abordagens na Camada de Controle

As abordagens de implementação de segurança na camada de controle utilizam a interface de comunicação sul para entregar ao plano de dados as regras e políticas de segurança. Mousavi e St-Hilaire 2015 [11] apresentaram como solução para detectar ataques DDoS o uso de medição da aleatoriedade do tráfego de entrada por meio da entropia. Eles consideraram que existem dois componentes essenciais para a detecção de DDoS usando entropia: (i) tamanho da janela e (ii) um limiar. A estratégia usada para definir o tamanho da janela foi atribuindo um tamanho menor ou igual a quantidade de *hosts*, definida no experimento em 50 fluxos e para definir o melhor limiar utilizaram a metodologia de vários testes com tráfego de ataque em taxas de 25%, 50% e 75%, selecionando o de melhor resultado na rodada de testes. Eles utilizaram a capacidade do controlador de monitorar os fluxos na camada de dados e adicionaram outro conjunto de coleta de estatísticas no controlador. A entropia de cada janela foi calculada e comparada com um limiar selecionado. Se a entropia for menor que o limiar, um ataque será detectado. Essas funções são pequenas (em termos de quantidade de código adicionado) e transparentes às outras funcionalidades do controlador. Essas duas propriedades tornam a solução aplicável a diferentes controladores com alterações mínimas. O resultado obtido foi de 96% de precisão.

Também na abordagem de entropia temos Prasetiawan et al. 2017 [15] que propuseram um método para a identificação de ataques Distribuídos de Negação de Serviço direcionados as redes SDN, onde os autores utilizaram a metodologia de escolha do limiar

definido como três vezes o Desvio Padrão do resultado obtido na entropia e para definição do tamanho da janela se apoiam em outro trabalho na literatura. Como conclusão os autores apresentaram o método estatístico como adequado e aderente à rede SDN pela sua característica de programabilidade na camada de controle e o método de três vezes o desvio padrão para definir o limiar como adequado para medir a aleatoriedade dos pacotes. Na apresentação dos resultados os autores se concentraram nos dados da abordagem da metodologia do desvio padrão e não evidenciaram os dados de precisão de detecção do ataque.

Sambandam et al. 2019 [13] trataram do problema do crescente número de dispositivos IoT na infraestrutura e das vulnerabilidades existentes, tais como DDoS, que precisam ser detectadas e mitigadas em SDN, a solução proposta é baseada no fator de entropia da rede, a fim de evitar sobrecarga no controlador, atenuando o ataque e interrompendo o processamento desses pacotes maliciosos, aumentando a disponibilidade de recursos para usuários legítimos. Para definição do tamanho da janela são testados tamanhos de 25 até 125 fluxos, incrementando de 25 em 25 fluxos, onde a escolha de melhor resultado foi de 50 fluxos. Para definir o limiar foi utilizado o melhor resultado nos testes de fluxos como o limiar ideal. Os resultados obtidos foram considerados pelos autores como positivos, contudo apontam para a necessidade de melhorar as taxas de erro, Falsos Negativos (FN) e Falsos Positivos (FP), mas sem apresentar os valores de precisão e taxa de erro.

Alguns trabalhos usaram a abordagem de *Machine Learning* (ML) como Dey et al. 2018 [26], neste trabalho os autores não definem limiares, pois deixam a cargo dos algoritmos de previsão este trabalho. Eles aplicaram o conjunto de dados KDD-99 versão NSL-KDD para prever possíveis intrusões usando métodos de classificação. Os algoritmos foram *Info Gain Random Forest (IGRF), Gain Ratio Random Forest (GRRF), Chi-square Random Forest (CRF), CFS Subset Evatuator PART (CFS-SEP) e Symmetric Uncertainty Random Forest (SURF)* e abordagem seguiu a seguinte sequência: (1) Seleção do algoritmo de ML mais apropriado; (2) Treino do modelo ML usando o conjunto de dados NSL-KDD; (3) Normalização das características apropriadas após o uso de diferentes métodos de seleção de características; (4) Se o modelo treinado de predição de anomalias de fluxo de melhor resultado de classificação detectar uma anomalia; (5) Atualiza as regras do controlador SDN para bloquear o tipo classificado como ataque; (6) Se o modelo não identificar anomalia o tráfego passará pelo controlador e poderá acessar os recursos de rede disponíveis. O melhor resultado obtido foi com *Gain Ratio Random Forest* a uma taxa de detecção de quase 81.95% de acurácia e uma taxa de falsos alarmes de 0.287%.

Aung e Htaik 2020 [7] utilizaram um método com entropia e *Machine Learning*, com o algoritmo *Support Vector Machine* (SVM), nos dados sintéticos ASNM-NPBO *(Advanced*

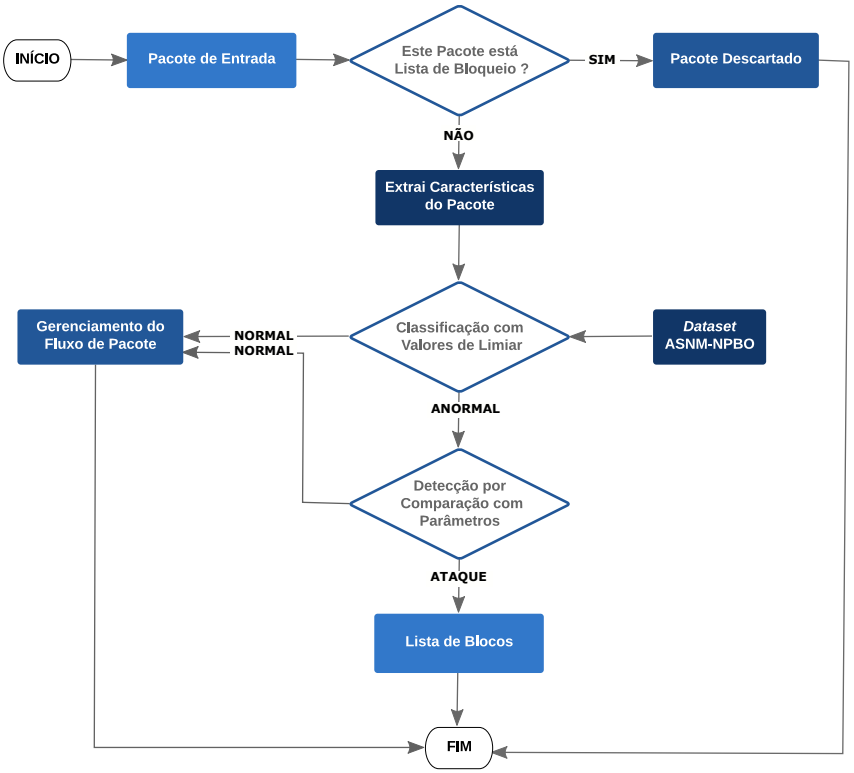

Figura 3.2: Baseado na Visão geral da abordagem híbrida proposta por Aung e Htaik [7]

Security Network Metrics and Non- Payload-Based Obfuscations). Conforme a Figura 3.2 a solução consulta uma lista de bloqueio e descarta o pacote em caso positivo, senão extrai as características do pacote, classifica em normal e segue para gestão dos fluxos ou classifica como anormal e realiza a detecção comparando com um limiar predefinido. A solução é implementada no plano de controle com controlador POX e seu resultado é comparado com a entropia tradicional sendo superior com 76.12% de acurácia e 0.05% de Taxa de Falso Positivo (FPR). Esta proposta servirá também de comparação final com os resultados desta pesquisa e se assemelha mais com dos outros dois autores de comparação, um implementado no plano de aplicação e o outro no plano de dados.

3.1.3 Abordagens na Camada de Dados

Algumas abordagens de implementação de segurança são feitas na camada de dados, desonerando a camada de controle e entregando um pouco de controle ao plano de dados. Nessa linha Braga et al. [14] também não definiram limiares e deixaram a cargo do *Machine Learning* para tal propósito com o uso da técnica SOM *(Self-Organizing Maps)*, rede neural artificial não supervisionada, treinada com recursos do fluxo de tráfego, com uma abordagem baseada no conjunto de dados KDD-99 para detectar ataques DDoS. O método foi divido em três partes: (1) controlador NOX; (2) *Loop* de detecção; (3) Infraestrutura

de rede. A detecção consiste em monitorar todos os *switches* registrados no controlador NOX durante intervalos de tempo predeterminados, durante estes intervalos os fluxos de dados são coletados. Através da SOM o modelo foi treinado com a amostra de tráfegos estatísticos de fluxos de dados e depois submetidos a análise do classificador de rede neural artificial a fim de detectar se o tráfego é normal ou ataque. O melhor resultado obtido foi de 99.11% de acurácia e taxa de erro de 0.46%.

Xu e Liu [53] também não definiram limiares e também aplicaram técnica SOM *(Self-Organizing Maps)* em SDN, implementando *switches* com memória TCAM, por serem mais rápidas e permitirem em um único ciclo a varredura de todo o conteúdo. Para capturar as características de taxa de tráfego e desvio/assimetria da taxa de tráfego, para obter alta precisão na detecção cada *range* de IP de vítima suspeita, foi instalado um par de regras para capturar os fluxos de entrada do *range* e os de saída, garantindo que as granularidades de *range* do par de regras sejam consistentes. Para utilizar colaborativamente o recurso limitado do TCAM disponível em todos os comutadores para monitorar toda a rede, coordenaram o posicionamento das regras do monitor em todos os comutadores para utilizar com eficiência todas as entradas do TCAM disponíveis em toda a rede para maximizar a cobertura e minimizar a granularidade da detecção. Um procedimento adaptativo foi proposto para ampliar dinamicamente os possíveis *ranges* de IP da vítima e do atacante e diminuição do *range* de IPs normais. Um método sequencial e um método concorrente para detectar vítimas e atacantes também são propostos para fins de comparação. Os resultados mostraram que, se a prioridade da detecção de DDoS é encontrar vítimas, o Método Sequencial é preferível, pois ele pode detectar *range* de IPs potenciais mais granular, considerando tamanhos TCAM limitados. Se o objetivo da detecção de DDoS é encontrar vítimas e atacantes, o Método concorrente é preferível se os tamanhos do TCAM forem abundantes, pois ele encontra vítimas e atacantes mais rapidamente.

Sun et al. 2018 [54] apresentam o desafio de se diferenciar vários tipos de DDoS como: *SYN flood, ICMP flood, UDP flood, ACK flood e TCP connection flood* de *Flash Events* (FE) que são tráfegos legítimos mas que se assemelham ao volume gerado por um DDoS com base nas características do fluxo de modo a obter alta precisão de resolução. Para isso é proposto um método de detecção de DDoS de vários tipos e de FE com base nas características de fluxo, otimizando o método *K Nearest Neighbors* (KNN) com adoção de características multidimensionais da tabela de fluxo para integração e classificação, incluindo os cálculos de φ-Entropia de alguns atributos para diferenciar o tráfego normal do anormal, distinguindo assim DDoS de FE. Os limiares para φ-Entropia em α são predefinidos (0.1, 0.3, 0.5, 0.7 e 0.9).

Para obter os resultados os autores seguiram com a Coleta da Tabela de Fluxos, extração e agregação das principais informações de recurso como tipo de protocolo, duração

da tabela de fluxo, divergência do IP de origem que compartilha o endereço IP de destino comum e a taxa crescente da tabela de fluxo e, de acordo com os diferentes tipos de ataques, selecionou os recursos correspondentes para detecção. Como medida de dispersão de algumas características, usou a distância de informação de diferentes concentrações de dados que pode ser efetivamente aumentada, de modo a melhorar a precisão da discriminação. Se a detecção de ataques for considerada como um problema de classificação, os dados fornecidos são classificados e o status atual da rede será normal ou anormal. O KNN é considerado adequado para classificar eventos raros e particularmente adequado para problemas de multi-classificação baseados nas características de tráfego de ataque DDoS e compara com outros algoritmos de classificação como o SVM.

A tabela de fluxo gerada pelo tráfego foi coletada para gerar amostras de aprendizado e divididas em sete grupos: amostras de treinamento de tráfego normal, amostras de treinamento de eventos FE e cinco tipos de amostras de treinamento de tráfego de ataque DDoS. Na fase de detecção de tráfego, foi coletada a tabela de fluxo no *switch* e extraídas várias tuplas para classificar usando o KNN. Ao mesmo tempo, usaram o classificador SVM como experimento de controle. A partir dos resultados, demonstram que usando o algoritmo KNN e entropia de shannon para a classificação obtém-se maior precisão de 94% e menor taxa de falso positivo de 0.010% do que usando o algoritmo SVM. Concluem ainda que a taxa de precisão dos eventos FE ainda precisa ser melhorada, mas pode impedir que um tráfego grande normal seja interceptado ou descartado até certo ponto.

No trabalho de Lapolli et al 2019 [55] os autores apresentam uma abordagem de detecção de ataques DDoS em tempo real diretamente no plano de dados, implementado no *switch* com P4. Esta proposta é aderente ao conceito de NG-SDN com funcionalidade de controle de *pipeline* que permite programar dinamicamente as tabelas de encaminhamento em um *switch* SDN. Para isso construíram o mecanismo de detecção sobre a implementação do modelo de referência comportamental P4 (BMv2). Definiram as janelas em 218 fluxos para um total de 250 a 500 janelas de observação para carga de trabalho nas fases de treinamento (tráfego sem ataque) e detecção (tráfego com ataque), num total de 7 cargas de trabalho. Foram utilizados os datasets: com ataque do *"CAIDA DDoS Attack 2007"* e sem ataque o *"CAIDA Anonymized Internet Traces 2016"*. Com 15 repetições para cada configuração com coeficientes de *hashing* aleatórios e IC 95%.

A solução é composta pelo seguinte processo: Das janelas se extrai os IPs de Origem e Destino que servem de entrada para a etapa dentro do contexto P4 onde os cálculos de entropia são realizados e depois seguem para uma etapa de caracterização, que é o treinamento realizado com tráfego legítimo, e também para a etapa seguinte, juntamente com os resultados da caracterização, onde ocorre detecção da anomalia, utilizando o tráfego com ataque, e alarma em caso de ataque. Para o alarme a solução faz uso de

coeficientes de sensibilidade para calcular o limiar k que é um parâmetro configurável e escalonável, sendo multiplicado pelo índice de dispersão proporcional às características do tráfego. O aumento do k resulta em maior assertividade. Um valor menor para k pode expandir a cobertura de detecção de anomalias com aumento de falsos alarmes. Os autores recomendam que os administradores de rede ajustem de forma adaptável o valor de k para alcançar um equilíbrio entre as taxas de verdadeiros e falsos positivos. Como resultado obtiveram 98.2% de acurácia e latência de $\approx 250ms$ com baixo uso de recursos do dispositivo (em termos de SRAM e TCAM).

Finalizando os trabalhos no plano de dados a proposta de Agarwal e Mittal 2012 [8] e que também servirá de comparação com os resultados obtidos neste trabalho de pesquisa. Neste trabalho os autores utilizaram um conjunto de dados do DARPA-1999 e propuseram um método de detecção híbrido baseado em entropia e ML onde os resultados da entropia que são entregues para o ML, utilizando SVM e este por sua vez faz a detecção do ataque, conforme Figura 3.3. Um limiar é escolhido manualmente e uma janela por tempo é definida. São calculadas a entropia de IP de origem, IP de destino, porta de origem, porta de destino, tamanho do pacote e tipo do pacote. Depois de calculada a entropia individual desses atributos os resultados são submetidos ao treino do *Machine Learnig* (ML) utilizando o algorítimo SVM, em seguida é realizado o teste, conforme preconizado pelo KDD e por fim a detecção do ataque é realizada pelo ML. Os resultados de Agarwal e Mittal 2012 foram de 97.25% de acurácia e 2.75% de taxa de erro. Os autores ainda apresentam o problema da ausência de uma técnica para definir um limiar que seja dinâmico para melhorar a detecção dos ataques por entropia e da dificuldade de se conseguir um bom ajuste.

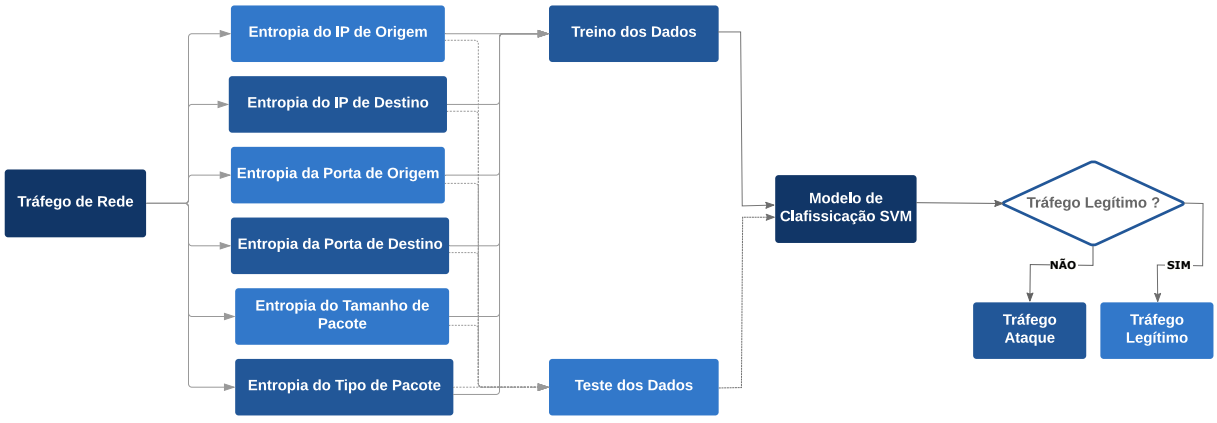

Figura 3.3: Baseado no Fluxograma da abordagem híbrida proposta por Agarwal e Mittal [8]

As abordagens dos autores nos trabalhos comentados diferem do presente trabalho

de pesquisa tanto em relação aos algoritmos utilizados quanto em relação às técnicas. Enquanto utilizam SVM comum e aplicam entropia e ML como filtro um do outro, aqui usamos algoritmos derivados do SVM (SVR e SVC), onde o SVR, computacionalmente mais vantajoso segundo [50,51], é usado para encontrar o melhor coeficiente de previsão do limiar e devolvê-lo ao algoritmo de entropia em ML-Entropia. Além disso apresentamos técnicas alternativas híbridas com algoritmos de ML de regressão (SVR) e classificação (SVC e *Random Forest*), recebendo os cálculos de entropia como entrada de dados e realizando a detecção do ataque, sem filtro. Outro diferencial é que aqui o limiar é dinâmico e ajustável ao estado atual da rede com uso de ML por meio de novo treino/teste do modelo, conforme uma rega, sem impactar o desempenho geral da rede, uma vez que ele é realizado fora e a detecção é realizada pela entropia que é mais leve [11, 13, 15, 41].

3.2 Discussão

As soluções de segurança, conforme Tabela 3.1, classificadas por camada da SDN, apresentadas na seção anterior, representam o estado da arte de trabalhos relacionados com a proposta deste trabalho de pesquisa. A estratégia de proteção de segurança SDN completa ainda não foi proposta [10]. Os trabalhos utilizam diversas estratégias para alcançar o objetivo de melhor separar um tráfego espúrio do legítimo com definição de um limite. Técnicas alternativas para melhorar o limiar aproximam estes trabalhos em termos de desafio na busca de maior acurácia e menor taxa de erros possíveis. Foram apresentadas uma gama de possibilidades de usabilidade de diversos métodos na detecção de ataque em SDN, em especial no uso de *Entropia* e *Machine Learning*. Mas ressaltamos que é importante garantir o desempenho, escalabilidade e outras características relevantes das redes. Em entropia eles utilizam métodos com limiares estáticos ou dinâmicos a partir de uma determinada abordagem que não acompanha as constantes mudanças de estado da rede. No caso das abordagens com ML simplesmente entregam aos algoritmos esta missão, sem considerar o custo computacional no desempenho geral da rede no treino e teste do modelo.

Tabela 3.1: Sumário das soluções classificadas por camada SDN.

PLANO SDN	AUTOR	MÉTODO	RESULTADO	VANTAGEM	DESVANTAGEM
	Proposto por [52]	Aplicação baseada em script que define regras e políticas de segurança ao controlador SDN. Trabalham com valores de limiares predefinidos baseados em confiança.	90% menos códigos de soluções similares	Permite a definição de regras e políticas por API	Vulnerável a sobreposição e conflito de regras de aplicativos. O Limiar é baseado em confiança.
Aplicação	Proposto por [25]	Uso de ML (MLR, SVM, RBF). Limites calculados, baseado em dados estatísticos e probabilidades comparado ao esperado.	MLR = 95.1% de acurácia e SVM = 89%.	Automação de contramedidas nos dispositivos afetados para manter a disponibilidade dos serviços da rede.	Custo computacional impacta no desempenho geral da rede. O Limiar é probabilístico.
	Proposto por [6]	Uso de entropia e ML (BayesNet, J48, RandomTree, logistic regression e REPTree). Limiares estáticos definidos e dinâmico baseado em uma técnica em entropia.	Foram comparados com trabalhos relacionados com melhor resultado de 99.85% de acurácia e 0.1% de FPR.	Uso de abordagem com coleta sem sobrecarga do controlador e Alta acurácia e baixa taxa de falso positivo.	Uso de dados sintéticos. Usa Limiar estático e Limiar dinâmico que está limitado às possibilidades em entropia.
	Proposto por [11]	Entropia. Para definir o limiar utilizaram a metodologia de vários testes com tráfego de ataque em taxas de 25%, 50% e 75%.	96% de precisão	Aplicável a diferentes controladores com alterações mínimas.	Janela e limiar são configurados manualmente. O Limiar não acompanha o estado atual da rede.
	Proposto por [15]	Entropia. Método de três vezes o desvio padrão para definir o limiar.	Na apresentação dos resultados os autores se concentraram nos dados da abordagem da metodologia do desvio padrão.	O mecanismo permite ajuste ao tráfego de rede em relação a uma definição manual do limiar.	Não apresentam os resultados de precisão de detecção do ataque. O Limiar está limitado às possibilidades em entropia.
Controle	Proposto por [13]	Entropia. Para definir o limiar foi utilizado o melhor resultado nos testes de fluxos como o limiar ideal.	Os resultados considerados pelos autores como positivos, mas sem apresentar os valores de precisão e taxa de erro.	Utilizou Raspberry Pi como OpenvSwitches que reduz custos.	Necessidade de melhoria das taxas de erro, Falsos Negativos (FN) e Falsos Positivos (FP). O Limiar não acompanha o estado atual da rede.
	Proposto por [26]	Uso de ML (IGRF, GRRF, CRF, CFS-SEP e SURF). Os autores não definem limiares, pois deixam a cargo dos algoritmos de previsão este trabalho.	GRRF = 81.95% de acurácia e uma taxa de falsos = 0,287%.	O método se adapta às características da rede no momento do treino.	O aprendizado e definição do modelo ML é feito em determinado estado da rede e não acompanha sua mudança. Custo computacional no treino e teste do Modelo em ML.
	Proposto por [7]	Uso de entropia e ML (SVM), nos dados sintéticos ASNM-NPBO, faz uso de lista de bloqueio e filtro com ML e depois com entropia na detecção. Faz uso de um limiar predefinido.	76.12% de acurácia e 0.05% de FPR.	Baixa taxa de falso positivo.	Gera sobrecarga no Plano de Controle. Acurácia baixa. O Limiar não acompanha o estado atual da rede.
	Proposto por [14]	Uso de ML (SOM), rede neural artificial não supervisionada. Os autores não definiram limiares e deixaram a cargo do ML.	99.11% de acurácia e taxa de erro de 0.46%.	O método se adapta às características da rede no momento do treino.	O aprendizado e definição do modelo ML é feito em determinado estado da rede e não acompanha sua mudança. Custo computacional no treino e teste do Modelo em ML.
	Proposto por [53]	Uso de ML (SOM), em *switches* com memória TCAM. Os autores não definiram limiares e deixaram a cargo do ML.	Recomenda Método Sequencial em condições de TCAM limitados, do contrário Concorrente.	O método se adapta às características da rede no momento do treino.	TCAM limitado com maior tempo para detecção. TCAM abundante maior custo. Custo computacional no treino e teste do Modelo em ML.
Dados	Proposto por [54]	Uso de ML (KNN) recebendo atributos relevantes e cálculos de φ-Entropia de alguns atributos. Os limiares para φ-Entropia em α são predefinidos	Precisão de 94% e Taxa de falso positivo de 0.010%	Distingue tráfego normal FE de tráfego espúrio DDoS.	Afeta o desempenho do plano de dados. O Limiar não acompanha o estado atual da rede.
	Proposto por [55]	Uso de entropia com linguagem P4 para implementação no switch SDN	98.2% de acurácia e latência de $\approx 250ms$. Baixo uso de recursos (em SRAM e TCAM)	Viabiliza a implementação de segurança no *switch*	Conjunto de instruções no P4 é muito restrito. Limiares não acompanham o estado atual da rede
	Proposto por [8]	Entropia e ML (SVM). Um limiar é escolhido manualmente.	97.25% de acurácia e 2.75% de taxa de erro.	O método se adapta às características da rede no momento do treino.	O aprendizado é definido em um estado da rede apenas. O Limiar não acompanha o estado atual da rede.

Capítulo 4

Proposta de Trabalho

O objetivo deste trabalho é desenvolver um mecanismo híbrido de detecção de ataque DDoS em SDN, utilizando Entropia e *Machine Learning* que melhor separe o tráfego legítimo do espúrio, pela ausência de um limiar dinâmico que melhore a eficiência dos algoritmos baseados em entropia. Aqui exploramos a utilização de ML, especificamente algoritmo de regressão (SVR-RBF), fornecendo o valor do limiar que é o coeficiente de previsão encontrado, tornando dinâmico os limiares de algoritmos que utilizam entropia e apresentamos um método híbrido utilizando entropia e ML, aqui denominado ML-Entropia, para diferenciá-lo da entropia tradicional e também três alternativas híbridas de ML (SVR híbrido, SVC híbrido e *Random Forest* híbrido).

O foco em relação ao limiar é aprimorar o valor/peso do atributo/parâmetro selecionado como limite na comparação par a par, conforme exemplificado no Capítulo 2, Seção 2.5, calculado pela entropia. O limiar que é definido é arbitrado pelo administrador de rede, conforme sua realidade, aqui denominado de forma genérica de "Limiar-X", que serve para definir a separação dos dados de ataque e não ataque no conjunto de dados, mas são seus pesos/coeficientes, que será representado como T (para Entropia) e como T^i (para ML-Entropia) que determinam o ajuste fino e o quão próximos os parâmetros estão da fronteira de melhor separação desses dados. Procuramos provar que seja qual for o limiar definido (Limiar-X) ele pode ter o seu T melhorado em T^i pela técnica de ML-Entropia, melhorando os resultados na detecção com redução das taxas de erro em relação a Entropia tradicional. Isto é possível por meio do hiperplano encontrado pelo algoritmo de regressão, ou seja o coeficiente de previsão que retornará ao algoritmo de entropia, substituindo T por T^i no Limiar-X. Além disso ela permite que o limiar seja dinâmico uma vez que o processo pode ser automatizado e o treino/teste pode ser refeito de tempos em tempos, conforme um critério arbitrado de tempo ou variações percebidas no fluxo de dados, para assim ajustá-lo ao estado atual da rede.

Para atingir o objetivo, a abordagem proposta seguiu conforme a Figura 4.1, em cinco

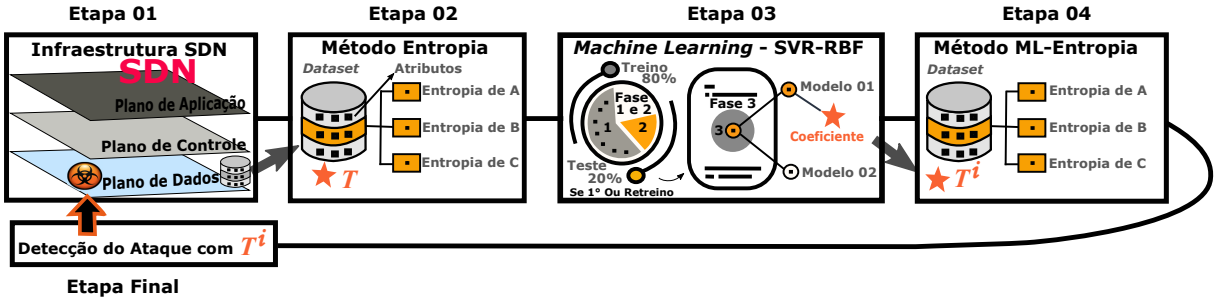

Figura 4.1: Abordagem proposta para detecção de ataque usando Entropia e *Machine Learning*.

etapas da seguinte forma: Na Etapa 01 usando um método onde os dados coletados no plano de dados são entregues para a Etapa 02; Os cálculos de entropia são realizados e utilizando o Limiar-X com peso T entropia realiza a detecção do ataque por si só, continuando o processo, os resultados dos cálculos de entropia são entregues para a Etapa 03; Em ML o processo é dividido em três Fases, na Fase 1 ocorre o treino com 80% dos dados aleatórios, na Fase 2 ocorre o teste com 20% dos dados aleatórios e por fim a Fase 3 o melhor Modelo encontrado será usado pelos Algoritmos SRV, SVC e *Random Forest* na detecção do ataque por si só, mais uma vez continuando o processo, o Algoritmo de Regressão (SVR-RBF) fornece o coeficiente de previsão encontrado para a Etapa 04; O coeficiente de previsão é atribuído para T^i que substitui T no Limiar-X em "ML-Entropia" que realiza a detecção de ataque na Etapa Final na abordagem ML-Entropia. Ao final do processo temos a técnica tradicional e as técnicas aqui propostas todas realizando a detecção do ataque da seguinte forma:

1. Entropia (com Limiar-X em T) = Abordagem tradicional;
2. ML-Entropia (com Limiar-X em T^i recebido do SVR-RBF) = Abordagem híbrida;
3. SVR (Regressão) = Abordagem híbrida;
4. SVC (Classificação) = Abordagem híbrida;
5. Random Forest (Classificação) = Abordagem híbrida.

O Treino (Fase 1) e Teste (Fase 2) do modelo de ML, na Etapa 03, são realizados na primeira execução e poderão ser refeitos conforme regra definida pelo administrador da rede, como por exemplo sempre que um determinado tempo for atingido, ou o *throughput* ou ainda a latência de rede alterar uma quantidade de vezes o desvio padrão medido inicialmente, ou seja, a regra depende da imaginação e necessidade do administrador da rede, assim sendo quando a regra for satisfeita será realizado o retreino e descoberta do melhor coeficiente de limiar para a detecção, conforme o estado da rede daquele momento,

com devolução ao algoritmo de entropia em ML-Entropia. Este processo permite que o Limiar seja dinâmico e acompanhe o estado atual da rede.

Como o processo de treinamento e aprendizado no ML (Fases 1 e 2) representa um custo de processamento mais alto, onde o treinamento será necessário de tempos em tempos para ajustar as constantes mudanças na rede e, assim, fornecer um mecanismo dinâmico ou adaptativo de melhor limiar a implementação é feita fora no Plano de Controle. Essa estratégia também resolve o problema da implementação no mundo real onde se tem um *blackbox* como controlador, além de não impor mais carga ao controle central do que o necessário e nem impactar no desempenho geral da rede.

Para simplificar o entendimento e comprovação da abordagem foi definido como regra para refazer os cálculos (Fases 1 e 2 da Etapa 03) e assim acompanhar o estado atual da rede o fator tempo decorrido desde o último cálculo, cabendo ao administrador da rede a definição de melhor fator de variação percebida no fluxo que represente a mudança do estado de rede atual de seu ambiente de rede. Esta é uma questão que ficará em aberto para pesquisas futuras aferirem qual é a melhor estratégia de definição de qual comportamento da rede melhor caracteriza a necessidade de recálculo, minimizando o impacto e maximizando os resultados.

Nossas contribuições são: Apresentar um método híbrido que melhor separa o tráfego legítimo do espúrio com redução da taxa de erro em relação à entropia tradicional; Usar ambiente Real para validar o experimento e o método; Usar tráfego real ao invés de sintético na validação do método; Usar uma técnica alternativa na coleta de dados que não induza sobrecarga de processamento ao controle centralizado e finalmente apresentar não apenas uma mas quatro técnicas no método híbrido viáveis de implementação, Habilitando uma Técnica com utilização de ML para o cálculo dinâmico dos limiares acompanhando o estado atual da rede.

Foram comparados os resultados de três abordagens semelhantes na literatura com os resultados deste experimento. Além disso comparamos também Entropia tradicional e todas as técnicas híbridas entre si, ou seja, as quatro soluções derivadas do mecanismo híbrido implementado quais sejam: ML-Entropia, SVR Híbrido, SVC Híbrido e *Random Forest* Híbrido. Este experimento abrangeu um método estatístico (Entropia tradicional) e da parte de ML um Algoritmo de Regressão (SVR) e dois de Classificação (SVC e *Random Forest*), além de ML-Entropia que viabiliza o uso de ML nos algoritmos de entropia, todos na abordagem híbrida. Todos os resultados foram apresentados e comparados no Capítulo 6, Seção 6.2.

O experimento foi conduzido primeiro com a definição do escopo de implementação no ambiente real em termos de segmento lógico, visto que o ambiente real possui vários segmentos de rede, tais como segmento de clientes, segmento de servidores de produção,

segmento de servidores de desenvolvimento e teste, segmento de fornecedores, DMZ, entre outros. O escopo do projeto neste sentido foi definido no segmento de servidores de produção, incluindo a representação do segmento de clientes para simplificar o Modelo e representar os acessos legítimos e espúrios, emulado no MiniNet, que serviram de modelo para verificar a viabilidade da abordagem em ambiente real e a confirmação dos resultados esperados.

Foi implementado um coletor de dados fora do Plano de Dados para não impactar no desempenho geral da rede e realizar todo o processo neste servidor. Foram promovidos ataques DDoS volumétrico com *spoofing* em servidores alvos visando a aplicação, gerando tráfego legítimo aleatório nesse ambiente emulado para reproduzir ao máximo possível as condições que seriam encontradas no ambiente real. Foram coletados dados com ataque e sem ataque para análise e aplicação das técnicas e algoritmos já comentados. Foram definidas três janelas por fluxos e três janelas por tempo em dados com ataque e dados sem ataque para aferição e comparação dos resultados. As métricas foram definidas e todo o experimento foi detalhado no Capítulo 6, Seção 6.1. Ressaltando todo este processo foi feito primeiramente no Modelo Emulado (MiniNet) e depois aplicado em Ambiente Corporativo Real, onde os resultados são compilados, apresentados e analisados no Capítulo 6, Seção 6.2. Os fluxos coletados foram entregues ao cálculo de Entropia tendo como entrada os seguintes atributos selecionados:

1. IP de Origem;
2. IP de Destino;
3. Porta de Origem;
4. Porta de Destino;
5. Freqüência de Aparições do IP de Origem;
6. Freqüência de Aparições do IP de Destino;
7. Freqüência de Aparições do Porta de Origem;
8. Freqüência de Aparições do Porta de Destino;
9. N-Clientes = Freqüência de Distintos IPs de Origem.

Os resultados dos cálculos de Entropia de todos os itens anteriores foram então utilizados para dar entrada no ML com os seguintes dados:

1. Entropia Máxima de todos;
2. Entropia Média de todos;

3. Entropia Mínima de todos;
4. Desvio Padrão de todos;
5. Entropia Total de cada;

O Diagrama de Fluxo, ilustrado na Figura 4.2, descreve o processo implementado desde a coleta dos dados até o resultado da detecção com os seguintes passos:

- Se for a primeira execução ou se a regra que define a atualização do Modelo for satisfeita o fluxo seguirá para 01A, senão seguirá para 01B;
- 01A - Calcula a Entropia Individual dos Atributos de Entradas Definidos;
- 02A - Calcula por janela a Entropia Máxima, Média, Mínima, Desvio Padrão e a Entropia Total de cada um deles, além de uma coluna classe criada para informar ao algoritmo no método supervisionado o que é realmente "Ataque" e o que é "Normal" para aferição das métricas definidas de Acurácia, Precisão, Taxa de Erro, *Recall* e *F1-Score* a partir da Matriz de Confusão;
- 03A - Cria um Arquivo de Saída com os resultados do Passo 02A e entrega estes resultados para o próximo passo;
- 04A - Seleciona o(s) Limiar-X(s) definido a partir dos resultados do Passo 03A para a detecção do ataque;
- 05A - Aplica o Limiar-X do 04A para a Entropia sozinha Detectar o Ataque e gera uma Matriz de Confusão.

Na sequência, já passado pelas etapas anteriores, o arquivo criado no Passo 03A é entregue para o *Machine Learning* que aplicará para cada um dos algoritmos citados anteriormente realizando o treino (com 80% dos dados aleatórios) e teste (com 20% dos dados aleatórios) para encontrar o melhor Modelo, o fluxo seguirá com os seguintes Passos:

- 01B Cada Algoritmo SVR, SVC e *Random Forest* com seu melhor Modelo encontrado realiza a detecção do ataque por si só e geram suas Matrizes de Confusão que servirá para futura comparação de resultados;
- 02B Aplica o coeficiente de previsão fornecido pelo Algoritmo SVR no Limiar-X na ML-Entropia que realiza a detecção do ataque e gera a Matriz de Confusão;
- Por fim temos todos os resultados de detecção nas Matrizes de Confusão de cada um que serviram de aferição nas métricas definidas e melhor comentadas no Capítulo 6 "Experimento, Resultados e Análise".

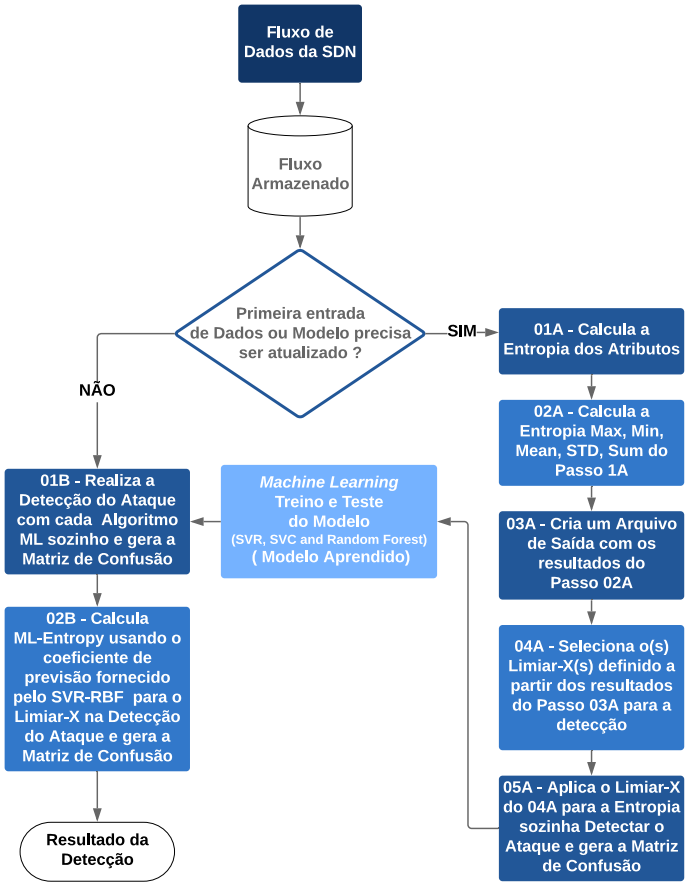

Figura 4.2: Diagrama de Fluxo de Detecção de Ataque.

O uso do cálculo de entropia foi abordado por [11] aqui explicado de uma maneira muito breve, onde, caso o tráfego aumente em direção a um determinado IP muito mais do que o previsto na entropia, esse tráfego é visto como ataque e descartado. No cálculo da entropia usado, os cálculos das entropias individuais para cada atributo ou elemento são realizados em janelas definidas por N fluxos ou segundos decorridos de acordo com a Equação 4.1.

$$E_i = -P_i \log_2 P_i \tag{4.1}$$

Na Equação 4.1, P é a probabilidade de um elemento de entrada, como exemplo segue-se um elemento específico aparecer em uma janela de N fluxos de n elementos, onde E_i é a entropia de um elemento individual. Após o cálculo da entropia individual, é necessário calcular a entropia média de acordo com a Equação 4.2.

$$\overline{E} = \frac{\sum_{i=1}^{n} E_i}{n} \tag{4.2}$$

Na Equação 4.2, $\overline{E}$ é a entropia média, n é a quantidade de um elemento específico, mostrados na janela de N fluxos e E_i, é a entropia individual, para cada elemento. O

desvio padrão calculado de acordo com a Equação 4.3.

$$E_{SD} = \sqrt{\frac{\sum_{i=1}^{n}(E_i - \overline{E})^2}{n}} \tag{4.3}$$

A entropia máxima é encontrada com a Equação 4.4, onde a entropia individual é calculada com a seleção do maior valor obtido.

$$E_{Max} = Max_{i=1}^{n} E_i, \tag{4.4}$$

Da mesma forma, a entropia individual é calculada selecionando-se o menor valor obtido em entropia mínima com a Equação 4.5.

$$E_{Min} = Min_{i=1}^{n} E_i. \tag{4.5}$$

Somados os resultados de cada um dos elementos temos a entropia total de cada um. Todos os resultados são armazenados em um arquivo para uso posterior com todos os resultados de entropia, juntamente com a coluna classe criada para informar aos algoritmos os verdadeiros positivos e verdadeiros negativos na criação das Matrizes de Confusão. Os resultados desses cálculos foram introduzidos no ML, usando SVR com RBF, a fim de encontrar o coeficiente de previsão para retornar ao cálculo da entropia, substituindo o valor em Limiar-X, para melhorar a detecção de ataques. O SVR é um tipo de regressão e em conjunto com o *kernel RBF* é usado na predição de classes. Ele é um modelo de aprendizagem de ML que constrói um hiperplano ou um conjunto de hiperplanos nas dimensões do espaço superior ou inferior que pode ser utilizado para realizar a classificação e regressão [17].

Nem sempre as amostras são linearmente separadas, neste caso há a necessidade de se fazer uma transformação para uma base superior, que foi realizado com o auxílio de uma outra função. No caso deste trabalho foi utilizada a *Radial Basis Function* (RBF). Matematicamente essa transformação apresenta, conforme [56], no lugar do produto escalar entre os vetores, uma transformação não linear entre eles, a qual é referenciada como função *kernel* (RBF), resultando na seguinte Equação 4.6.

$$f(x) = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)k(x_i, x) \; + \; b \tag{4.6}$$

Essa função ajuda a encontrar um hiperplano mais adequado que acompanha as características do conjunto de dados analisado e assim permite uma menor taxa de erro, uma vez que os dados dificilmente estão distribuídos linearmente, assim essa distribuição não linear é melhor separada por uma função com essa característica.

Capítulo 5

Metodologia

5.1 Metodologia do Projeto de Pesquisa

Para atingir os objetivos deste projeto foi utilizada a metodologia *Design Science Research* (DSR) que apoia um paradigma de pesquisa pragmática para promover a criação de artefatos a fim de resolver problemas da vida real [57,58] e conforme [59] sua metodologia de DSR, sintetizada a partir de propostas anteriores, pode ser resumida em **Construção** e **Avaliação** que é composto da seguinte forma:

A Construção é composta por Definição do Problema, Proposta de Solução e Desenvolvimento da Solução, conforme descrito:

- Definição do Problema: Abrange a identificação do Problema e Motivação, Definição do Problema de forma clara com a demonstração da relevância. Nessa etapa de identificação do problema em DSR serve ao propósito de assegurar que um problema seja significativo.
- Proposta de Solução: Define objetivos da solução e o que um artefato melhor realizaria. Deve ser demonstrado que o problema de design previsto é importante para a prática, novo e representa uma lacuna de pesquisa ou resulta da incapacidade de artefatos existentes para acomodar um novo ambiente ou contexto. Os seguintes métodos podem ser aplicados: Afirmação, Revisão de literatura (identificar estudos de questões críticas, lacunas de pesquisa ou artefatos), Rever iniciativas de profissionais, Entrevista especializada, Grupos focais e Pesquisas [58]. O método aqui escolhido foi a Revisão da Literatura.
- Desenvolvimento da Solução: Aqui é definido o projeto, seu desenvolvimento e os artefatos. A atividade de projeto serve para mostrar que um projeto de artefato contém a solução para o problema declarado. Como o artefato ainda não foi construído e, portanto, não foi aplicado, essa avaliação é artificial. Os possíveis critérios

de projeto pertinentes a essa atividade de avaliação são viabilidade, acessibilidade, inteligibilidade, simplicidade, elegância, integridade ou nível de detalhamento. Os métodos a seguir geralmente se aplicam a essa atividade: Afirmação, Prova matemática, Raciocínio lógico, Demonstração - *Ex ante*, Emulação, Simulação, Benchmarking, Entrevista especializada e Foco do grupo [58]. Aqui foi utilizado Emulação e Demonstração *ex ante*.

A Avaliação é composta por Demonstração da viabilidade, Avaliação dos resultados e Comunicação dos resultados, conforme descrito:

- Demonstração da Viabilidade: Aqui deve-se encontrar o contexto adequado e Uso de artefato para resolver o problema. Serve para demonstrar inicialmente se e quão bem o artefato funciona enquanto interage com elementos organizacionais. Nesta atividade, algumas inferências sobre a utilidade de um artefato já podem ser feitas. Como essa atividade vincula avaliações *ex ante e ex post* de um artefato, é fundamental refletir um projeto de artefato e, assim, iniciar e informar iterações subseqüentes da atividade de design de artefato. Tanto métodos de avaliação artificiais quanto naturais podem ser aplicados aqui. Assim, as "realidades"aqui consideradas podem compreender subconjuntos de "tarefas reais", "sistema real"e "usuários reais". Protótipos são freqüentemente usados neste estágio. Possíveis critérios de projeto podem incluir viabilidade, facilidade de uso, eficácia, eficiência, fidelidade com o fenômeno do mundo real, operacionalidade, robustez ou adequação. Os seguintes métodos podem ser aplicados: Demonstração com protótipo, Experimento com o protótipo, Experimento com o sistema, Benchmarking, Pesquisas, Entrevista especializada e Foco do grupo. Aqui foi utilizado Demonstração com protótipo.

- Avaliação dos Resultados: Observação da eficácia e eficiência; Iteração de retorno com o design. Em última análise demonstra que um artefato é aplicável e útil na prática, além disso, os pesquisadores podem querer teorizar sobre os princípios de design subjacentes ao artefato.

- Comunicação dos Resultados: Publicações acadêmicas ou Publicações profissionais para demonstrar credibilidade. Somente avaliações naturalísticas serão aplicadas aqui, isto é, o contexto organizacional. Os possíveis critérios de projeto pertinentes a essa atividade de avaliação são aplicabilidade, eficácia, eficiência, fidelidade com o fenômeno do mundo real, generalidade, impacto no ambiente de artefatos e usuário, consistência interna ou consistência externa. Os métodos a seguir geralmente se aplicam a essa atividade: Estudo de caso, Experimento de campo, Pesquisa, Entrevista especializada e Grupo focal. O resultado deste projeto de pesquisa foi submetido para avaliação de publicação no *Journal of Information of Security and*

Application da revista Elsevier, com resposta de aceitação pendente, até esta data, para publicação em 2021.

5.2 Metodologia da Avaliação de Desempenho da Solução

A avaliação de desempenho da solução proposta de mecanismo híbrido de detecção de ataques DDoS em SDN segue a metodologia do livro "*The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modelin*" de Raj Jain 1991 [60] por ser adequada para este tipo de experimento, seguindo o protocolo científico quanto a necessidade de avaliação, segundo ele dez etapas devem ser seguidas para uma boa avaliação de desempenho de uma solução que são:

1. Definir os **Objetivos** do estudo e o que constitui o **Sistema**, delineando seus limites;
2. Listar os **Serviços** desse sistema e os resultados possíveis, desejáveis ou não;
3. Selecionar critérios ou **Métricas** para comparar o desempenho;
4. Listar todos os **Parâmetros** que afetam o desempenho, que pode ser dividida em parâmetros do sistema, que geralmente não variam entre as várias instalações, e parâmetros da carga de trabalho, que caracterizam as solicitações dos usuários e variam de uma instalação para a outra;
5. Listar os **Fatores**, que são os parâmetros que variam, definindo seus valores ou níveis e selecionando os que mais impactam no desempenho;
6. Selecionar a **Técnica de Avaliação** de desempenho apropriada para o estudo. Esta divida em três técnicas: modelagem analítica, simulação/emulação e medição de um sistema real;
7. Selecionar a **Carga de Trabalho** que consiste em uma lista de solicitações de serviço ao sistema. Para modelagem analítica normalmente expressa como uma probabilidade de várias solicitações. Para simulação solicitações medidas em um sistema real. É essencial que ela seja representativa na vida real.
8. Definir o **Experimento** do Projeto em termos de sequência, oferecendo o máximo de informações com o mínimo de esforço;
9. **Analisar e Interpretar os Dados**, reconhecendo que os resultados das medições e simulações, contam com variabilidade dos resultados. Assim sendo faz-se necessário o uso de técnicas estatísticas para comparar alternativas. A interpretação dos

resultados produzem resultados e não conclusões. Cabe ao analista a conclusão, que pode ser diferente de um analista para outro;

10. **Apresentação dos Resultados** é a etapa final da avaliação de desempenho e deve ser comunicada de forma a facilitar a compreensão para tomada de decisão, sem jargão estatístico, preferencialmente em gráficos.

O Objetivo deste projeto é vencer o desafio do melhor o limiar que separa o tráfego legítimo do espúrio pela ausência de um limiar dinâmico que melhore a eficiência dos algoritmos baseados em entropia. Aqui exploramos a utilização de ML para o cálculo dinâmico dos limiares de algoritmos que utilizam entropia e apresentamos um método híbrido utilizando entropia e ML o ML-Entropia e também três alternativas híbridas de ML (SVR híbrido, SVC híbrido e *Random Forest* híbrido). O Sistema é definido como Redes SDN, para implementação da solução, fora dos planos de dados e de controle, a fim de não impactar no desempenho geral da rede. Com isto, até o ponto da detecção, esta proposta de solução também é aplicável às Redes Tradicionais. Como diferencial, evoluindo para a mitigação, ela já é habilitada para SDN, permitindo explorar todo seu potencial, incluindo implementação em qualquer um de seus planos. Os limites do Sistema são definidos juntamente com os Parâmetros mais a frente. Como Técnica de Avaliação duas foram escolhidas a Emulação, que serviu como Modelo de viabilização da proposta, posteriormente um Ambiente Real e de produção, para validação da proposta.

Definição dos Serviços no Modelo, considerados mais relevantes, foram: Serviços muito comuns disponibilizados para acesso legítimo nas portas 8080, 8443 e 8025, que representam os serviços mais utilizados no ambiente real e adicionado 8 ou 80 apenas para diferenciar de forma clara nas análises o tráfego sintético do tráfego real. Reprodução da anomalia do serviço ou seja do Ataque no Modelo com possibilidade de Aferição: Para o ataque DDoS volumétrico foi utilizada a ferramenta T50 e os serviços disponibilizados no Modelo Emulado são as portas 8011, 8012, 8013 escolhidas para diferenciar das portas de tráfego normal e assim permitir a identificação do tráfego de ataque e legítimo na criação da coluna de classificação e assim permitir o método supervisionado em *Machine Learning*. Foram explorados no ataque tanto o protocolo UDP quanto TCP nos alvos para testar a capacidade da solução proposta de identificar ataques independente do tipo de protocolo explorado.

Os Serviços mais relevantes no Mundo Real não precisam ser definidos, pois se trata do mundo real, contudo para preservar a disponibilidade dos serviços reais e de produção desse ambiente, foram implementados servidores específicos para receberem os ataques e tráfegos legítimos foram gerados nestes para acesso legítimo nas portas 8050, 8051 e 8052, que foram escolhidas para diferenciar das portas de ataque e permitir sua identificação na criação da coluna classe no método supervisionado, conforme já comentado. São gerados

tráfegos legítimos aleatórios nos servidores alvo, que se misturam ao tráfego real, para evitar que somente tráfego de ataques chegassem até estes servidores e assim não provocasse um aprendizado equivocado nos algoritmos de ML. Para a reprodução do Ataque, também utilizando T50, no Mundo Real com possibilidade de Aferição: Os serviços disponibilizados no Ambiente Real para ataque DDoS volumétrico foram as portas 80, 443, 25, 8080, 143, 22, 9200, 8443 e 2575 escolhidas por representar os serviços mais utilizados no ambiente real tanto para produção quanto desenvolvimento de sistemas, misturando-se com os fluxos do ambiente real que também possui acesso a estas mesmas portas nos servidores de aplicação reais da corporação, assim garantimos que não haverá viés que interfira nos resultados, permitindo uma comparação justa. No ambiente real o uso destas portas permite a identificação do tráfego de ataque e normal para aplicação do método supervisionado em ML uma vez que os servidores alvos são disponibilizados em ambiente real para este fim, assim a identificação passa a ser pela porta explorada e o IP vítima juntos.

As Métricas definidas para aferir o desempenho da solução proposta são as mais relevantes para este tipo de pesquisa, onde de acordo com a Matriz de Confusão gerada a partir do resultado da detecção, conforme Figura 5.1, são apresentados os erros e acertos, contendo no eixo X os valores preditos pelo mecanismo de detecção e no eixo Y os valores reais ou verdadeiros que são dados a partir da coluna classe criada para aplicação do método supervisionado que informa o valor verdadeiro. Essas métricas são listadas e explicadas a seguir.

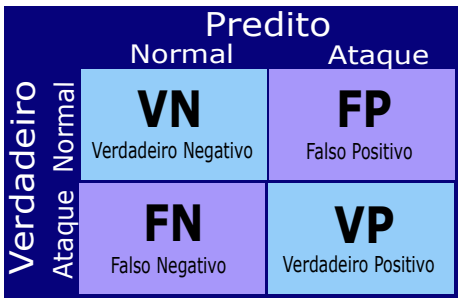

Figura 5.1: Modelo da Matriz de Confusão.

Acurácia: Indica quantas classificações o modelo classificou corretamente, entre todas os Verdadeiros Positivos (VP) e Verdadeiros Negativos (VN) e é dada pela seguinte Equação 5.1.

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN} \tag{5.1}$$

Taxa de Erro: Tem o propósito inverso ao da acurácia, onde ele serve para medir o quanto a solução errou entre todos os erros e acertos e é dada pela seguinte Equação 5.2.

$$TaxaDeErro = \frac{FP + FN}{VP + VN + FP + FN} \tag{5.2}$$

Precisão: Considera as classes Positivas e quantas classificações do modelo estão corretas. Utilizado onde os Falsos Positivos (FP) são considerados mais nocivos do que os Falsos Negativos (FN) e é dada pela seguinte Equação 5.3

$$Precisão = \frac{VP}{VP + FP}. \tag{5.3}$$

Recall/Revocação ou Sensibilidade: Os três termos podem ser encontrados na literatura e serve para medir dentre todas as situações de classe Positivo como valor esperado, quantas estão corretas. Seguiremos nos referindo a esta como *Recall*. A métrica *Recall* tem o propósito inverso ao da precisão e pode ser usada em uma situação em que os Falsos Negativos (FN) são considerados mais nocivos que os Falsos Positivos (FP) e é dada pela seguinte Equação 5.4.

$$Recall = \frac{VP}{VP + FN} \tag{5.4}$$

F1-Score: Essa métrica é a média harmônica entre Precisão e *Recall*. É uma maneira de observar somente uma métrica ao invés de duas (precisão e recall) e é dada pela seguinte Equação 5.5.

$$F1\text{-}Score = 2 * \frac{Precisão * Recall}{Precisão + Recall} \tag{5.5}$$

Além dessas métricas descritas que servem de comparação entre as diversas técnicas apresentadas, também comparamos com três trabalhos semelhantes da literatura.

Os Parâmetros definidos foram com relação a topologia da rede e capacidades que serão encontradas no ambiente real a fim de recriar, em escala menor, no ambiente Emulado um Modelo que melhor represente o Ambiente Real para implementação, assim sendo obter resultados que sejam realmente aplicáveis em condições reais. O ambiente real que serviu de implementação da proposta de solução aqui discutida, é composto de vários segmentos de rede tais como: segmento de clientes, segmento de servidores de produção, segmento DMZ que faz fronteira com a Internet, segmento de fornecedores e outros, concentramos e definimos o escopo de implementação o "Segmento de Servidores de Produção" onde ficam os Servidores de aplicação, pelas seguintes razões:

- É o ambiente mais sensível de toda a corporação, onde são disponibilizados os principais serviços;

- Todos os usuários internos da rede têm acesso aos serviços lá hospedados, portanto o tráfego entre cliente e servidor transitam no barramento;
- Os Serviços disponibilizados aos clientes externos são oriundos desse barramento que fazem a replicação a partir dele, permitindo a captura e análise desses dados, bem como permite a percepção de possíveis brechas, não percebidas pelos dispositivos de segurança perimetral existentes, que possibilitem um atacante ganhar acesso ao segmento de produção e promover o ataque;
- A solução proposta apesar de definir como escopo este segmento de produção em sua implementação e análise de viabilidade, permite que o mecanismo híbrido de detecção de ataque DDoS seja replicável aos demais segmentos que a organização julgar pertinente.

Dessa forma foi definida como topologia para o ambiente emulado uma rede contendo um segmento de servidores de produção e outro segmento de clientes com as seguintes capacidades:

- 06 Servidores de aplicação no ambiente emulado;
- 124 Servidores de aplicação no ambiente real;
- 40 clientes legítimos acessando serviços disponibilizados no ambiente emulado;
- Aproximadamente 2000 clientes legítimos acessando serviços disponibilizados no ambiente real;
- 03 Servidores alvo que recebem ataque e tráfego normal (mesma quantidade para ambiente emulado e real);
- 04 atacantes explorando 01 porta em cada servidor e os protocolos UDP e TCP, com *spoofing* no ambiente emulado;
- 03 atacantes explorando 03 portas em cada servidor e os protocolos UDP e TCP, com *spoofing* no ambiente real. O ajuste foi feito para se adequar as diferenças de capacidades dos ambientes e garantir o balanceamento dos dados, assim também garantimos que não haverá tendência do Modelo emulado nos resultados do ambiente real.

A escala reduzida, conforme demonstrado na Figura 5.2, ajuda na observação dos comportamentos e variações existentes, contudo é fundamental que se encontrem presentes no modelo emulado as variáveis mais relevantes e que interferem no resultado do projeto de pesquisa. Vale ressaltar também que é importante evitar viés nos dados para garantir

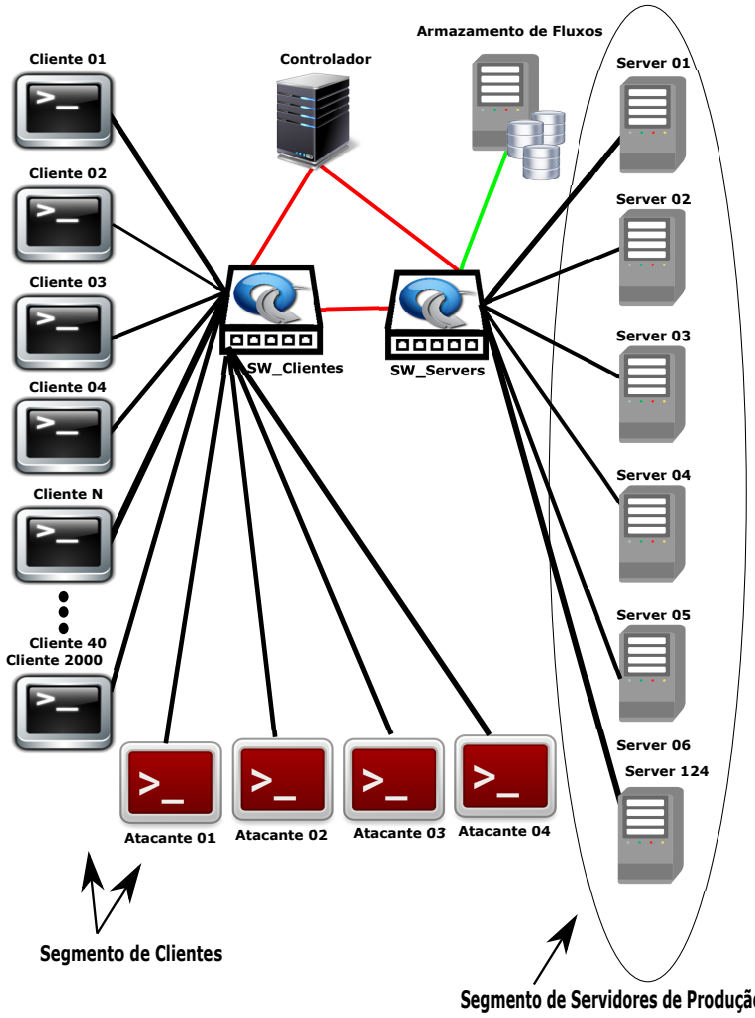

Figura 5.2: Rede SDN Representando Ambiente Modelo e Real.

a generalização do modelo e sua aplicabilidade no mundo real, evitando com isso possíveis *Overfitting e Underfitting*.

Como Fatores foram definidos três tamanhos de janelas por fluxo e mais três tamanhos de janelas por tempo diferentes para aferição dos resultados. Como limitador do escopo de fatores foi definido como regra a escolha das janelas seguindo da menor janela viável para a maior janela de melhor desempenho ou início da degradação dos resultados em relação às janelas anteriores. Totalizaram doze janelas, sendo três janelas por fluxo ambiente emulado e três do ambiente real, somadas a mais três janelas por tempo do ambiente emulado e três do ambiente real. O ataque é caracterizado pelo tipo DDoS, volumétrico, com *Spoofing*, tendo como vítima as aplicações e impacto a degradação de desempenho dos servidores alvo e não a sua paralisação. Para garantir isso os servidores foram monitorados durante o ataque, que foi calibrado até atingir entre 90% a 95% de utilização da máquina, entre tráfego legítimo e espúrio, para evitar a paralisação dos serviços, tanto no Modelo quanto no Real. O fluxo estabelecido no ataque é intermitente, no ambiente Emulado, de

30 em 30 segundos, perfazendo 5 ciclos, devido a sua capacidade reduzida a fim de evitar o efeito indesejado de *Overfitting, Underfitting* e desbalanceamento dos dados coletados. Ressaltamos que testamos o fluxo constante neste ambiente e o efeito indesejado ocorreu, por esse motivo o fluxo intermitente foi escolhido para o ambiente emulado. O fluxo foi constante para o ambiente Real devido às suas capacidades serem muito maiores e para se aproximar ao máximo das condições reais desse tipo de ataque sem paralisação do serviço. Foram utilizados quatro atacantes para o ambiente emulado explorando uma porta em cada servidor alvo, enquanto são utilizados três atacantes no ambiente Real, com um deles posicionado via VPN, todos explorando três portas em cada servidor alvo, a fim de produzir magnitudes diferentes compatíveis com o tamanho do ambiente utilizado (emulado ou real). Foram explorados os protocolos UDP e TCP no ataque para testar a capacidade de detecção independente do tipo de protocolo utilizado.

A Carga de Trabalho no ambiente emulado foi gerada utilizando o iPerf3 com a escrita de um programa em Python3 para disparar os acessos de forma aleatória aos 06 servidores. No ambiente real foram coletados os dados reais do segmento de servidores de produção, contendo todos os IP de Origem que alcançam o segmento, incluindo segmentos adjacentes.

O Experimento, a Apresentação dos Resultados e a Análise e Interpretação dos Dados estão descritos no Capítulo 6, Seções 6.1, 6.2 e 6.3, respectivamente. Foram comparados resultados do mecanismo híbrido proposto com três trabalhos na literatura relacionados e também comparados com os resultados do próprio experimento, considerando o seguinte:

1. Entropia = Abordagem tradicional;
2. ML-Entropia (Recebe coeficiente da regressão) = Abordagem híbrida;
3. SVR (Regressão) = Abordagem híbrida;
4. SVC (Classificação) = Abordagem híbrida;
5. Random Forest (Classificação) = Abordagem híbrida.

Capítulo 6

Experimento, Resultados e Análise

6.1 Experimento

Esta seção foi subdividida em "Experimento no Ambiente Emulado" (Subseção 6.1.1) e "Experimento no Ambiente Real" (Subseção 6.1.2). Nelas foram detalhadas as etapas e configurações necessárias para a viabilização da proposta em cada um dos ambientes. O detalhamento dos aspectos gerais e as definições feitas que valem para os dois ambientes, Emulado e Real, estão aqui nesta inicial. Este experimento foi conduzindo conforme metodologia detalhada no Capítulo 5. Iniciou-se com a montagem do Ambiente Emulado para construção do Modelo e comprovação da viabilidade no mundo real. Posteriormente, após a obtenção dos resultados preliminares, contendo a comprovação da viabilidade, as técnicas e etapas, que se mostraram mais viáveis, foram replicadas em Ambiente Real corporativo.

A prototipação com a construção Modelo garantiu, em ambiente reduzido e seguro, com menor custo, maior agilidade e controle, o desenvolvimento da solução proposta, possibilitando vencer os principais desafios que se apresentam no mundo real. Neste experimento o Modelo também permitiu melhor entender comportamentos, devido a escala reduzida, que seriam mais difíceis de entender no mundo real dado as múltiplas variáveis existentes no Ambiente Real. Contudo é importante entender que o fato do Modelo se apresentar viável na emulação não é garantia de que ele terá o mesmo resultado, quando aplicado no mundo real. Isto por que novos desafios podem se apresentar, devido a variáveis que não são relevantes no contexto do escopo da solução, mas que em grau menor podem impor ao pesquisador encontrar soluções de contorno de problema que permitam se manter na mesma linha mestra da solução viável apresentada no Modelo. Estes desafios ocorreram, foram vencidos e são melhor explanados e detalhados na Subseção 6.1.2.

As janelas foram definidas por fluxos e janelas por tempo de três tamanhos diferentes para o Ambiente Real Corporativo e Emulado cada uma, com seis janelas para cada

ambiente, totalizando doze janelas diferentes. Para defini-las seguiu-se com testes da menor para a maior janela viável, com três graus de tamanhos, observando a crescente dos bons resultados até que a janela máxima atingisse um resultado máximo possível ou iniciasse uma degradação de desempenho na detecção de ataque em relação as janelas anteriores. Somente as três melhores janelas de cada *Dataset* (Real e Emulado) para Entropia tradicional, ML-Entropia, SVR híbrido, SVC híbrido e *Random Forest* híbrido são apresentadas, comparadas e comentadas na Subseção 6.2.

Enfatiza-se que os atributos mais relevantes foram apontados pelos algoritmos, após a limpeza e tratamento dos dados, de acordo com a exploração dos dados e o resultado de suas correlações mais fortes ou mais fracas na previsão desejada e não simplesmente escolhidos pelo pesquisador de forma aleatória. Portanto, a chave neste caso é "explorar os dados" apoiado pelas ferramentas apropriadas. Para a seleção dos melhores atributos que serviram de limiar para a detecção do ataque em Entropia e ML-Entropia foram consideradas algumas regras predefinidas para garantir comparações justas e sem tendência, foram elas:

- Os Limiares selecionados deveriam estar contidos no arquivo gerado no passo 03A do diagrama de fluxo na Figura 4.2, não sendo permitido criar novos a partir do que somente uma das janelas permite (fluxos ou tempo), a fim de manter a igualdade o mais próximo possível, de modo a não comprometer as comparações;

- O melhor Limiar-X com os melhores resultados de detecção para o conjunto de dados com e sem ataque foram buscados a partir da técnica de Entropia tradicional e quando selecionados estes deveriam ser os mesmos usados em ML-Entropia com a mesma lógica de comparação para garantir a precisão da comparação e assim medir sua efetividade. Ex.: Na comparação "(λ) **= Se (**$\overline{E}\_IPDest > T$**) Então**", a lógica foi a mesma para ambos;

- Foi permitido fazer ajustes de magnitude do Limiar-X em relação ao conjunto de dados, uma vez que eles não interferem na comparação dos resultados. Contudo sem alterar a estrutura do λ e os limiares selecionados. Esta regra está alinhada à disciplina de ML que preconiza a observação dos dados e o que eles nos informam sobre os resultados. Por exemplo: Onde $\lambda =$
  Se ($\overline{E}\_IPDest > T$**/2) Então** (Para Entropia) e
  Se ($\overline{E}\_IPDest > T$***2) Então** (Para ML-Entropia)
  OBS.: Permaneceram preservados os Limiares-X e o λ (a diferença se limita na magnitude **(/2 e *2)**.
  Comentário: Os ajustes permitidos foram justificados uma vez que a busca por um melhor posicionamento do hiperplano na separação de dados segue lógicas diferentes,

um pode ser posicionado acima dos dados alvo e o outro abaixo dos mesmos dados alvo, adequando-se assim à interpretação do pesquisador no ajuste para cima ou para baixo do Limiar, porém permanece o mesmo Limiar-X.

- Os Limiares definidos e seus respectivos ajustes, quando necessário, para um determinado ambiente e seu *Dataset* (Emulado ou Real) deveriam ser os mesmos para dados com Ataque e sem Ataque.

Os Limiares-X foram os mesmos para os *Datasets*, com e sem Ataque, no ambiente Emulado. Também foram os mesmos para os *Datasets*, com e sem Ataque, no ambiente Real. Com relação a este ponto ressalta-se, para que não haja erro de entendimento, que existe a necessidade de se analisar e entender os *Datasets* dos diferentes ambientes e se definir seus respectivos Limiares-X. Assim sendo a análise feita para o conjunto de dados do ambiente Emulado deve ser repetida para o ambiente Real. Isto por que estamos lidando com ambientes diferentes, com dimensões diferentes e no caso do ambiente real com uma quantidade maior de variáveis, onde foram replicadas em emulação as principais características do mundo real e não todas as existentes, somente as relevantes.

Para se obter uma boa separação dos dados é importante que o limiar siga o contorno de separação conforme eles se apresentam no *Dataset*. Para melhor entender isto, conforme exemplo da Figura 6.1, a imagem mostra dois planos, linear à esquerda e não linear à direita, onde a linha azul é o hiperplano traçado na separação dos dados. Nem sempre a melhor separação seguirá uma linha reta dentro do conjunto de dados, às vezes pode ser necessário fazer uma curva e minimizar a margem de erro na separação com uma função *kernel* [47]. Para isso alguns algoritmos de regressão não lineares resolvem o problema.

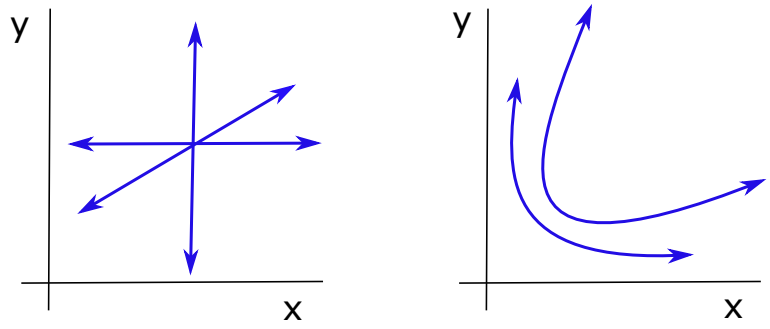

Figura 6.1: Hiperplano de Separação dos Dados Linear (esquerda) e Não Linear (direita).

O*Support Vector Regression (SVR)* com a função *kernel* RBF é um algoritmo de regressão não linear que pode ser usado para alcançar um limiar mais apropriado na implementação desta proposta, conforme demonstrado na Figura 6.2 que apresenta suas características em dados deste experimento. Na imagem vemos os pontos, denominados Vetores de Suporte (do inglês - *Support Vectors*), que servem de guia para Margem Má-

xima, que são as margens traçadas para delimitar a fronteira de cada grupo de dados, azuis são dados normais e vermelhos são dados do ataque. O hiperplano é traçado tomando como base o ponto médio entre as margens e ele representa um coeficiente de previsão.

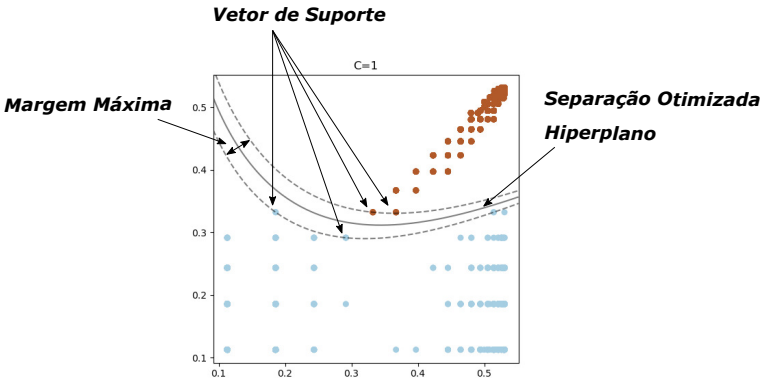

Figura 6.2: *Support Vectors* Definido nos Dados do Mininet (janela = 50 Fluxos). De acordo com [3].

O aumento da eficiência na detecção de ataques é demonstrado em outra amostra deste experimento, na Figura 6.3, na qual é possível ver as duas equações que determinam as linhas da Margem Máxima, bem como a equação que determina o traçado do hiperplano. Além disso temos representado o ξ_i que é a distância do ponto classificado errado até a margem correta, conforme já explanado na Seção 2.6, ele representa a margem de erro (FP e FN). Outra vantagem do RBF também conhecido como *LS-SVM (Least Squares Support Vector Machine)* é que ele é computacionalmente mais eficiente do que o SVM comum [50].

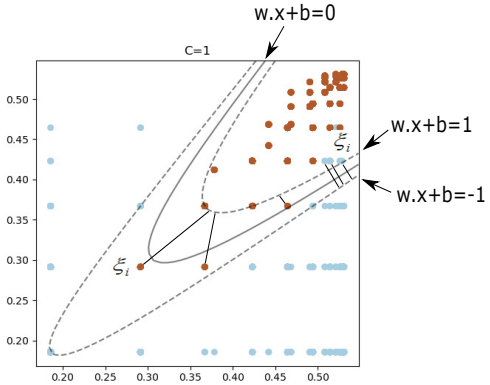

Figura 6.3: SVR com *Kernel* RBF na Predição de Ataque nos Dados do Mininet (janela = 25 Fluxos). Baseado em [4] e [5].

A abordagem utilizando regressão não linear ajuda na previsão dos ataques a acompanhar as curvas no conjunto de dados de forma a minimizar a taxa de erro, conforme pôde ser percebido nas Figuras 6.2 e 6.3 apresentadas. Esta característica foi explorada para

beneficiar a Entropia na abordagem ML-Entropia usando o coeficiente de previsão do hiperplano com um limiar mais adequado para melhorar a separação dos dados legítimos e espúrios. O algoritmo usado para implementação da solução, aplicado tanto no Ambiente Emulado quanto no Ambiente Real, está descrito nos pseudocódigos dos Algoritmos 1 e 2.

Algorithm 1 Attack Detection using only Entropy

Require: A = set of attributes; n = number of flows; $M = \emptyset$;

1: **for** $i = 1$ to n **do**
2: Calculate E_i using Eq.(4.1) and E_{SD} Eq.(4.3), respectively, $\forall x | x \in A$;
3: $M \leftarrow M \cup E_i \cup E_{SD}$;
4: **end for**
5: Compute $\overline{E}$, E_{Max}, and E_{Min} using Eqs. (4.2, 4.4, 4.5) respectively, $\forall x | x \in A$;
6: $M \leftarrow M \cup \overline{E} \cup E_{Max} \cup E_{Min}$;
7: **Output** M to Entropy-File;
8: Let $T = \{t_1, t_2, \cdots, t_m\}$, $m > 0$, where $T \subset M$ holds the best m values in M;
9: **for** $i = 1$ to n **do**
10: **if** $(E_i \otimes t_1 \bigodot E_i \otimes t_2 \bigodot, \cdots, \bigodot E_i \otimes t_m)$ **then**
11: **return** (Attack);
12: **else**
13: **return** (Normal Traffic);
14: **end if**
15: **end for**

O Mecanismo de Entropia é detalhado no Alg. 1. O algoritmo recebe como entrada um conjunto de atributos (A), tais como endereço IP, portas, freqüência de fluxo, etc. Estes atributos são coletados de diferentes tamanhos de janela, coletados ao longo do tempo. No primeiro for-loop, os E_i e E_{SD}, para cada atributo $x \in A$, são computados usando Eq. (4.1) e (4.3), respectivamente. Em seguida, $\overline{E}$, E_{Max} e E_{Min} são computados usando Eq. (4.2), (4.4), (4.5), respectivamente, para cada $x \in A$. Os valores computados são armazenados no conjunto M, $M = M \cup E_i \cup E_{SD} \cup \overline{E} \cup E_{Max} \cup E_{Min}$. O conjunto M é armazenado em "Entropy-File" para entrada no Alg. 2. Os valores em M são selecionados com base na análise empírica. Os melhores valores de m em M são atribuídos a T a partir de $|A| \times n$ limiares possíveis que melhor caracterizam um ataque. A detecção de ataque é realizada no segundo for-loop do Alg. 1, onde cada E_i, $1 \leq i \leq n$, é avaliado com cada elemento em T. Os operadores $\otimes$ e $\bigodot$ representam uma operação relacional utilizada para determinar se E_i representa um fluxo legítimo ou um fluxo espúrio. Neste último caso, um ataque é relatado.

Algorithm 2 Attack Detection using ML-Entropy

Require: EF; $n \leftarrow$ number of flows; δ; $LastCall$; $TimeOut$;

1: **Function:** ComputeCoefficients($\delta, EF, LastCall, TimeOut$)
2: **if** $\delta == \emptyset \vee LastCall > TimeOut$ **then**
3: $\delta \leftarrow \delta \cup SVR[RBF](EF)$;
4: Call ML-Entropy (δ);
5: **else**
6: Call ML-Entropy (δ);
7: **end if**
8: **Output** δ;
9: **End Function**
10: **Function:** ML-Entropy (δ)
11: Let $T^i = \{t_1, t_2, \cdots, t_m\}$, $m > 0$, where $T^i \subset \delta$ holds the best m values in δ;
12: **for** $i = 1$ to n **do**
13: **if** $(E_i \otimes t_1 \bigodot E_i \otimes t_2 \bigodot, \cdots, \bigodot E_i \otimes t_m)$ **then**
14: **return** (Attack);
15: **else**
16: **return** (Normal Traffic);
17: **end if**
18: **end for**
19: **End Function**
20: Call ComputeCoefficients($\delta, EF, LastCall, TimeOut$);

Os detalhes do mecanismo ML-Entropia são fornecidos no Alg. 2. Como entrada, o algoritmo recebe os seguintes parâmetros: "Entropy-File" (EF), vindo do Alg.1, o conjunto δ, que é vazio na primeira chamada; $LastCall$ que é o tempo desde o último cálculo e; o $Timeout$ que define a quantidade mínima de tempo para realizar o novo treino/teste. A função "ComputeCoefficients"recebe estes parâmetros que são usados para atualizar os valores em δ. Na primeira chamada, ou depois de um $TimeOut$, o SVR-RBF é chamado. A função fornece os coeficientes de previsão com base nos valores em EF para o treino/teste. Os resultados são armazenados em δ. Depois, a função ML-Entropia é chamada. A função ML-Entropia recebe como parâmetro o coeficiente em δ que detém os pesos de m do subconjunto de A encontrado pelo ajuste do hiperplano. Os melhores valores em δ são atribuídos a T^i contendo pelo menos um atributo do subconjunto. Observe o teste realizado no (Alg. 1, Linha 10) e (Alg. 2, Linha 13) essa igualdade tem como objetivo fornecer uma comparação justa entre as duas abordagens (Entropia e ML-Entropy). Assim, é possível aferir o desempenho da proposta ML-Entropia com Entropia

tradicional, já que a diferença entre elas se baseia apenas nos pesos T (Alg. 1, Linha 8) e T^i (Alg. 2, Linha 11). No segundo for-loop do Alg. 2 a detecção do ataque é realizada por ML-Entropia, seguindo a mesma avaliação com cada elemento em T (Alg. 1, Linha 10) só que agora em T^i (Alg. 2, Linha 13).

Para os algoritmos SVR, SVC, *Random Forest* o arquivo EF serve também de entrada para a realização do treino/teste, onde os algoritmos de classificação aqui utilizados realizaram *Cross Validation (CV)* = 5 para o *Dataset* do ambiente Real. Para o *Dataset* do ambiente Emulado foi aplicado o método *K-Fold* onde foram feitos *Cross Validation (CV)* = 5 para 10 Modelos diferentes e calculada a média de todos, ou seja, a média de 50 repetições. O tamanho do *Dataset* do ambiente Real inviabilizou a aplicação deste método nele, pois um único teste aplicado nesse conjunto de dados demorou 12 horas de processamento, optamos então pelo CV=5 nesse *Dataset*. Para o algoritmo de regressão todo o *Dataset* foi utilizado para Real e Emulado. Com o melhor Modelo encontrado cada algoritmo realizou a detecção do ataque, todos gerando uma matriz de confusão para aferição do resultados e comparação, que são apresentados na Seção 6.2. O longo tempo relatado só ocorre durante o treino/teste dos algoritmos de classificação, mas uma vez encontrado o Modelo ideal a detecção de ataque é rápida.

6.1.1 Experimento no Ambiente Emulado

A construção do Ambiente Emulado (Modelo) foi feita utilizando um servidor com CPU i7 de 2,9 GHz, 16 GB de RAM, 256 GB de SSD com Ubuntu versão 18, onde foi selecionado como ferramenta de emulação o MiniNet, por ser a solução mais utilizada em pesquisas desse tipo, na versão 2.2.2-170321 para a emulação SDN, Ryu como controlador, *Openflow Protocol* 1.3 e protocolo *Netflow* para coleta de dados, em uma topologia com quarenta *hosts*, quatro atacantes e seis servidores de aplicação, sendo três como alvos de ataque DDoS. O controlador foi conectado em dois *Switches OVS (Open Virtual Switch)*. Os fluxos foram coletados no plano de dados, a ferramenta escolhida para este fim foi o Graylog na versão gratuita 3.2.6 [61] pela facilidade de implementação, pois já existe a compatibilidade com SDN, permitindo a coleta de fluxos através do protocolo *Netflow*, com armazenamento no banco de dados MongoDB que é um banco de dados gratuito orientado a documentos de código aberto, usa formato de documento como JSON, e Elastic Search que é um mecanismo de pesquisa de código aberto baseado no Lucene e desenvolvido em JAVA [62]. O servidor para coleta e armazenamento de dados foi conectado ao *Switch* de Servidores e a coleta e análise de dados sem intervir no desempenho geral da rede segue a topologia que é representada na Figura 6.4.

O tráfego aleatório normal foi gerado utilizando o iPerf versão 3 com tamanhos de pacotes aleatórios (de 150 a 1500) e intervalos de tempo aleatórios (de 1 a 5 segundos),

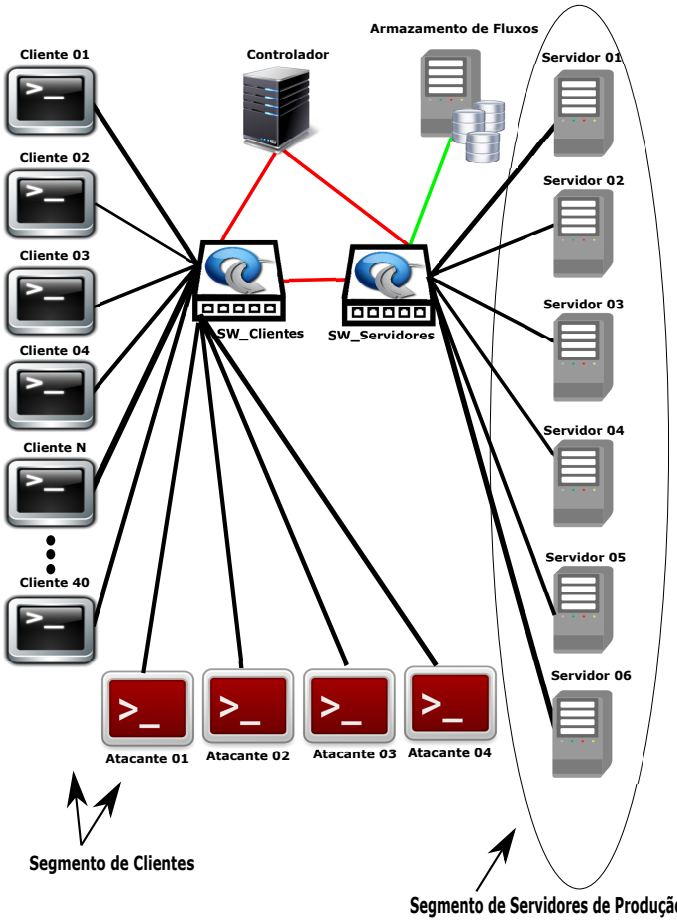

Figura 6.4: Topologia SDN para o experimento.

cada cliente gerou pedidos aleatoriamente para seis servidores, através do uso de *scripts* instanciados por um código escrito em Python 3. As portas utilizadas para acesso legítimo foram 8080, 8443, 8025 que representam serviços legítimos mais comuns nas redes reais (80, 443 e 25), contudo adicionados 8 ou 80 antes de cada número para distinguir, na análise e por cautela, os dados coletados do MiniNet dos dados coletados no ambiente real, e também diferenciadas das portas de ataque para permitir a identificação do tráfego de ataque e legítimo na criação da coluna de classificação e assim permitir o método supervisionado em *Machine Learning*. Quarenta clientes fazem acessos a todos os seis servidores, inclusive os três alvos para evitar viés nos dados.

O ataque DDoS que foi realizado é volumétrico com *spoofing*, conforme ilustrado na Figura 6.5, utilizando quatro atacantes com o T50 versão 5.8.7 [63], testando uma frequência de solicitações próximo à ocorrência de paralisação, mas sem a interrupção do serviço durante a medição com carga de trabalho no servidor alvo entre 90% a 95%, entre dados legítimos e espúrios. As portas exploradas são 8011, 8012 e 8013, com dois *hosts* explorando o protocolo UDP e outros dois explorando o protocolo TCP para testar a capacidade da solução proposta de identificar ataques independente do tipo de protocolo

explorado. Os dados que contêm apenas tráfego normal e depois com ataque são coletados, para treinamento e criação dos modelos de Machine Learning (ML) com 80% e 20% da amostra e Intervalo de Confiança de 95%. O RStudio versão 1.2.5042 - © 2009-2020, na exploração de dados e busca de padrões, devido ao seu desempenho e praticidade, onde foram aplicados para os seguintes algoritmos: Regressão Linear e Não Linear (*Support Vector Regression (SVR)*), *Support Vector Classifier (SVC)* e Random Forest. Posteriormente, exploradas as características e confirmado o balanceamento dos dados, os melhores atributos são selecionados e a implementação continua dessa vez utilizando o Python para a codificação do algoritmo com as bibliotecas mais apropriadas para o objetivo desejado, permitindo assim a automação do processo com os mesmos resultados diretamente no ambiente.

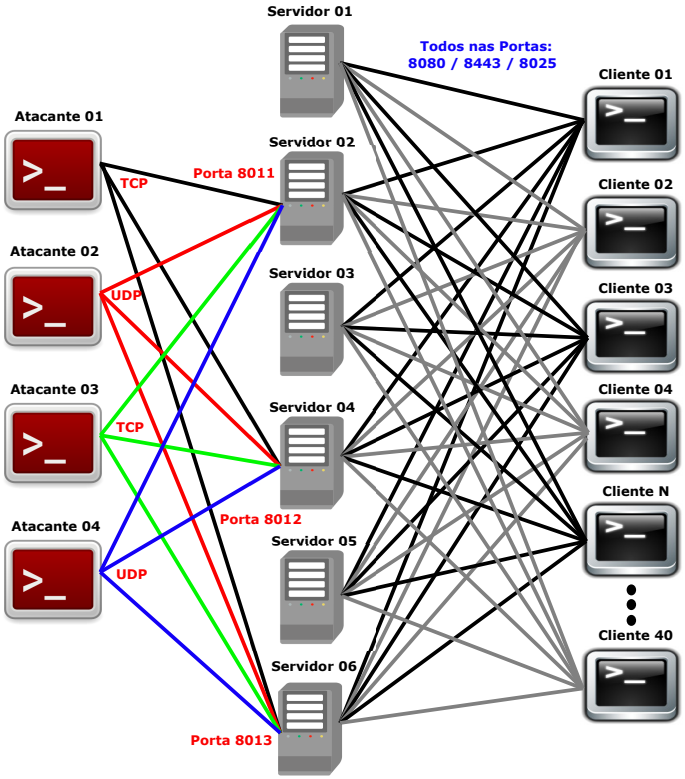

Figura 6.5: Topologia do Ataque no MiniNet.

O ataque foi realizado de forma intervalada de 30 segundos para balancear os dados e viabilizar melhor compreensão dos algoritmos do que é normal e anormal no fluxo de dados da rede, dessa forma são definidos os intervalos de 30 segundos apenas de tráfego normal - 30 segundos de ataque - 30 segundos apenas de tráfego normal - 30 segundos de ataque - 30 segundos apenas de tráfego normal. O ataque contínuo no ambiente emulado também foi realizado, contudo acreditamos que dada as capacidades menores do ambiente

houve incidência de *Overfitting e Underfitting*, onde a abordagem intervalada se mostrou mais apropriada para o aprendizado e generalização do Modelo.

As janelas foram definidas pelo tipo fluxos e tempo de três tamanhos diferentes para cada um dos tipos, totalizando seis janelas diferentes para este ambiente. As janelas por fluxos deste ambiente foram definidas em 25, 50 e 100 fluxos e as janelas por tempo foram de 1, 2 e 3 segundos.

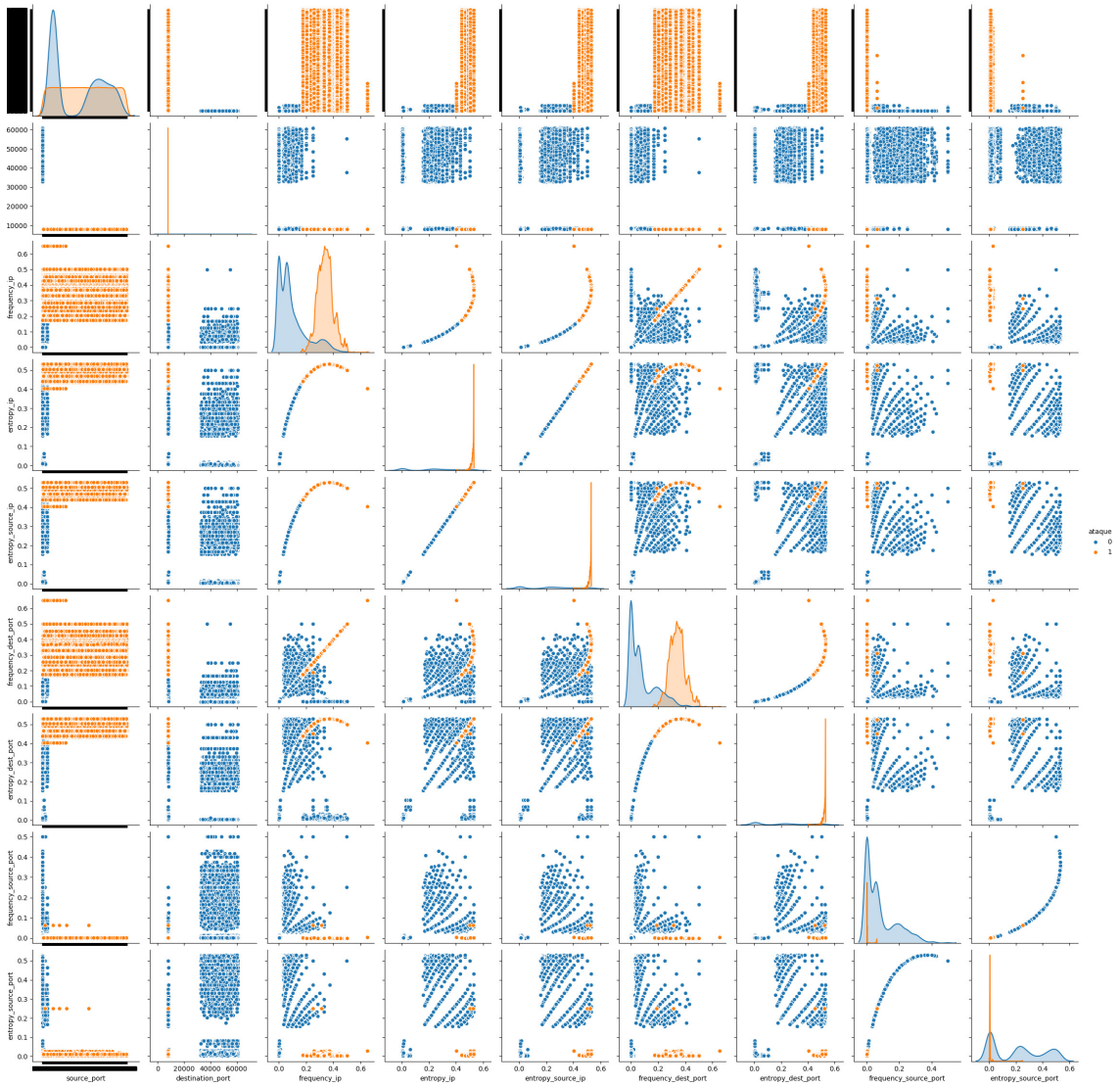

Figura 6.6: Comparação dos Atributos no *Dataset* do MiniNet.

Os atributos mais relevantes foram apontados pelos algoritmos, apoiados por uma ferramenta, neste caso pelo pacote "*Seaborn*" do Python apresentando a combinação de pares de atributos no conjunto de dados do MiniNet, de acordo com a Figura 6.6, onde os dados normais representados pelos pontos azuis e os de ataque pelos pontos laranja. Nos eixos X e Y estão os nomes dos atributos, do conjunto de dados, que foram comparados pelo algoritmo e apresentam suas combinações nas interseções, já nos cruzamentos dos

mesmos atributos, no eixo diagonal da Figura 6.6, estão os gráficos de linha de cada um desses pares de mesmo atributo. Diversos pares foram testados e aqui foram apresentados os pares que melhor separaram os dados e serviram de Limiares-X em Entropia e em ML-Entropia para detecção do ataque. Na Figura 6.7 os dados no MiniNet são ilustrados em 2D e 3D com os gráficos gerados com os limiares em 03 eixos para demonstrar a separação dos dados, onde os dados de ataque estão na cor amarela e dados normais na cor azul.

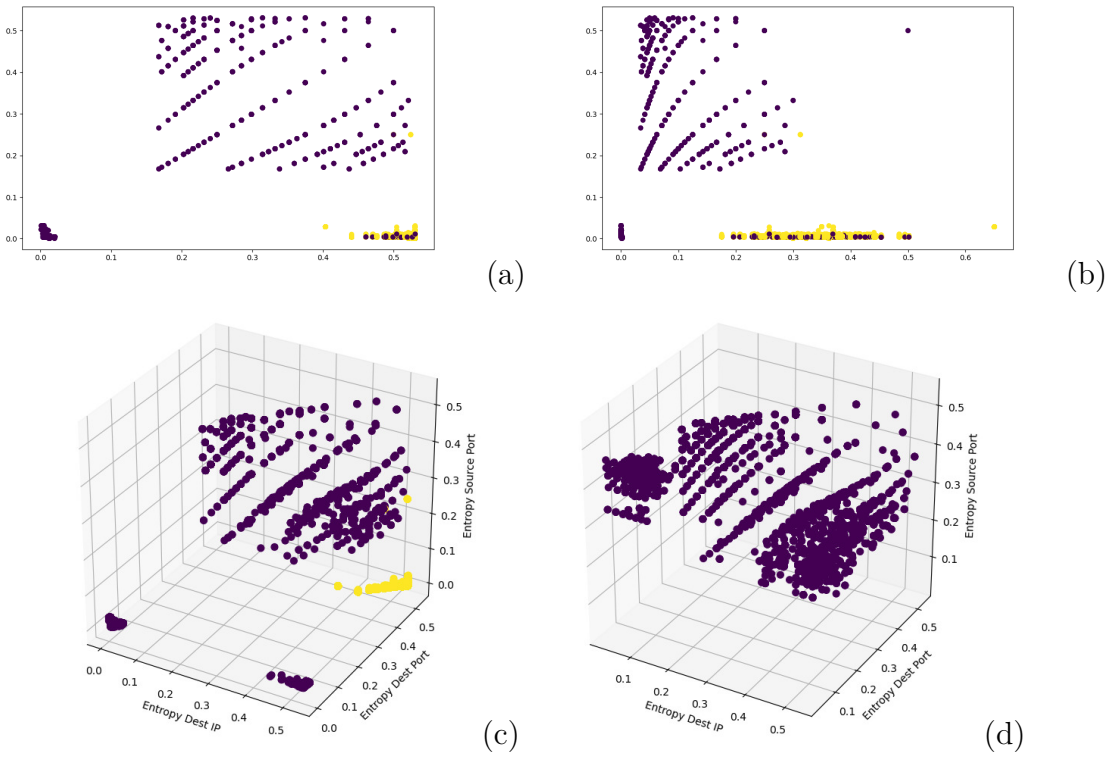

Figura 6.7: *Scatterplot* do MiniNet: Max Entropia IP de Dest. com Porta de Origem (a), Entropia Freq. do IP de Dest. com IP de Origem (b), *Scatterplot* dos 3 Eixos com Ataque (c) e *Scatterplot* dos 3 Eixos sem Ataque (d).

É possível perceber na Figura 6.8 que ilustra as detecções do algoritmo SVC em dados do ambiente emulado, com dados legítimos na cor azul e os dados de ataque na cor vermelha, para cada tamanho de janela por fluxo, as diferenças de cada *kernel* do algoritmo no conjunto de quatro quadros que apresentam os resultados do SVC com o *kernel* Linear, Linear com Ajuste, RBF e Polynomial, nota-se a linha branca que representa o hiperplano, encontrado pelo algoritmo, às vezes sobreposta pela alta densidade dos pontos, na separação dos dados. Os pontos azuis que se encontram na separação em vermelho fazem parte da margem de erro, assim como os pontos vermelho na separação azul, depois do hiperplano. Essa margem de erro que procuramos reduzir e que são obtidas por RBF e Polynomial, não lineares, conforme ilustrado.

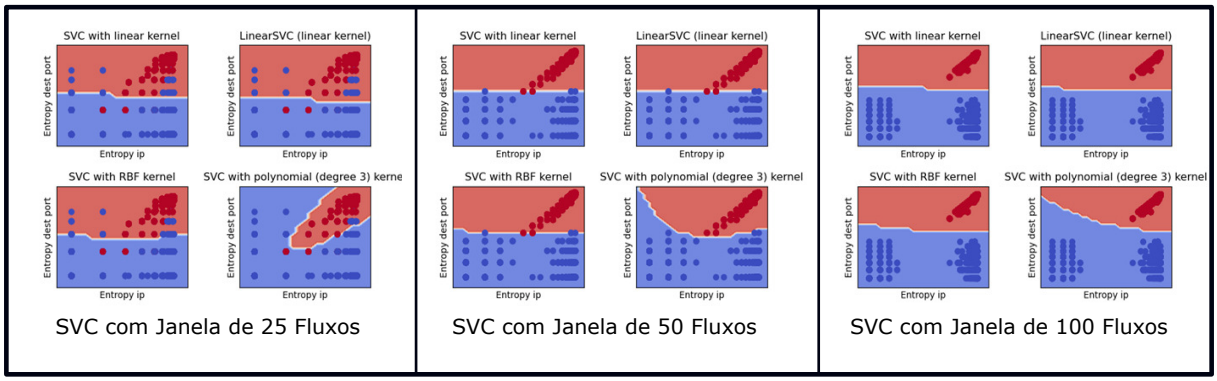

Figura 6.8: *Support Vector Classifier (SVC)* na Predição do Ataque nos Dados do Ambiente Emulado.

Observe, ainda na Figura 6.8, que à medida que a janela cresce em tamanho os dados vão ficando mais separados e melhor é a separação dos dados no hiperplano, até um tamanho que é ideal, como em 100 fluxos. Nos dados do MiniNet esta imagem é mais clara, contudo notamos no experimento que a partir de um determinado aumento de janela os dados ficam muito dispersos, acima de 100 fluxos, refletindo na detecção que começa a degradar suas métricas aferidas.

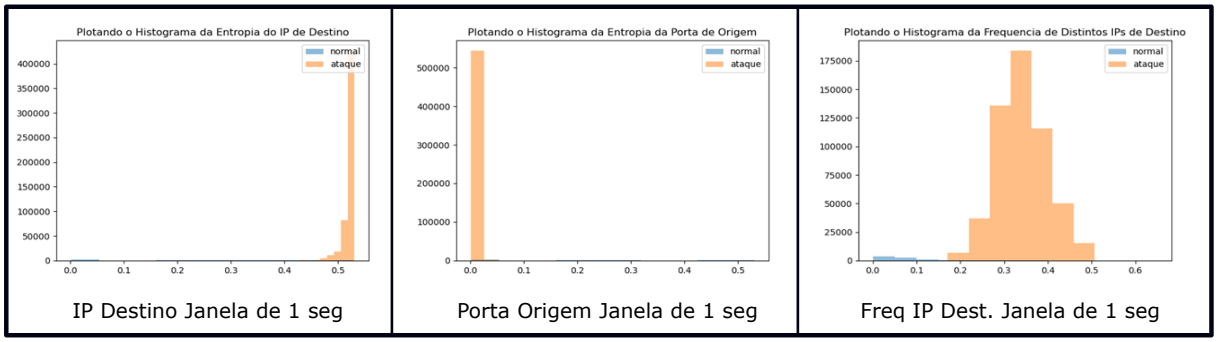

Figura 6.9: Histogramas com Janelas de 1 segundo para os Dados do Ambiente Emulado.

No processo de definição dos melhores limiares para uso na detecção de ataque DDoS por Entropia a análise foi feita apoiada nos gráficos gerados a partir da correlação apresentada na Figura 6.6 para o emulado (MiniNet) com a geração de histogramas, na Figura 6.9, dos pares que apresentaram melhor correlação de separação dos dados. Os candidatos a limiares foram plotados no histograma conforme ilustrado na Figura 6.10 onde os possíveis ajustes foram aplicados e posteriormente testados para aferição dos resultados até encontrar os melhor parâmetros de comparação para selecioná-los como Limiares-X na detecção. Isso foi primeiramente realizado nas janelas definidas por fluxo e depois por tempo. Para cada ambiente, emulado e real.

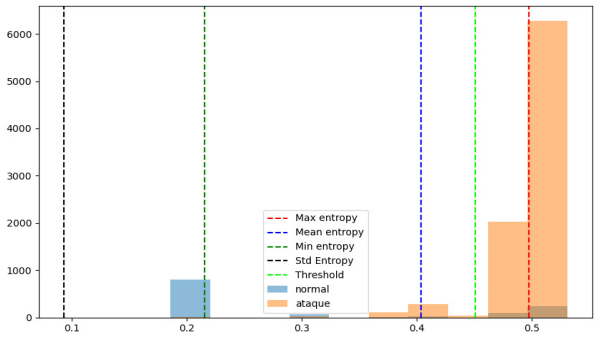

Figura 6.10: Histograma do IP de Destino nos Dados do Ambiente Emulado, Plotando as Entropias Calculadas.

6.1.2 Experimento no Ambiente Real

Todo o processo foi repetido no Ambiente Real, com um servidor para coleta de dados com 02 vCPUs, 32 GB RAM, 500 GB HD, Ubuntu versão 18 e três Servidores *Honeypotes*, foi utilizado o T-Pot 20.06 para Debian (Stable), disponível no GitHub link "https://github.com/dtag-dev-sec/tpotceconcept" [64], foram instalados com a seguinte configuração 02 vCPUs, 16 GB RAM, 200 GB HD, implementados no barramento dos servidores de aplicação real do ambiente de produção da empresa, que serviram como alvos dos ataques DDoS realizados, bem como receberam pedidos de tráfego normal para evitar o viés nos dados relacionados ao acesso a estes três servidores, que se misturaram na coleta de dados de outros em uma estrutura de virtualização de *cluster* com a seguinte configuração: Hypervisor = VMware ESXi, 6.7.0.7098360, Modelo = PowerEdge M640, Tipo de processador = Intel(R) Xeon(R) Gold 5118 CPU @2.30GHz, Processadores lógicos = 48, com cento e vinte e quatro servidores de aplicação e quase dois mil clientes que acessam estes servidores de aplicação legítimos.

A topologia ilustrada na Figura 6.11 contem apenas o escopo definido da implementação no ambiente real que foi reproduzido no emulado com a diferença das capacidades aqui apresentadas e representadas na ilustração para melhor entendimento. Na coleta dos dados no ambiente real via protocolo *Netflow* não foi possível a extração dos dados devido impedimentos da ferramenta Graylog que não habilitou o protocolo para uso sem licenciamento, então para contornar este problema, foi feito uso do protocolo *IPFix* disponível na ferramenta Graylog e também disponível no ambiente NSX da VMware, contudo para uso deste protocolo se fez necessário a criação de uma arquivo JSON contendo a estrutura do padrão IPFix a ser utilizado pelo Graylog para entendimento da ferramenta de como interpretar e colunar os dados recebidos da forma correta. Para isso foram utilizadas as instruções existentes no sítio da *Internet Assigned Numbers Authority (IANA)* "https://www.iana.org/assignments/ipfix/ipfix.xhtml" [65].

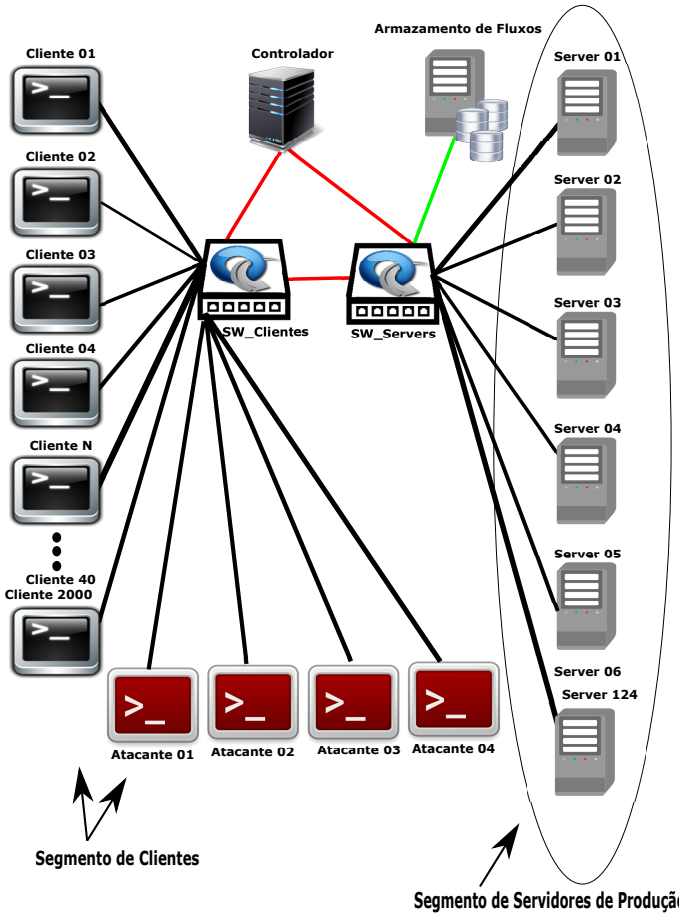

Figura 6.11: Topologia SDN do Escopo do Ambiente Real.

Este ajuste em nada prejudicou o Modelo realizado no MiniNet, pois todos os campos/atributos que foram selecionados e utilizados no projeto de pesquisa existem tanto em um como no outro protocolo, isso foi verificado por meio de análise que gerou uma Planilha ilustrada na Figura 6.12 que comparou as estruturas dos dois protocolos *Netflow* e IPFix onde se percebe que existe mais opções de campos de atributos no IPFix do que no *Netflow* com sutis diferenças na nomenclatura de campos que representam a mesma informação e outros poucos que possuem correlatos em um e no outro conforme verificado na Tabela com extração de trechos nos dois casos.

Na ilustração o que está na cor verde foram os atributos selecionados em primeira análise para uso no MiniNet e depois repetido em Ambiente Real, os que estão na cor verde com texto na cor vermelha são os atributos que permaneceram em segunda análise e foram utilizados no Projeto de Pesquisa, por fim os estão na cor amarela foram os campos/atributos que não encontramos correspondentes em nenhum dos dois, mas que são irrelevantes para este projeto por não terem sido selecionados e utilizados em nenhum momento. Para confirmação do entendimento da análise feita, segundo o comparativo dos campos/atributos dos dois protocolos, foram extraídas amostras dos dados utilizando um

Coluna	Netflow		IPFIX
1	timestamp		timestamp
2	source		source
3	message		message
4	nf_dst_address		destinationIPv4Address
5	nf_dst_port		destinationTransportPort
6	nf_dst		egressInterface
7	nf_dst_as		egressInterfaceAttr
8	nf_stop		flowEndMilliseconds
9	nf_start		flowStartMilliseconds
10	full_message		full_message
11	gl2_accounted_message_size		gl2_accounted_message_size
12	gl2_message_id		gl2_message_id
13	gl2_processing_timestamp		gl2_processing_timestamp
14	gl2_receive_timestamp		gl2_receive_timestamp
15	gl2_remote_ip		gl2_remote_ip
16	gl2_remote_port		gl2_remote_port
17	gl2_source_input		gl2_source_input
18	gl2_source_node		gl2_source_node
19	nf_src		ingressInterface
20	nf_src_as		ingressInterfaceAttr
21	nf_proto_name		ipClassOfService
22	nf_bytes		octetDeltaCount
23	nf_pkts		packetDeltaCount
24	nf_proto		protocolIdentifier
25	nf_src_address		sourceIPv4Address
26	nf_src_port		sourceTransportPort
27	streams		streams
28	nf_tcp_flags		tcpControlBits
29	nf_snmp_input		sourceIPv6Address
30	nf_snmp_output		destinationIPv6Address
31	nf_flow_packet_id		paddingOctets
32	nf_src_mask		flowDirection
33	nf_dst_mask		flowEndReason
34	nf_tos		layer2SegmentId
35	nf_version		maximumTTL
36			vxlanExportRole

Figura 6.12: Comparativo da Estrutura *Netflow* e IPFix.

protocolo e depois o outro, onde são apresentados na Figura 6.13 uma pequena amostra do conteúdo destes campos que confirmam o entendimento de suas igualdades, quanto aos atributos selecionados e utilizados no Projeto de Pesquisa e que não são impactados por esta diferença de protocolos na coleta de dados, em MiniNet com *Netflow* e no ambiente Real, com IPFix, conforme aqui demonstrado e confirmados pelos resultados que serão apresentados mais à frente. Os atributos selecionados e utilizados estão na Figura 6.13 no Trecho 01 Colunas B, E e F; Trecho 04 Colunas Z e AA.

Trecho 01

	A	B	C	D	E	F	G
1	Netflow	timestamp	source	message	nf_dst_address	nf_dst_port	nf_dst
2	Amostra Netflow	2020-09-26T14:04:08.390Z	10.0.2.15	NetFlowV5 [10.0.0.20]:0 <> [10.0.0.17]:0 proto:1 pkts:1 bytes:98	10.0.0.17	0	10.0.0.17:0
3	Amostra Netflow	2020-09-26T14:04:07.390Z	10.0.2.15	NetFlowV5 [10.0.0.20]:0 <> [10.0.0.1]:0 proto:1 pkts:1 bytes:98	10.0.0.1	0	10.0.0.1:0
4							
5	IPFIX	timestamp	source	message	destinationIPv4Address	destinationTransportPort	egressInterface
6	Amostra IPFIX	2020-09-24T19:39:37.000Z	10.195.22.1	Ipfix [10.115.1.3]:54325 <> [10.195.1.57]:63752 proto:6 pkts:1 bytes:224	10.195.1.57	63752	442
7	Amostra IPFIX	2020-09-24T19:39:37.000Z	10.195.22.1	Ipfix [10.115.1.3]:54324 <> [10.195.1.57]:63751 proto:6 pkts:1 bytes:125	10.195.1.57	63751	442

Trecho 02

	H	I	J	K	L	M	N	O
1	nf_dst_as	nf_stop	nf_start	full_message	gl2_accounted_message_size	gl2_message_id	gl2_processing_timestamp	gl2_receive_times
2	0	2020-09-26T14:03:58.299Z	2020-09-26T14:03:58.299Z		498	01EK5BMTAW6FYW3YF4EXQFRZPZ		
3	0	2020-09-26T14:03:57.300Z	2020-09-26T14:03:57.300Z		495	01EK5BMSGFC52SXMPM2XFMYX9S		
4								
5	egressInterfaceAttr	flowEndMilliseconds	flowStartMilliseconds	full_message	gl2_accounted_message_size	gl2_message_id	gl2_processing_timestamp	gl2_receive_times
6	2	2020-09-24T19:39:22Z	2020-09-24T19:39:22Z		678	01EK0T1NG0A9P1QV9HKQ6NRJN3		
7	2	2020-09-24T19:39:22Z	2020-09-24T19:39:22Z		678	01EK0T1NG07FJP35KTSK13GJN4		

Trecho 03

	P	Q	R	S	T	U	V	W	X
1	gl2_remote_ip	gl2_remote_port	gl2_source_input	gl2_source_node	nf_src	nf_src_as	nf_proto_name	nf_bytes	nf_pkts
2	10.0.2.15	34252	5f6f4a35a14d46040daf853d	8df3fd7e-3631-441f-b77e-a0260041c422	10.0.0.20:0	0	ICMP	98	1
3	10.0.2.15	34252	5f6f4a35a14d46040daf853d	8df3fd7e-3631-441f-b77e-a0260041c422	10.0.0.20:0	0	ICMP	98	1
4									
5	gl2_remote_ip	gl2_remote_port	gl2_source_input	gl2_source_node	ingressInterface	ingressInterfaceAttr	ipClassOfService	octetDeltaCount	packetDeltaCount
6	10.195.22.1	12055	5f6cf5f7cf653f35525d3630	73b46e4c-ac2d-40e3-9f1f-2e06ebe10561	405	2	2	224	1
7	10.195.22.1	12055	5f6cf5f7cf653f35525d3630	73b46e4c-ac2d-40e3-9f1f-2e06ebe10561	405	2	2	125	1

Trecho 04

	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK
1	nf_proto	nf_src_address	nf_src_port	streams	nf_tcp_flags	nf_snmp_input	nf_snmp_output	nf_flow_packet_id	nf_src_mask	nf_dst_mask	nf_tos	nf_version	
2	1	10.0.0.20	0	[00000000000000000000000001]	0	20	17	17	0	0	0	5	
3	1	10.0.0.20	0	[00000000000000000000000001]	0	20	1	1	0	0	0	5	
4													
5	protocolIdentifier	sourceIPv4Address	sourceTransportPort	streams	tcpControlBits	sourceIPv6Address	destinationIPv6Address	paddingOctets	flowDirection	flowEndReason	layer2SegmentId	maximumTTL	vxlanExportRole
6	6	10.115.1.3	54325	[00000000000000000000000001]	24			0	1	1	7,20576E+16	125	1
7	6	10.115.1.3	54324	[00000000000000000000000001]	24			0	1	1	7,20576E+16	125	1

Figura 6.13: Amostras dos Conteúdos dos Campos dos Protocolos *Netflow* e IPFix.

O ataque DDoS volumétrico no ambiente Real, conforme ilustrado na Figura 6.14, foi realizado com três atacantes utilizando o T50 versão 5.8.7 [63] com *spoofing* explorando os

protocolos UDP e TCP, testando uma frequência de solicitações próximo à ocorrência de paralisação, mas sem a interrupção do serviço durante a medição com carga de trabalho no servidor alvo entre 90% a 95%, entre dados legítimos e espúrios. Foram definidas as portas mais utilizadas para serviços em produção, desenvolvimento e outras aplicações, que são as seguintes: 80, 443, 25, 8080, 143, 22, 9200, 8443 e 2575, com cada um dos três servidores alvo respondendo por três portas diferentes e todos os atacantes explorando todas essas portas nos três alvos. No ataque, várias portas abertas, com serviços legítimos foram acessadas para evitar o viés de dados e não prejudicar os algoritmos ML. Um dos atacantes foi posicionado com acesso via VPN para se aproximar mais da realidade e testar a solução com acesso partindo fora do segmento de clientes.

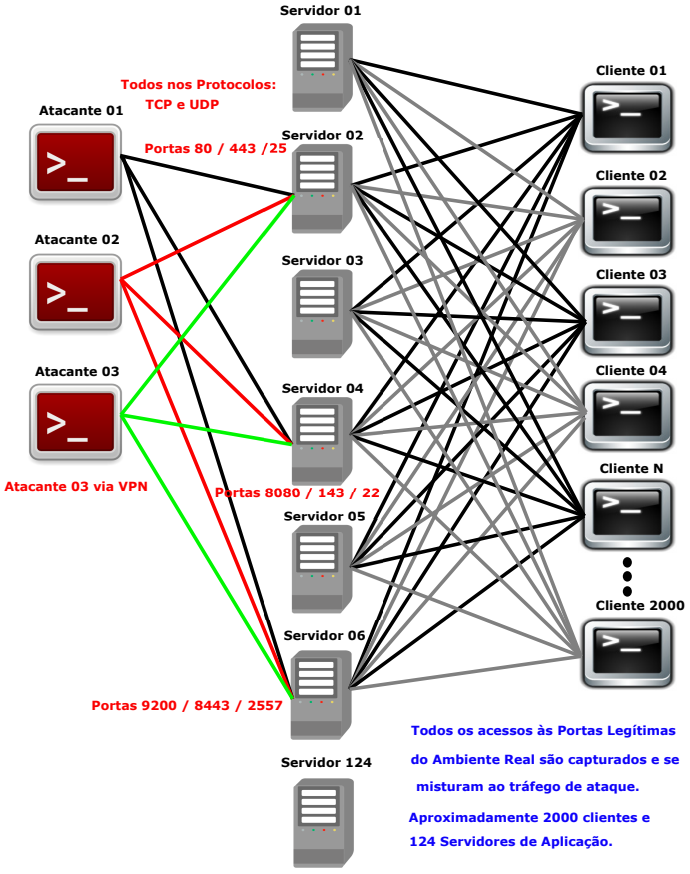

Figura 6.14: Topologia do Ataque no Ambiente Real.

No ambiente real o ataque foi realizado de forma contínua sem interrupção com 03 minutos anteriores de tráfego normal e 03 minutos posteriores com ataque, finalizando com mais 03 minutos de tráfego normal, procurando se aproximar ao máximo das condições reais e comprovação de que as condições aplicadas no ambiente emulado não tendenciaram os resultados no mundo real. As janelas foram definidas pelo tipo fluxos e tempo de três tamanhos diferentes para cada um dos tipos, totalizando seis janelas diferentes para este

ambiente, assim como no Modelo. As janelas por fluxos deste ambiente foram definidas em 100, 125 e 150 fluxos e as janelas por tempo foram de 3, 6 e 9 segundos.

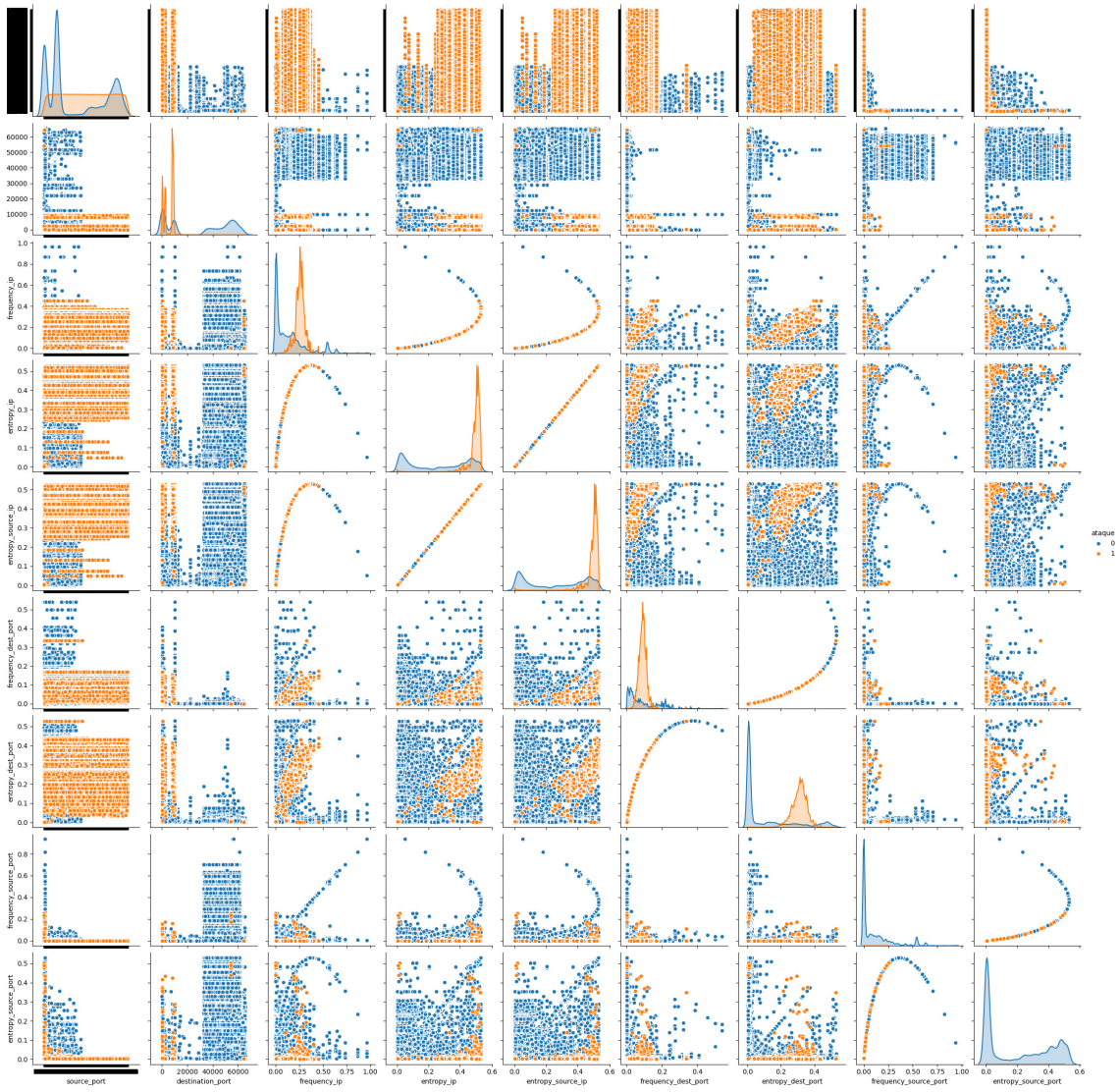

Figura 6.15: Comparação dos Atributos no *Dataset* do Ambiente Real.

Os atributos mais relevantes foram apontados pelos algoritmos, apoiados pelo pacote "*Seaborn*" do Python por meio da combinação de pares de atributos no conjunto de dados do Ambiente Real, de acordo com a Figura 6.15, com os dados normais representados pelos pontos azuis e os de ataque pelos pontos laranja, onde os resultados obtidos que demonstram similaridades de comportamentos do ambiente emulado e real. Nos eixos X e Y estão os nomes dos atributos, do conjunto de dados, que foram comparados pelo algoritmo e apresentam suas combinações nas interseções e nos cruzamentos de mesmos atributos o gráfico de linha é plotado no eixo diagonal da Figura 6.15.

Comparando a Figura 6.6 do Ambiente Emulado com a Figura 6.15 do Ambiente Real a diferença entre os dois gráficos é basicamente na densidade de dados que é significativa-

mente maior no conjunto de dados real o que demonstra que o ambiente emulado montado atendeu o requisito de replicar o mundo real de maneira simplificada para testar a solução em Modelo emulado antes de aplicar no mundo real. Diversos pares foram testados e aqui foram apresentados os pares que melhor separaram os dados e serviram de Limiares-X em Entropia e em ML-Entropia para detecção do ataque. Na Figura 6.16 os dados do Ambiente Real são ilustrados em 2D e 3D com os gráficos gerados com os limiares em três eixos para demonstrar a separação dos dados, onde os dados de ataque estão na cor amarela e dados normais na cor azul. Da mesma forma que ocorreu no Ambiente Emulado.

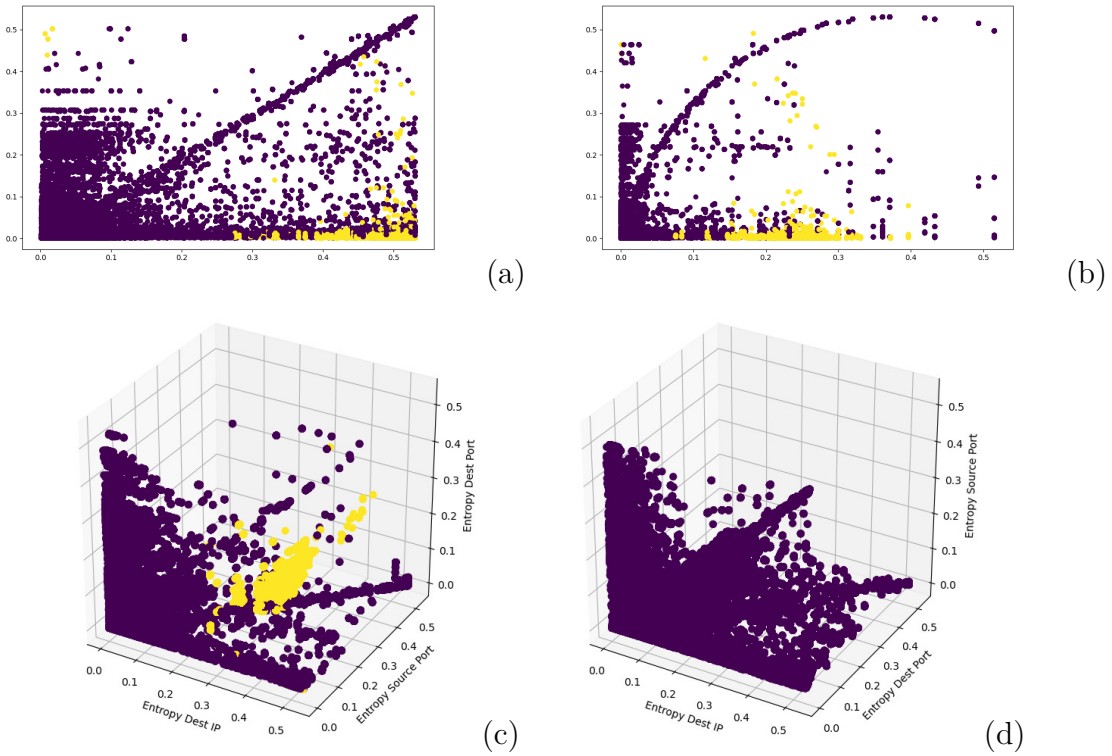

Figura 6.16: *Scatterplot* dos Dados do Ambiente Real: Max Entropia IP de Dest. com Porta de Origem (a), Entropia Freq. do IP de Dest. com IP de Origem (b), *Scatterplot* dos 3 Eixos com Ataque (c) e *Scatterplot* dos 3 Eixos sem Ataque (d).

É possível perceber melhor a similaridade entre os ambientes com a nítida diferença da densidade dos dados na Figura 6.17 que ilustra as detecções do algoritmo SVC em dados do ambiente emulado (acima) e do ambiente real (abaixo). Os dados legítimos estão na cor azul e os dados de ataque na cor vermelha, para cada tamanho de janela por fluxo. Para cada conjunto de quatro quadros que apresentam os resultados do SVC com o *kernel* Linear, Linear com Ajuste, RBF e Polynomial. Note a linha branca que representa o hiperplano, encontrado pelo algoritmo, às vezes sobreposta pela alta densidade dos pontos, na separação dos dados. Os pontos azuis que se encontram na separação em vermelho fazem parte da margem de erro, assim como os pontos vermelho na separação

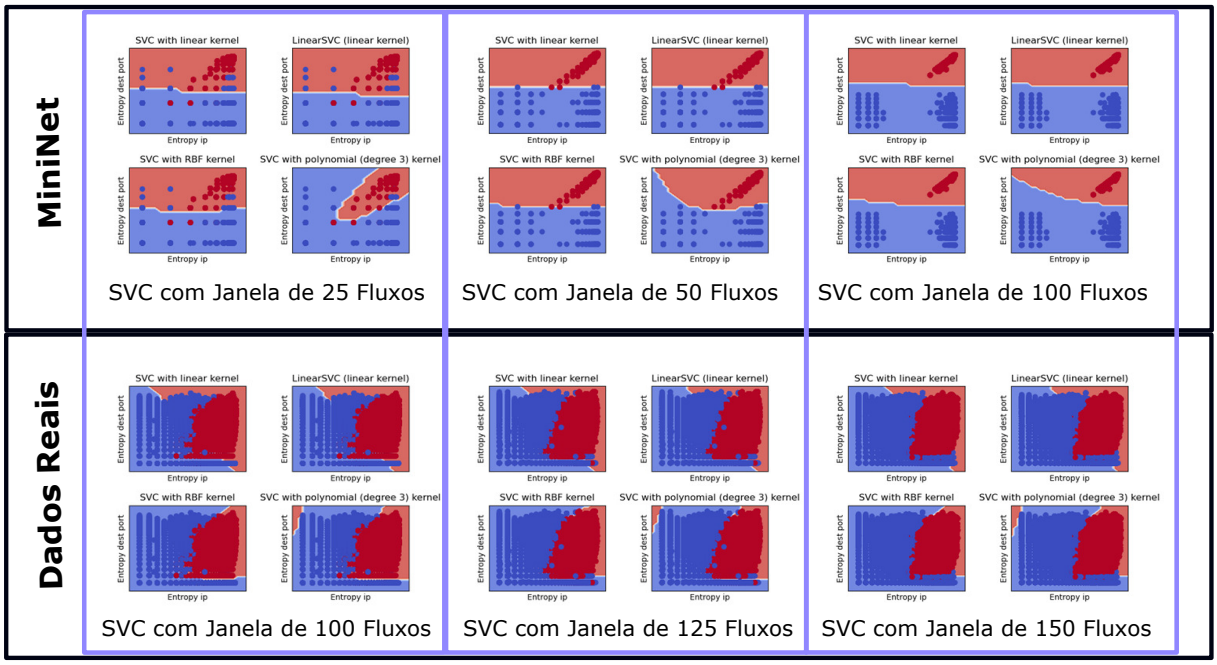

Figura 6.17: *Support Vector Classifier (SVC)* na Predição do Ataque nos Dados do Mininet e do Ambiente Real.

azul, depois do hiperplano. Essa margem de erro que procuramos reduzir e que são obtidas por RBF e Polynomial, conforme ilustrado. Observe, ainda na Figura 6.17, que à medida que a janela cresce em tamanho os dados vão ficando mais separados e melhor é a separação dos dados no hiperplano. Nos dados do MiniNet esta imagem é mais clara, contudo notamos no experimento que a partir de um determinado aumento de janela os dados ficam muito dispersos e a detecção começa a degradar suas métricas aferidas.

No processo de definição dos melhores limiares para uso na detecção de ataque DDoS por Entropia a análise foi feita apoiada nos gráficos gerados a partir da correlação apresentada na Figura 6.15 para os dados reais e com a geração de histogramas dos pares que apresentaram melhor correlação de separação dos dados, aqui apresentamos um exemplo na Figura 6.18. Nessa análise é possível perceber que apesar das topologias dos ataques serem diferentes, conforme já comentado e justificado, o comportamento dos dados em ambos os ambientes MIniNet (acima) e Real (abaixo) são similares. Os candidatos a limiares foram plotados no histograma conforme ilustrado na Figura 6.19 onde os possíveis ajustes foram aplicados e posteriormente testados para aferição dos resultados até encontrar os melhor parâmetros de comparação para selecioná-los como Limiares-X na detecção. Isso foi primeiramente realizado nas janelas definidas por fluxo e depois por tempo. Para cada ambiente, emulado e real.

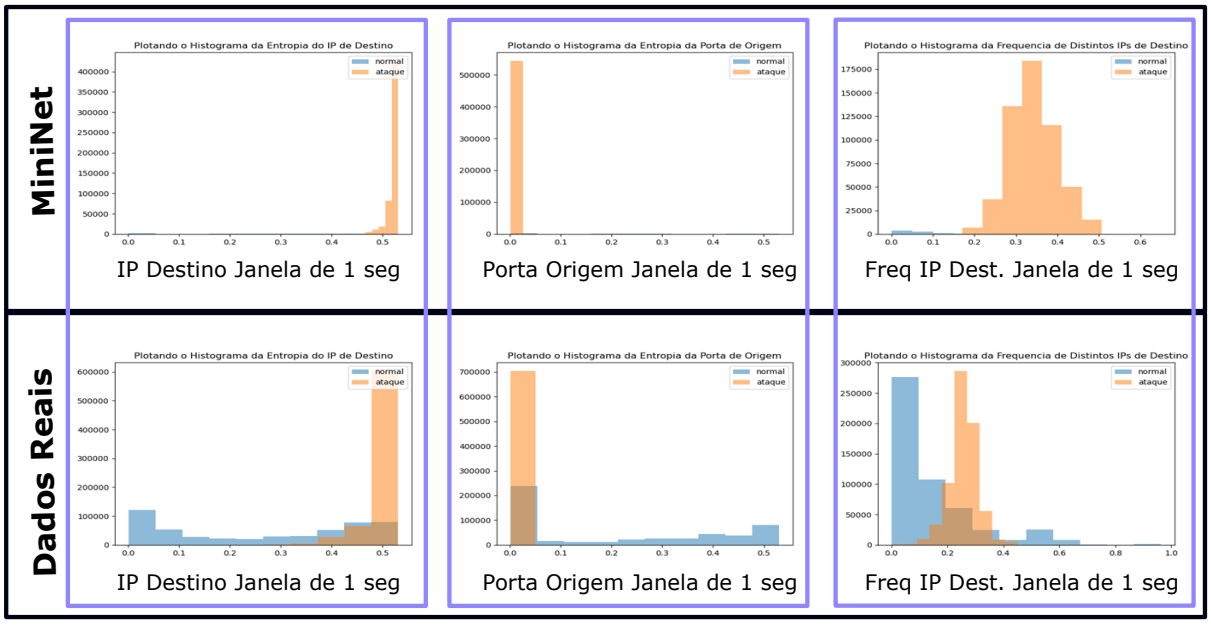

Figura 6.18: Histogramas com Janelas de 1 segundo para os Dados Emulados e Reais.

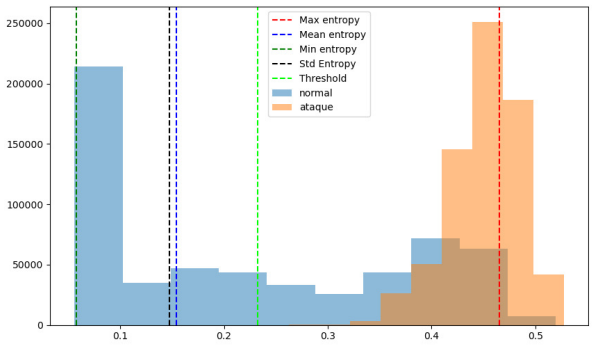

Figura 6.19: Histograma do IP de Destino nos Dados do Ambiente Real, Plotando as Entropias Calculadas.

6.2 Apresentação dos Resultados

Nesta seção os resultados são apresentados, comentados e comparados entre si, dividindo em subseções separando os resultados do Ambiente Emulado e do Ambiente Real para melhor compreensão e clareza das informações. Todos os resultados das Matrizes de Confusão (MC) com leitura conforme Figura 6.20, geradas a partir das detecções realizadas foram consolidados em planilha contendo os resultados, com intervalo de confiança (IC) de 95%, de: Entropia - Tradicional; ML-Entropia - Híbrida; SVR - Híbrido; SVC - Híbrido e *Random Forest* - Híbrido.

Para aferição dos resultados as métricas, já definidas e explicadas no Capítulo 5, Seção 5.1, foram usadas para possibilitar a comparação e avaliação dos resultados de todas as

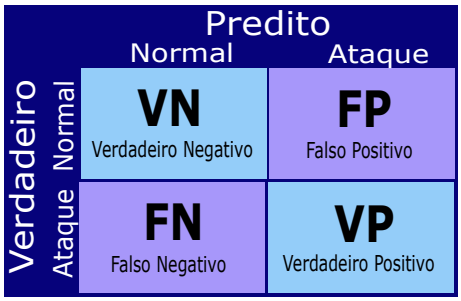

Figura 6.20: Matriz de Confusão de Referência.

técnicas aplicadas, sendo elas Acurácia, Precisão, Taxa de Erro, *Recall* e *F1-Score*. A estratégia de apresentação dos resultados foi definida de duas formas por meio de Tabelas, contendo todas as métricas e valores, e depois por Gráficos, contendo apenas Acurácia e F1-Score (média harmônica de precisão e *Recall*), para simplificar e facilitar o entendimento, visto que acurácia é o inverso de taxa de erro, assim na apresentação de um o outro já é evidente, com as seguintes considerações: Os valores são apresentados com três casas decimais após o zero para evitar a imprecisão do arredondamento, visto que foram muito próximos de 100%. Primeiro foram apresentados e comentados como Tabelas, para uma melhor compreensão dos dados gerados e analisados, com duas janelas no formato de Matriz de Confusão, contendo as Métricas do pior e do melhor resultado de cada ambiente (Emulado e Real), seja por janela definida por fluxo ou tempo, nos conjuntos de dados (Com e Sem Ataque). Não foram apresentados neste formato os demais resultados por se tratarem de doze Tabelas ao todo, de difícil análise e compreensão, contudo são apresentados em gráficos todos os resultados deste projeto de pesquisa. Depois foram apresentados e comentados todos os resultados obtidos no experimento de cada ambiente no formato de Gráficos, explorando os recursos visuais que facilitam a análise e o entendimento dos ganhos, seguido de um comentário geral para o ambiente emulado e outro para o ambiente real.

6.2.1 Apresentação dos Resultados do Ambiente Emulado

Os dados da distribuição dos fluxos de tráfego normal e de ataque foram apresentados na Tabela 6.1 para o Ambiente Emulado (MiniNet). A Tabela 6.2, apresenta os piores resultados das soluções dentre todos os tamanhos de janelas, seja por fluxo ou tempo, no Ambiente Emulado (MiniNet) que foi janela de 25 fluxos.

À esquerda da Tabela estão os resultados para o conjunto de dados com ataque e à direita estão os resultados para o conjunto de dados sem ataque. A coluna janela indica o tamanho definido, seguido pelo tipo por fluxo ou tempo. Depois a Matriz de Confusão, seguida pelas Métricas referentes à MC adjacente. Os sub-títulos indicam qual foi a

Tabela 6.1: Distribuição do Tráfego no MiniNet.

Distribuição do Tráfego de Ataque		
IC 95%	**N. Fluxos**	%
Ataque	95555	87.34%
Normal	13855	12.66%
Total	109410	100%

Tabela 6.2: Resultados do Ambiente Emulado (MiniNet) para Janelas de 25 Fluxos.

Dataset Com Ataque - IC 95%							**Dataset Sem Ataque** - IC 95%				
Entropia - Tradicional							**Entropia - Tradicional**				
Janela	**Tipo**	**Matriz de Confusão**			**Métricas**		**Matriz de Confusão**			**Métricas**	
25	Fluxos		**Normal**	**Ataque**	**Acurácia**	85.489%		**Normal**	**Ataque**	**Acurácia**	99.445%
		Normal	8940	146	**Precisão**	99.969%	**Normal**	13795	77	**Precisão**	Não se aplica
		Ataque	80139	464053	**Taxa de Erro**	14.511%	**Ataque**	0	0	**Taxa de Erro**	0.555%
					Recall	85.274%				**Recall**	Não se aplica
					F1-Score	92.038%				**F1-Score**	Não se aplica
ML-Entropia - Híbrido							**ML-Entropia - Híbrido**				
Janela	**Tipo**	**Matriz de Confusão**			**Métricas**		**Matriz de Confusão**			**Métricas**	
25	Fluxos		**Normal**	**Ataque**	**Acurácia**	99.433%		**Normal**	**Ataque**	**Acurácia**	97.290%
		Normal	8918	168	**Precisão**	99.969%	**Normal**	13496	376	**Precisão**	Não se aplica
		Ataque	2968	541224	**Taxa de Erro**	0.567%	**Ataque**	0	0	**Taxa de Erro**	2.710%
					Recall	99.455%				**Recall**	Não se aplica
					F1-Score	99.711%				**F1-Score**	Não se aplica
SVR - Híbrido							**SVR - Híbrido**				
Janela	**Tipo**	**Matriz de Confusão**			**Métricas**		**Matriz de Confusão**			**Métricas**	
25	Fluxos		**Normal**	**Ataque**	**Acurácia**	99.970%		**Normal**	**Ataque**	**Acurácia**	92.990%
		Normal	152	2	**Precisão**	99.980%	**Normal**	9299	701	**Precisão**	Não se aplica
		Ataque	1	9845	**Taxa de Erro**	0.030%	**Ataque**	0	0	**Taxa de Erro**	7.010%
					Recall	99.990%				**Recall**	Não se aplica
					F1-Score	99.985%				**F1-Score**	Não se aplica
SVC - Híbrido							**SVC - Híbrido**				
Janela	**Janela**	**Matriz de Confusão**			**Métricas**		**Matriz de Confusão**			**Métricas**	
25	Fluxos		**Normal**	**Ataque**	**Acurácia**	100.00%		**Normal**	**Ataque**	**Acurácia**	50.470%
		Normal	161	0	**Precisão**	100,00%	**Normal**	5047	4953	**Precisão**	Não se aplica
		Ataque	0	9839	**Taxa de Erro**	0.00%	**Ataque**	0	0	**Taxa de Erro**	49.530%
					Recall	100.00%				**Recall**	Não se aplica
					F1-Score	100.00%				**F1-Score**	Não se aplica
Random Forest - Híbrido							**Random Forest - Híbrido**				
Janela	**Janela**	**Matriz de Confusão**			**Métricas**		**Matriz de Confusão**			**Métricas**	
25	Fluxos		**Normal**	**Ataque**	**Acurácia**	99.700%		**Normal**	**Ataque**	**Acurácia**	100.00%
		Normal	24	6	**Precisão**	99.696%	**Normal**	2000	0	**Precisão**	Não se aplica
		Ataque	0	1970	**Taxa de Erro**	0.300%	**Ataque**	0	0	**Taxa de Erro**	0.00%
					Recall	100.000%				**Recall**	Não se aplica
					F1-Score	99.848%				**F1-Score**	Não se aplica

solução testada para os resultados apresentados. Observa-se que ML-Entropia foi melhor em todas as métricas que Entropia nos dados com ataque e Entropia melhor nos dados

sem ataque, enquanto SVR foi um pouco melhor que ML-Entropia nos dados com ataque e ML-Entropia superior ao SVR nos dados sem ataque. Já o SVC obteve desempenho ótimo nos dados com ataque, mas errou quase 50% nos dados sem ataque. Por fim nota-se que o *Random Forest* obteve desempenho muito bom nos dados com ataque e ótimo nos dados sem ataque.

Tabela 6.3: Resultados do Ambiente Emulado (MiniNet) para Janelas de 100 Fluxos.

Dataset Com Ataque - IC 95%							**Dataset Sem Ataque** - IC 95%				
Entropia - Tradicional							**Entropia - Tradicional**				
Janela	**Tipo**	**Matriz de Confusão**			**Métricas**		**Matriz de Confusão**			**Métricas**	
100	Fluxos		**Normal**	**Ataque**	**Acurácia**	100.00%		**Normal**	**Ataque**	**Acurácia**	100.00%
		Normal	9086	0	**Precisão**	100.00%	**Normal**	13872	0	**Precisão**	Não se aplica
		Ataque	0	544192	**Taxa de Erro**	0.00%	**Ataque**	0	0	**Taxa de Erro**	0.00%
					Recall	100.00%				**Recall**	Não se aplica
					F1-Score	100.00%				**F1-Score**	Não se aplica
ML-Entropia - Híbrido							**ML-Entropia - Híbrido**				
Janela	**Tipo**	**Matriz de Confusão**			**Métricas**		**Matriz de Confusão**			**Métricas**	
100	Fluxos		**Normal**	**Ataque**	**Acurácia**	100.00%		**Normal**	**Ataque**	**Acurácia**	100.00%
		Normal	9086	0	**Precisão**	100.00%	**Normal**	13872	0	**Precisão**	Não se aplica
		Ataque	0	544192	**Taxa de Erro**	0.00%	**Ataque**	0	0	**Taxa de Erro**	0.00%
					Recall	100.00%				**Recall**	Não se aplica
					F1-Score	100.00%				**F1-Score**	Não se aplica
SVR - Híbrido							**SVR - Híbrido**				
Janela	**Tipo**	**Matriz de Confusão**			**Métricas**		**Matriz de Confusão**			**Métricas**	
100	Fluxos		**Normal**	**Ataque**	**Acurácia**	99.561%		**Normal**	**Ataque**	**Acurácia**	99.856%
		Normal	6658	2428	**Precisão**	99.556%	**Normal**	13852	20	**Precisão**	Não se aplica
		Ataque	0	544192	**Taxa de Erro**	0.439%	**Ataque**	0	0	**Taxa de Erro**	0.144%
					Recall	100.000%				**Recall**	Não se aplica
					F1-Score	99.777%				**F1-Score**	Não se aplica
SVC - Híbrido							**SVC - Híbrido**				
Janela	**Janela**	**Matriz de Confusão**			**Métricas**		**Matriz de Confusão**			**Métricas**	
100	Fluxos		**Normal**	**Ataque**	**Acurácia**	100.00%		**Normal**	**Ataque**	**Acurácia**	51.640%
		Normal	9086	0	**Precisão**	100.00%	**Normal**	5164	4836	**Precisão**	Não se aplica
		Ataque	0	544192	**Taxa de Erro**	0.00%	**Ataque**	0	0	**Taxa de Erro**	48.360%
					Recall	100.00%				**Recall**	Não se aplica
					F1-Score	100.00%				**F1-Score**	Não se aplica
Random Forest - Híbrido							**Random Forest - Híbrido**				
Janela	**Janela**	**Matriz de Confusão**			**Métricas**		**Matriz de Confusão**			**Métricas**	
100	Fluxos		**Normal**	**Ataque**	**Acurácia**	100.00%		**Normal**	**Ataque**	**Acurácia**	100.00%
		Normal	1737	0	**Precisão**	100.00%	**Normal**	2775	0	**Precisão**	Não se aplica
		Ataque	0	108919	**Taxa de Erro**	0.00%	**Ataque**	0	0	**Taxa de Erro**	0.00%
					Recall	100.00%				**Recall**	Não se aplica
					F1-Score	100.00%				**F1-Score**	Não se aplica

A Tabela 6.3 apresenta os melhores resultados das soluções dentre todos os tamanhos de janelas, seja por fluxo ou tempo, no Ambiente Emulado (MiniNet) que foi janela de 100 fluxos. Observa-se que tanto Entropia quanto ML-Entropia obtiveram desempenho ótimo para os dados com e sem ataque, enquanto SVR melhor nos dados sem ataque mas piorou nos dados com ataque. O SVC praticamente não teve mudanças em relação

às janelas de 25 fluxos, enquanto o *Random Forest* também obteve resultado ótimo nos dados com e sem ataque.

Os Gráficos de todos os resultados obtidos de todas as janelas por fluxo e por tempo do Ambiente Emulado (MiniNet) são apresentados e comentados na sequência, com os resultados individuais lado a lado, servindo de comparação no projeto de pesquisa das diversas possibilidades de aplicabilidade do mecanismo híbrido em cada uma das soluções híbridas e da própria abordagem tradicional da Entropia. Primeiramente são apresentadas e comentadas as janelas por fluxo da menor para a maior e depois janelas por tempo com a mesma lógica. Na sequência os comentários gerais dos resultados obtidos para este ambiente foram feitos. Na Figura 6.21 são apresentados os resultados dos dados do ambiente emulado (MiniNet) com janelas de 25 fluxos para o conjunto de dados com ataque à esquerda e sem ataque à direita da Figura. Entropia obteve nos dados com ataque 85.489% de acurácia e 92.038% de F1-Score, mas foi o segundo melhor nos dados sem ataque. Já ML-Entropia obteve 99.433% de acurácia e 99.711% de F1-Score, ficando como terceiro melhor desempenho nos dados sem ataque. O SVR nos dados com ataque obteve 99.970% de acurácia e 99.985% de F1-Score, mas nos dados sem ataque ficou em quarto lugar. O SVC foi ótimo com 100% em todas as métricas nos dados com ataque, mas foi o pior resultado nos dados sem ataque. Finalmente o *Random Forest* teve um desempenho de 99.700% de acurácia e 99.848% de F1-Score, e foi o melhor nos dados sem ataque com métrica de 100%.

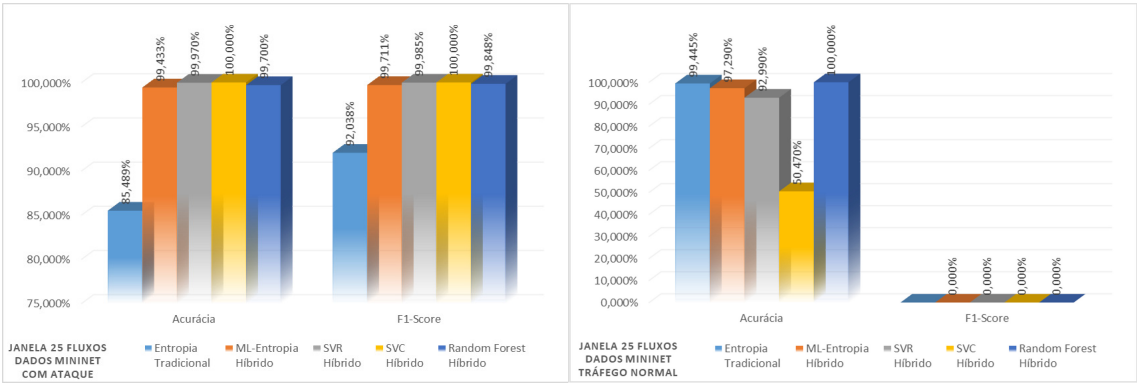

Figura 6.21: Detecção de Ataque no MiniNet com Janelas de 25 Fluxos.

Comentário da Figura 6.21: Para este tamanho de janela Entropia obteve o pior resultado nos dados com ataque, mas foi o segundo melhor nos dados sem ataque. Já ML-Entropia obteve um desempenho muito bom, acima de 99% nos dados com ataque, ficando como terceiro melhor desempenho nos dados sem ataque. O SVR foi muito bem nos dados com ataque, melhor inclusive que ML-Entropia, mas nos dados sem ataque ficou em quarto lugar. O SVC foi ótimo com 100% em todas as métricas nos dados com ataque, mas foi o pior resultado nos dados sem ataque. Finalmente o *Random Forest* teve

um desempenho muito bom nos dados com ataque e foi o melhor nos dados sem ataque com métrica de 100%.

Na Figura 6.22 são apresentados os resultados dos dados do ambiente emulado (MiniNet) com janelas de 50 fluxos para o conjunto de dados com ataque à esquerda e sem ataque à direita da Figura. Entropia obteve nos dados com ataque, 99.910% de acurácia e 99.954% de F1-Score, e 100% de acurácia sem ataque. ML-Entropia nos dados com ataque, 99.993% de acurácia e 99.997% de F1-Score, ficando acima de 99% nos dados sem ataque. O SVR nos dados com ataque, obteve 99.510% de acurácia e 99.752% de F1-Score, e sem ataque foi o segundo pior. O SVC nos dados com ataque ficou com 100% de acurácia e muito ruim nos dados sem ataque. O *Random Forest* nos dados com ataque, obteve 99.450% de acurácia e 99.721% de F1-Score, contudo nos dados sem ataque obteve 100% de acurácia.

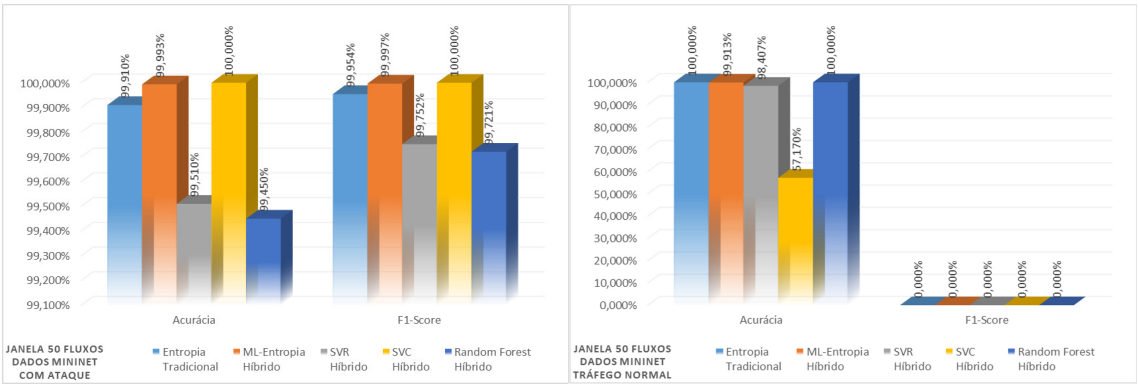

Figura 6.22: Detecção de Ataque no MiniNet com Janelas de 50 Fluxos.

Comentário da Figura 6.22: 50 fluxos por janela elevou os resultados de todas as soluções tanto nos dados com ataque quanto nos dados sem ataque. Entropia obteve um resultado quase ótimo nos dados com ataque e ótimo sem ataque. ML-Entropia foi um pouco melhor que Entropia nos dados com ataque e um pouco inferior a Entropia nos dados sem ataque. O SVR foi muito bem nos dados com e sem ataque, mas ficou atrás de Entropia e ML-Entropia. O SVC continuou com ótimo desempenho nos dados com ataque e muito ruim nos dados sem ataque. O *Random Forest* melhorou nos dados com ataque, mas foi inferior a ML-Entropia e Entropia e melhor nos dados sem ataque.

Na Figura 6.23 são apresentados os resultados dos dados do ambiente emulado (MiniNet) com janelas de 100 fluxos para o conjunto de dados com ataque à esquerda e sem ataque à direita da Figura. Entropia, ML-Entropia e *Random Forest* obtivem 100% de acurácia nos dois tipos de dados, com exceção do SVR nos dados com ataque, com 99.561% de acurácia e 99.777% de F1-Score. Já nos dados sem ataque além do SVR o SVC também obteve um desempenho inferior repetindo as ocorrências nas janelas anteriores em dados sem ataque.

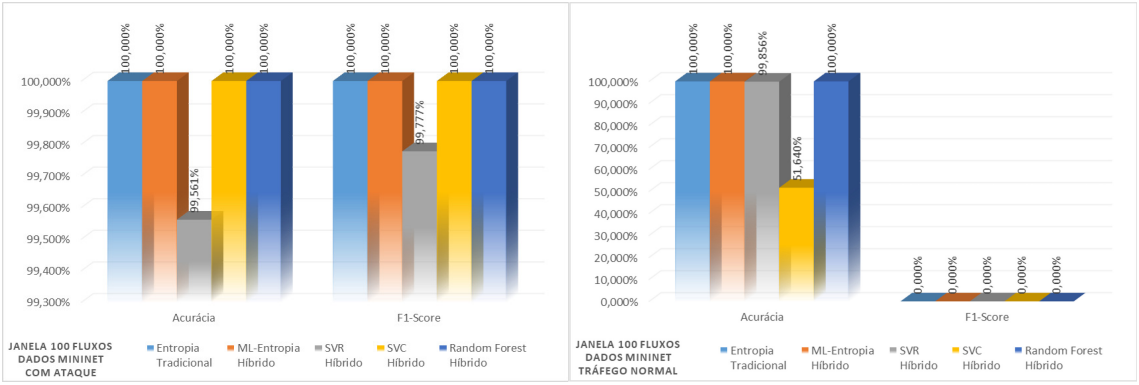

Figura 6.23: Detecção de Ataque no MiniNet com Janelas de 100 Fluxos.

Comentário da Figura 6.23: Esta janela de 100 fluxos representou a melhor opção para quase todas as soluções com exceção do SVC que manteve o desempenho das janelas anteriores. Entropia, ML-Entropia e *Random Forest* obtivem 100% de resultados nos dois tipos de dados e o SVR muito próximo do ótimo.

Na Figura 6.24 são apresentados os resultados dos dados do ambiente emulado (MiniNet) com janelas de 1 segundo para o conjunto de dados com ataque à esquerda e sem ataque à direita da Figura. Entropia nos dados com ataque, ficou com 92.738% de acurácia e 96.167% de F1-Score, e segundo melhor nos dados sem ataque. ML-Entropia nos dados com ataque, obteve 99.834% de acurácia e 99.916% de F1-Score, e nos dados sem ataque ficou em terceiro lugar. O SRV obteve nos dados com ataque, 99.392% de acurácia e 99.692% de F1-Score, e nos dados sem ataque foi o segundo pior resultado. O SVC nos dados com ataque, ficou com 99.924% de acurácia e 99.961% de F1-Score, e foi o pior desempenho de todos nos dados sem ataque. O *Random Forest* obteve nos dois tipos de dados 100% de acurácia.

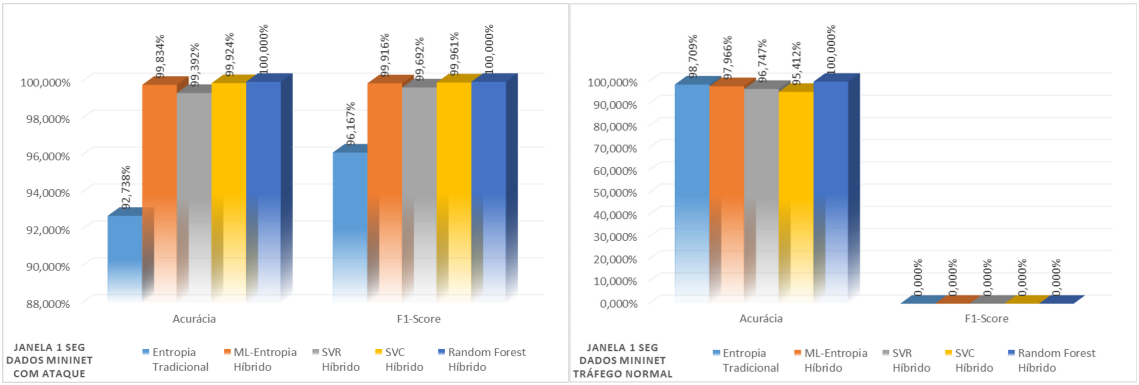

Figura 6.24: Detecção de Ataque no MiniNet com Janelas de 1 Segundo.

Comentário da Figura 6.24: Iniciando as janelas por tempo temos o MiniNet com janelas de 1 segundo que obteve bons resultados gerais, ficando Entropia muito abaixo

de ML-Entropia nos dados com ataque e pouco superior a este nos dados sem ataque. ML-Entropia foi superior ainda ao SVR nos dois tipos de dados. O SVC obteve pouca diferença acima de ML-Entropia nos dados com ataque mas com pior desempenho de todos nos dados sem ataque. O *Random Forest* foi ótimo nos dois tipos de dados.

Na Figura 6.25 são apresentados os resultados dos dados do ambiente emulado (MiniNet) com janelas de 2 segundos para o conjunto de dados com ataque à esquerda e sem ataque à direita da Figura. Entropia obteve nos dados com ataque, 94.491% de acurácia e 97.119% de F1-Score, e nos dados sem ataque foi o segundo melhor. ML-Entropia Nos dados com ataque obteve 99.902% de acurácia e 99.950% de F1-Score, e nos dados sem ataque acima de 99%. O SVC nos dados com ataque obteve 99.990% de acurácia e 99.995% de F1-Score, nos dados sem ataque foi o quarto colocado. O *Random Forest* nos dados com e sem ataque obteve 100% de acurácia.

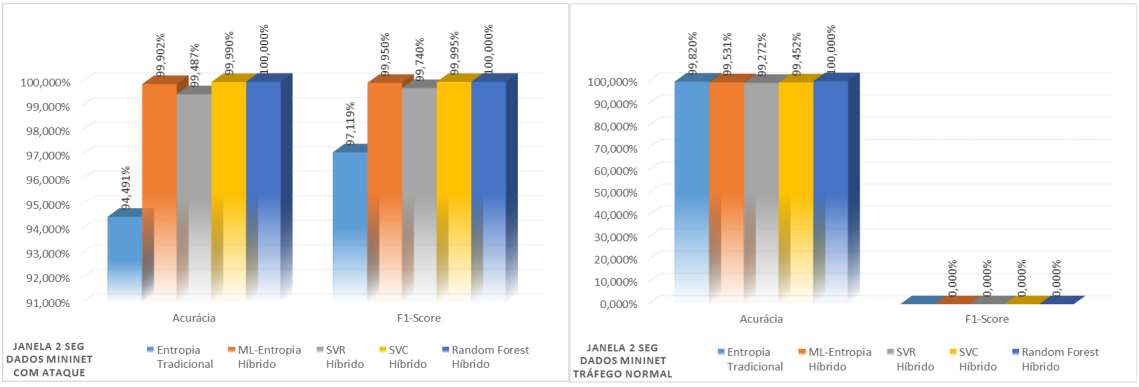

Figura 6.25: Detecção de Ataque no MiniNet com Janelas de 2 Segundos.

Comentário da Figura 6.25: Em janelas de 2 segundos Entropia obteve o pior resultado nos dados com ataque com Taxa de Erro acima de 5% enquanto todos os outros ficaram abaixo de 1%. Nos dados sem ataque obteve o segundo melhor desempenho ficando atrás apenas de *Random Forest*, contudo todos os outros ficaram acima de 99% de Acurácia. Apenas *Random Forest* obteve 100% nos dois tipos de dados.

Na Figura 6.26 são apresentados os resultados dos dados do ambiente emulado (MiniNet) com janelas de 3 segundos para o conjunto de dados com ataque à esquerda e sem ataque à direita da Figura. Entropia nos dados com ataque, 97.941% de acurácia e 98.942% de F1-Score, mas obteve 100% nos dados sem ataque. ML-Entropia nos dados com ataque ficou com 99.933% de acurácia e 99.966% de F1-Score, e nos dados sem ataque obteve 100% de acurácia. O SVR foi um pouco inferior aos dois anteriores nos dados com ataque, com 99.478% de acurácia e 99.736% de F1-Score, nos dados sem ataque foi acima de 99%. O *Random Forest* obteve 100% nos dois tipos de dados.

Comentário da Figura 6.26: Finalizando as janelas do ambiente Emulado temos janelas de 3 segundos temos resultados muito próximos do ótimo para todos com exceção de

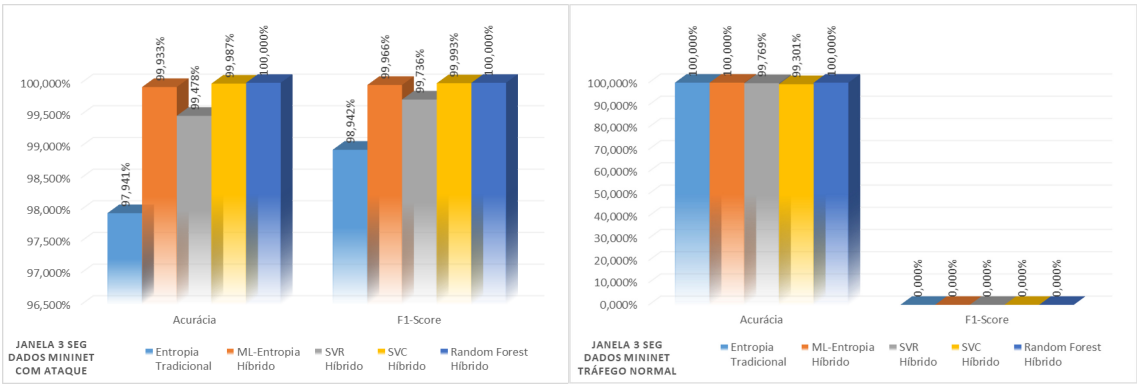

Figura 6.26: Detecção de Ataque no MiniNet com Janelas de 3 Segundos.

Entropia que obteve Taxa de Erro acima de 2% nos dados com ataque, enquanto o restante ficou abaixo de 1%. Nos dados sem ataque Entropia juntamente com ML-Entropia e *Random Forest* obtiveram 100%, enquanto SVR e SVC ficaram próximos disso.

Comentário Geral para os resultados do Ambiente Emulado (MiniNet): De modo geral os resultados de janelas por fluxos foram melhores para quase todas as soluções, chegando a 100% em janelas de 100 fluxos para Entropia, ML-Entropia e *Random Forest*, com exceção do SVC. Vale ressaltar que as janelas por tempo apesar de não ter atingido 100% para quase todos, como ocorrido nas janelas por fluxos, foi a melhor opção para o SVC que obteve desempenhos quase ótimos para os dois tipos de dados, o que não ocorreu para ele nas janelas por fluxos, indicando que para este algoritmo a abordagem temporal é a melhor indicação. Também há de se ressaltar que as diferenças de desempenho de ML-Entropia para Entropia nos dados sem ataque é muito menor nas janelas por tempo o que também indica melhor abordagem de trabalho para reduzir as diferenças com menores taxas de erro. Selecionar os melhores limiares a partir de janelas temporais pode ser uma alternativa para se obter ótimos resultados para ML-Entropia, assim como ocorreu em janelas de 100 fluxos.

6.2.2 Apresentação dos Resultados do Ambiente Real

Os dados da distribuição dos fluxos de tráfego normal e de ataque foram apresentados na Tabela 6.4 para o Ambiente Real. A Tabela 6.5 apresenta os piores resultados obtidos no Ambiente Real, com janelas de 03 segundos.

Note que assim como no ambiente emulado a ML-Entropia foi melhor que a Entropia nos dados com ataque, enquanto a Entropia foi melhor nos dados sem ataque. ML-Entropia foi melhor que SVR nos dados com e sem ataque. O SVC obteve desempenho muito bom nos dados com e sem ataque. Por fim o *Random Forest* foi ótimo nos dados com e sem ataque. Os resultados referentes ao ambiente Real que apresentaram as melhores

Tabela 6.4: Distribuição do Tráfego no Ambiente Real.

Distribuição do Tráfego de Ataque		
IC 95%	**N. Fluxos**	%
Ataque	705808	54.68%
Normal	584931	45.32%
Total	1290739	100%

Tabela 6.5: Resultados do Ambiente Real para Janelas de 03 Segundos.

Dataset Com Ataque - IC 95%							**Dataset Sem Ataque** - IC 95%				
Entropia - Tradicional							**Entropia - Tradicional**				
Janela	**Tipo**	**Matriz de Confusão**			**Métricas**		**Matriz de Confusão**			**Métricas**	
3	Segundos		**Normal**	**Ataque**	**Acurácia**	91.385%		**Normal**	**Ataque**	**Acurácia**	95.073%
		Normal	489468	25811	**Precisão**	96.043%	**Normal**	825990	42803	**Precisão**	Não se aplica
		Ataque	79384	626421	**Taxa de Erro**	8.615%	**Ataque**	0	0	**Taxa de Erro**	4.927%
					Recall	88.753%				**Recall**	Não se aplica
					F1-Score	92.254%				**F1-Score**	Não se aplica
ML-Entropia - Híbrido							**ML-Entropia - Híbrido**				
Janela	**Tipo**	**Matriz de Confusão**			**Métricas**		**Matriz de Confusão**			**Métricas**	
3	Segundos		**Normal**	**Ataque**	**Acurácia**	96.561%		**Normal**	**Ataque**	**Acurácia**	84.866%
		Normal	476802	38477	**Precisão**	94.806%	**Normal**	737309	131484	**Precisão**	Não se aplica
		Ataque	3514	702291	**Taxa de Erro**	3.439%	**Ataque**	0	0	**Taxa de Erro**	15.134%
					Recall	99.502%				**Recall**	Não se aplica
					F1-Score	97.097%				**F1-Score**	Não se aplica
SVR - Híbrido							**SVR - Híbrido**				
Janela	**Tipo**	**Matriz de Confusão**			**Métricas**		**Matriz de Confusão**			**Métricas**	
3	Segundos		**Normal**	**Ataque**	**Acurácia**	95.800%		**Normal**	**Ataque**	**Acurácia**	81.987%
		Normal	464235	51044	**Precisão**	93.254%	**Normal**	712301	156492	**Precisão**	Não se aplica
		Ataque	239	705566	**Taxa de Erro**	4.200%	**Ataque**	0	0	**Taxa de Erro**	18.013%
					Recall	99.966%				**Recall**	Não se aplica
					F1-Score	96.493%				**F1-Score**	Não se aplica
SVC - Híbrido							**SVC - Híbrido**				
Janela	**Janela**	**Matriz de Confusão**			**Métricas**		**Matriz de Confusão**			**Métricas**	
3	Segundos		**Normal**	**Ataque**	**Acurácia**	99.470%		**Normal**	**Ataque**	**Acurácia**	99.204%
		Normal	514136	1143	**Precisão**	99.837%	**Normal**	861879	6914	**Precisão**	Não se aplica
		Ataque	5330	700475	**Taxa de Erro**	0.530%	**Ataque**	0	0	**Taxa de Erro**	0.796%
					Recall	99.245%				**Recall**	Não se aplica
					F1-Score	99.540%				**F1-Score**	Não se aplica
Random Forest - Híbrido							**Random Forest - Híbrido**				
Janela	**Janela**	**Matriz de Confusão**			**Métricas**		**Matriz de Confusão**			**Métricas**	
3	Segundos		**Normal**	**Ataque**	**Acurácia**	100.00%		**Normal**	**Ataque**	**Acurácia**	100.00%
		Normal	102962	0	**Precisão**	100.00%	**Normal**	173759	0	**Precisão**	Não se aplica
		Ataque	0	141255	**Taxa de Erro**	0.00%	**Ataque**	0	0	**Taxa de Erro**	0.00%
					Recall	100.00%				**Recall**	Não se aplica
					F1-Score	100.00%				**F1-Score**	Não se aplica

métricas para todas as soluções estão na Tabela 6.6 com janelas de 100 fluxos. Percebe-se que a ML-Entropia foi melhor que Entropia nos dados com ataque e Entropia melhor nos dados sem ataque. ML-Entropia foi melhor que o SVR nos dados com e sem ataque. O

SVC foi pouco melhor que ML-Entropia nos dados com ataque, mas perdeu para ML-Entropia nos dados sem ataque. O *Random Forest* foi ótimo nos dados com e sem ataque.

Tabela 6.6: Resultados do Ambiente Real para Janelas de 100 Fluxos.

Dataset Com Ataque - IC 95%							**Dataset Sem Ataque** - IC 95%				
Entropia - Tradicional							**Entropia - Tradicional**				
Janela	**Tipo**	**Matriz de Confusão**			**Métricas**		**Matriz de Confusão**			**Métricas**	
100	Fluxos		**Normal**	**Ataque**	**Acurácia**	97.440%		**Normal**	**Ataque**	**Acurácia**	96.839%
		Normal	579329	5602	**Precisão**	99.181%	**Normal**	841331	27462	**Precisão**	Não se aplica
		Ataque	27440	678368	**Taxa de Erro**	2.560%	**Ataque**	0	0	**Taxa de Erro**	3.161%
					Recall	96.112%				**Recall**	Não se aplica
					F1-Score	97.622%				**F1-Score**	Não se aplica
ML-Entropia - Híbrido							**ML-Entropia - Híbrido**				
Janela	**Tipo**	**Matriz de Confusão**			**Métricas**		**Matriz de Confusão**			**Métricas**	
100	Fluxos		**Normal**	**Ataque**	**Acurácia**	99.090%		**Normal**	**Ataque**	**Acurácia**	94.160%
		Normal	575065	9866	**Precisão**	98.618%	**Normal**	818056	50737	**Precisão**	Não se aplica
		Ataque	1880	703928	**Taxa de Erro**	0.910%	**Ataque**	0	0	**Taxa de Erro**	5.840%
					Recall	99.734%				**Recall**	Não se aplica
					F1-Score	99.173%				**F1-Score**	Não se aplica
SVR - Híbrido							**SVR - Híbrido**				
Janela	**Tipo**	**Matriz de Confusão**			**Métricas**		**Matriz de Confusão**			**Métricas**	
100	Fluxos		**Normal**	**Ataque**	**Acurácia**	97.223%		**Normal**	**Ataque**	**Acurácia**	81.443%
		Normal	549956	34975	**Precisão**	95.273%	**Normal**	707573	161220	**Precisão**	Não se aplica
		Ataque	870	704938	**Taxa de Erro**	2.777%	**Ataque**	0	0	**Taxa de Erro**	18.557%
					Recall	99.877%				**Recall**	Não se aplica
					F1-Score	97.521%				**F1-Score**	Não se aplica
SVC - Híbrido							**SVC - Híbrido**				
Janela	**Janela**	**Matriz de Confusão**			**Métricas**		**Matriz de Confusão**			**Métricas**	
100	Fluxos		**Normal**	**Ataque**	**Acurácia**	99.764%		**Normal**	**Ataque**	**Acurácia**	88.890%
		Normal	583894	1037	**Precisão**	99.853%	**Normal**	8889	1111	**Precisão**	Não se aplica
		Ataque	2014	703794	**Taxa de Erro**	0.236%	**Ataque**	0	0	**Taxa de Erro**	11.110%
					Recall	99.715%				**Recall**	Não se aplica
					F1-Score	99.784%				**F1-Score**	Não se aplica
Random Forest - Híbrido							**Random Forest - Híbrido**				
Janela	**Janela**	**Matriz de Confusão**			**Métricas**		**Matriz de Confusão**			**Métricas**	
100	Fluxos		**Normal**	**Ataque**	**Acurácia**	100.00%		**Normal**	**Ataque**	**Acurácia**	100.00%
		Normal	116955	0	**Precisão**	100.00%	**Normal**	173759	0	**Precisão**	Não se aplica
		Ataque	0	141193	**Taxa de Erro**	0.00%	**Ataque**	0	0	**Taxa de Erro**	0.00%
					Recall	100.00%				**Recall**	Não se aplica
					F1-Score	100.00%				**F1-Score**	Não se aplica

Os Gráficos de todos os resultados obtidos de todas as janelas por fluxo e por tempo do Ambiente Real são apresentados e comentados na sequência, com os resultados individuais lado a lado, servindo de comparação no projeto de pesquisa das diversas possibilidades de aplicabilidade do mecanismo híbrido em cada uma das soluções híbridas e da própria abordagem tradicional da Entropia. Primeiro são apresentadas e comentadas as janelas por fluxo da menor para a maior e depois janelas por tempo com a mesma lógica. Na sequência os comentários gerais, finalizando a apresentação dos resultados com considerações referentes aos dois Ambientes.

Na Figura 6.27 são apresentados os resultados dos dados do Ambiente Real com janelas de 100 fluxos para o conjunto de dados com ataque à esquerda e sem ataque à direita da Figura. Entropia obteve nos dados com ataque, 97.440% de acurácia e 97.622% de F1-Score, e sem ataque ficou em segundo lugar. ML-Entropia nos dados com ataque, 99.090% de acurácia e 99.173% de F1-Score, nos dados sem ataque foi inferior à Entropia. O SVR nos dados com ataque, obteve 97.223% de acurácia e 97.521% de F1-Score, nos dados sem ataque foi o pior resultado. O SVC nos dados com ataque ficou com 99.764%

de acurácia e 99.784% de F1-Score, e nos dados sem ataque foi o segundo pior. O *Random Forest* obteve 100% em todos os resultados nos dados com e sem ataque.

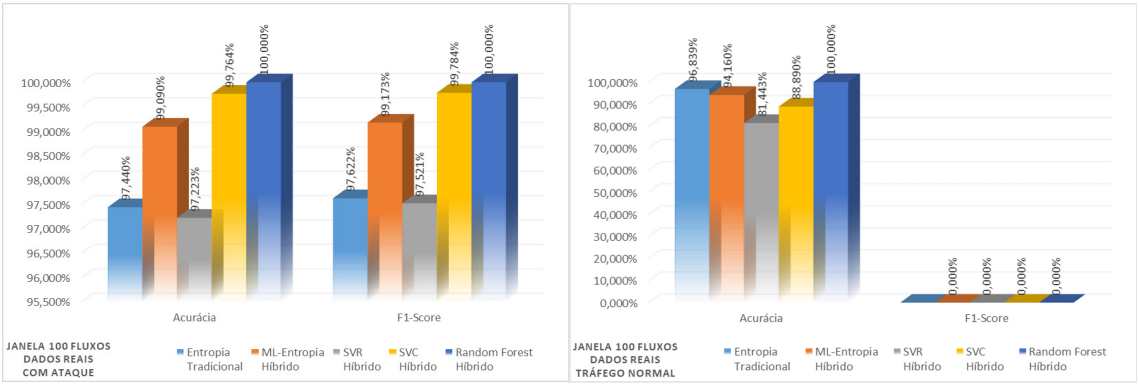

Figura 6.27: Detecção de Ataque no Ambiente Real com Janelas de 100 Fluxos.

Comentário da Figura 6.27: Com janelas de 100 fluxos Entropia e SVR obtiveram os piores resultados nos dados com ataque com Taxas de Erros acima de 2%, enquanto o restante ficou abaixo de 1%. Nos dados sem ataque Entropia ficou em segundo lugar mas com resultado inferior ao seu desempenho nos dados com ataque abaixo de 97% e o SVR ficou com o pior resultado. ML-Entropia foi melhor que Entropia e SVR nos dados com ataque e nos dados sem ataque manteve melhor desempenho que o SVR e SVC. O SVC foi quase ótimo nos dados com ataque e muito ruim nos dados sem ataque, semelhante ao que ocorreu nos resultados do ambiente emulado por fluxos. O *Random Forest* obteve 100% nos dois tipos de dados.

Na Figura 6.28 são apresentados os resultados dos dados do Ambiente Real com janelas de 125 fluxos para o conjunto de dados com ataque à esquerda e sem ataque à direita da Figura. Entropia obteve nos dados com ataque, 98.255% de acurácia e 98.390% de F1-Score, nos dados sem ataque foi o segundo melhor resultado. ML-Entropia obteve nos dados com ataque, 99.013% de acurácia e 99.104% de F1-Score, nos dados sem ataque ficou abaixo de Entropia. O SVR nos dados com ataque, 97.173% de acurácia e 97.477% de F1-Score, nos dados sem ataque foi o pior resultado. O SVC nos dados com ataque, 99.887% de acurácia e 99.869% de F1-Score, nos dados sem ataque foi o segundo pior. O *Random Forest* obteve 100% nos dois tipos de dados.

Comentário da Figura 6.28: Esta janela de 125 fluxos representou um início de degradação para o desempenho de ML-Entropia e SVR, mas um pequeno ganho para Entropia. O SVC e o *Random Forest* obtiveram resultados semelhantes à janela anterior.

Na Figura 6.29 são apresentados os resultados dos dados do Ambiente Real com janelas de 150 fluxos para o conjunto de dados com ataque à esquerda e sem ataque à direita da Figura. Entropia obteve nos dados com ataque, 98.014% de acurácia e 98.165% de F1-Score, nos dados sem ataque foi o segundo melhor resultado. ML-Entropia obteve nos

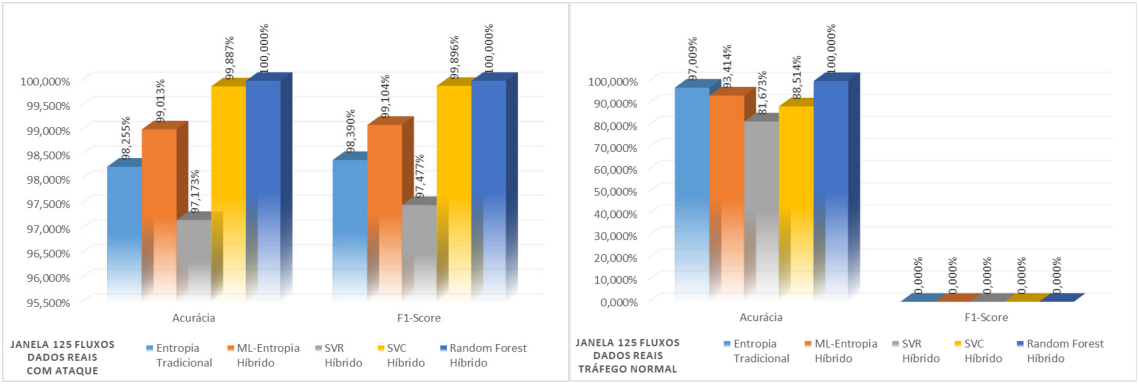

Figura 6.28: Detecção de Ataque no Ambiente Real com Janelas de 125 Fluxos.

dados com ataque, 98.858% de acurácia e 98.964% de F1-Score, nos dados sem ataque ficou abaixo de Entropia. O SVR obteve nos dados com ataque, 97.806% de acurácia e 98.030% de F1-Score, nos dados sem ataque foi o pior resultado. O SVC nos dados com ataque obteve 99.928% de acurácia e 99.934% de F1-Score, nos dados sem ataque foi o segundo pior. O *Random Forest* obteve 100% nos dois tipos de dados.

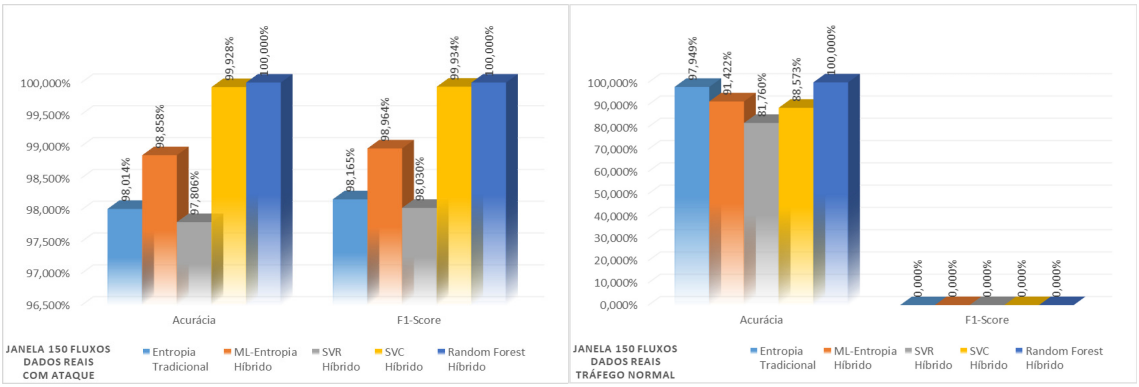

Figura 6.29: Detecção de Ataque no Ambiente Real com Janelas de 150 Fluxos.

Comentário da Figura 6.29: A degradação de desempenho se confirmou na janela de 150 fluxos acentuando a queda de resultados com exceção de Entropia que obteve pequeno aumento nos dois tipos de dados, mas irrelevantes no aspecto geral. O *Random Forest* manteve ótimo desempenho das janelas anteriores com 100% de resultado.

Na Figura 6.30 são apresentados os resultados dos dados do Ambiente Real com janelas de 3 segundos para o conjunto de dados com ataque à esquerda e sem ataque à direita da Figura. Entropia obteve nos dados com ataque, 91.385% de acurácia e 92.254% de F1-Score, nos dados sem ataque foi melhor que ML-Entropia. ML-Entropia obteve nos dados com ataque, 96.561% de acurácia e 97.097% de F1-Score, nos dados sem ataque ficou abaixo de Entropia. O SVR obteve nos dados com ataque 95.800% de acurácia e 96.493% de F1-Score, nos dados sem ataque foi o pior resultado. O SVC nos dados com

ataque obteve 99.470% de acurácia e 99.540% de F1-Score, nos dados sem ataque foi o segundo pior. O *Random Forest* obteve 100% nos dois tipos de dados.

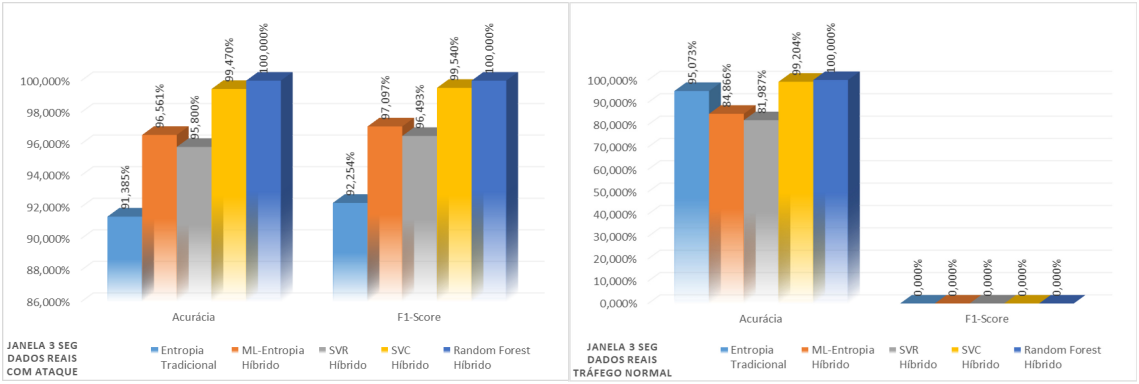

Figura 6.30: Detecção de Ataque no Ambiente Real com Janelas de 3 Segundos.

Comentário da Figura 6.30: Dentre as janelas por tempo a janela de 3 segundos, no ambiente Real, apresentou queda de desempenho geral em relação aos resultados das janelas por fluxo com exceção de *Random Forest* que manteve os ótimos números.

Na Figura 6.31 são apresentados os resultados dos dados do Ambiente Real com janelas de 6 segundos para o conjunto de dados com ataque à esquerda e sem ataque à direita da Figura. Entropia obteve nos dados com ataque, 95.672% de acurácia e 96.263% de F1-Score, nos dados sem ataque foi melhor que ML-Entropia. ML-Entropia obteve nos dados com ataque, 96.571% de acurácia e 97.114% de F1-Score, nos dados sem ataque ficou abaixo de Entropia. O SVR obteve nos dados com ataque 96.169% de acurácia e 96.791% de F1-Score, nos dados sem ataque foi o pior resultado. O SVC nos dados com ataque obteve 99.467% de acurácia e 99.537% de F1-Score, nos dados sem ataque foi o segundo melhor. O *Random Forest* obteve 96.169% de acurácia e 96.791% de F1-Score, nos dados sem ataque foi o melhor resultado.

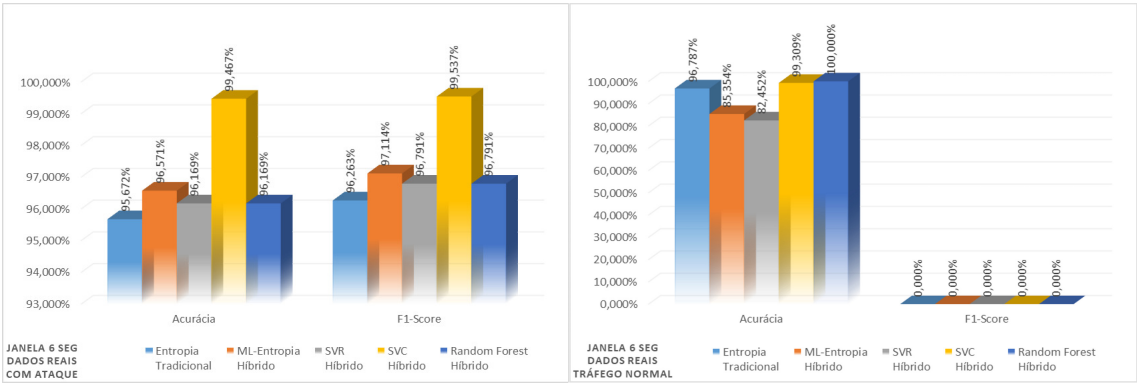

Figura 6.31: Detecção de Ataque no Ambiente Real com Janelas de 6 Segundos.

Comentário da Figura 6.31: Houve uma pequena melhora para Entropia, ML-Entropia e SVR, com ML-Entropia com melhor resultado que Entropia e SVR nos dados com ataque, mas com queda de desempenho nos dados sem ataque, onde Entropia obteve melhor resultado. O SVC foi quase ótimo nos dois tipos de dados, confirmando seu desempenho melhor em janelas por tempo, indicada no ambiente emulado. O *Random Forest* começou a degradar seu resultado geral nos dados com ataque.

Na Figura 6.32 são apresentados os resultados dos dados do Ambiente Real com janelas de 9 segundos para o conjunto de dados com ataque à esquerda e sem ataque à direita da Figura. Entropia obteve nos dados com ataque, 96.309% de acurácia e 96.853% de F1-Score, nos dados sem ataque foi o terceiro melhor. ML-Entropia obteve nos dados com ataque, 96.472% de acurácia e 97.025% de F1-Score, nos dados sem ataque ficou abaixo de Entropia. O SVR obteve nos dados com ataque 96.216% de acurácia e 96.822% de F1-Score, nos dados sem ataque foi o pior resultado. O SVC nos dados com ataque obteve 99.494% de acurácia e 99.561% de F1-Score, nos dados sem ataque foi o segundo melhor. O *Random Forest* obteve 100% nos dois tipos de dados.

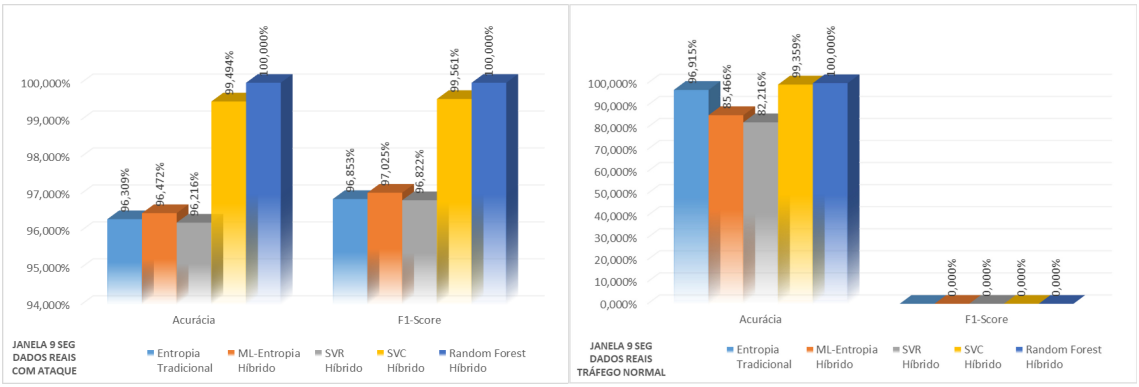

Figura 6.32: Detecção de Ataque no Ambiente Real com Janelas de 9 Segundos.

Comentário da Figura 6.32: O desempenho geral foi melhor para todas as soluções em relação às janelas por tempo anteriores. ML-Entropia foi melhor que Entropia e SVR nos dados com ataque, mas inferior a Entropia nos dados sem ataque. O SVC obteve desempenho quase ótimo nos dois tipos de dados. O *Random Forest* foi ótimo nos dois tipos de dados.

Comentário Geral para os resultados do Ambiente Real: De modo geral os resultados de janelas por fluxos também foram melhores para todas as soluções com exceção do SVC que confirmou seu desempenho no ambiente emulado com melhores resultados nas janelas definidas por tempo. O *Random Forest* obteve ótimo desempenho em quase todas as janelas por fluxo e tempo, com exceção apenas nas janelas de 6 segundos.

Uma consideração importante percebida pela análise dos resultados dos dois Ambientes (Emulado e Real) é que janelas muito pequenas e muito grandes têm os piores resultados.

O motivo disto é que observando as imagens nota-se que em janelas muito pequenas a separação dos dados é dificultada pela grande concentração dos dados, ou seja, eles se misturam. Já em janelas muito grandes, os dados se dispersam demais e a separação dos dados também é prejudicada por causar de uma má definição de hiperplano, conforme demonstrado na Figura 6.17.

Outra consideração reside em dois pontos: (Primeiro ponto) É que o fato dos atributos de seleção para escolha de melhores Limiares na detecção do ataque terem ocorrido primeiramente nas janelas por fluxo. (Segundo ponto) A necessidade de se definir uma regra de igualdade para as aferições das diversas soluções a fim de se manter um mesmo Limiar-X, evitando atributos que só seriam possíveis em janelas temporais. Estes dois pontos podem ter beneficiado as janelas por fluxo em detrimento das janelas temporais nessa análise. Contudo há de se ressaltar que esses dois pontos foram fundamentais para a comprovação desejada e sem essa regra não teríamos a possibilidade de comparar e comprovar se realmente a abordagem aqui proposta realmente surtiria efeito positivo, ou seja, melhorando o valor do Limiar-X na detecção do ataque. Testar o melhor desempenho usando os parâmetros de limiar mais apropriados em janelas temporais para ML-Entropia é uma questão em aberto.

A melhoria ocorrida por meio desta proposta de pesquisa ficou evidente tanto em relação à Entropia quanto em relação ao SVR por si só, com pequena diferença de resultado nos dados sem ataque melhores para Entropia. Contudo esta diferença pode estar relacionada à priorização de redução dos FN's, que será melhor detalhado na Seção 6.3 a seguir. Os FN's impactam diretamente na métrica *Recall*, enquanto os FP's impactam diretamente na métrica Precisão, assim sendo como só temos a medida de Precisão nos dados sem ataque, visto que é impossível termos FN's nesse conjunto de dados, dai temos a diferença encontrada nos dados sem ataque.

6.2.3 Comparação dos Resultados com a Literatura

Primeiro se faz necessário diferenciar as abordagens dos três trabalhos de referência na literatura de Agarwal e Mittal 2012 [8], de Dehkordi et al. 2020 [6] e depois deles Aung e Htaik 2020 [7] que também apresentaram uma abordagem híbrida utilizando Entropia e *Machine Learning*, para um melhor entendimento da comparação. No primeiro trabalho e Agarwal e Mittal 2012 [8] os autores utilizaram dados sintéticos apenas e realizam a detecção do ataque por classificação usando o algoritmo *Support Vector Machine (SVM)*, comparando com Entropia na detecção do ataque. Diferente desta abordagem aqui Entropia é aprimorada com um melhor coeficiente de limiar em ML-Entropia e os algorítimos de ML aqui usados são todos diferentes e mais eficientes que SVM comum. No segundo trabalho em comparação, de Dehkordi et al. 2020 [6] os autores também utilizaram dados

sintéticos apenas, em quatro *Datasets* com ataque e um sem ataque, realizando a detecção do ataque com primeiro filtro por entropia e os suspeitos no segundo filtro por classificação usando ML nos algoritmos de classificação: BayesNet, J48, RandomTree, logistic regression e REPTree, comparando com trabalhos na literatura, valendo ainda ressaltar que neste trabalho de comparação os autores submetem apenas os suspeitos da entropia ao ML como segunda fase de filtragem o que também difere da abordagem aqui apresentada. No terceiro e último trabalho Aung e Htaik 2020 [7] utilizaram SVM no plano de controle com o conjunto de dados sintéticos, ASNM-NPBO, a solução consulta uma lista de bloqueio como primeiro filtro, depois classifica, como segundo filtro, em normal ou anormal e neste caso os suspeitos são submetidos a detecção do ataque por limiar, também diferente da abordagem aqui proposta.

Aqui foi apresentado um mecanismo híbrido utilizando Entropia e *Machine Learning (ML)* onde o ML, utilizando um algoritmo de Regressão (SVR), retorna o coeficiente de previsão para entropia em ML-Entropia, melhorando as métricas na detecção do ataque com um melhor limite de separação os dados legítimos do espúrio obtido. Além desta técnica este trabalho de pesquisa também apresentou soluções híbridas com outros algoritmos de classificação: SVC e *Random Forest* e o próprio SVR que é de regressão, realizando a detecção do ataque, todos diferentes dos utilizados pelos autores citados e mais eficientes que SVM comum, servindo como técnicas alternativas e comparação adicional. Comparando estes resultados apresentados com os resultados dos trabalhos de:

- Agarwal e Mittal 2012 [8] onde os autores obtiveram Precisão de 97.25% e Taxa de Erro de 2.75%;
- Dehkordi et al. 2020 [6] com resultado de 99.85% de Acurácia e 0,1% de Taxa de Falso Positivo (FPR), neste caso os autores utilizam apenas FP e não Taxa de Erro que considera o total de erros entre FN e FP;
- Depois deles o trabalho de Aung e Htaik 2020 [7] com resultado de 76.12% de Acurácia e 0.05% de Taxa de Falso Positivo, também com FPR;
- Para ser justo são comparados resultados sintéticos com resultados sintéticos, ou seja, obtidos do ambiente emulado, onde em ML-Entropia no pior resultado, com janelas de 25 fluxos, uma Precisão de 99.969% e Taxa de Erro de 0.567% ainda assim aqui os resultados foram superiores aos três trabalhos, além disso todos os outros algoritmos que foram adicionados na pesquisa obtiveram melhor desempenho que os resultados deles;
- Já se comparado o melhor resultado obtido em ML-Entropia com janelas de 100 fluxos também do ambiente emulado a superioridade é ainda maior pois foi atingido 100% de Acurácia, sem falar dos outros algoritmos aqui apresentados;

- Agora os resultados do ambiente real, que não podem servir de comparação, visto seus conjuntos de dados são sintéticos. Ainda assim apresentamos como evidência da validação do método. Estes também foram superiores com resultados para o SVC em 99.928% de Acurácia e 0.072% de Taxa de Erro e *Random Forest* com 100% de acurácia. ML-Entropia ficou com 99.090% de Acurácia e 0.910% de Taxa de Erro, acima dos 99% e ainda com uma taxa de erro inferior à sintética comparada, além dos outros algoritmos híbridos que também tiveram resultados melhores do que os autores citados.

6.3 Análise e Avaliação dos Resultados

Os resultados comprovam que o uso de ML-Entropia como mecanismo híbrido de detecção de ataque DDoS em SDN, combinando as técnicas de Entropia e de ML é eficiente e melhorou o limiar, em relação a entropia e SVR sozinhos, na detecção com seu coeficiente sendo fornecido por um Algoritmo de Regressão, neste caso o SVR com RBF e com um limiar dinâmico acompanhando o estado atual da rede com o coeficiente sempre encontrando o melhor hiperplano de separação dos dados, por meio desta técnica.

Também demonstrou que os algoritmos SVR, SVC e *Random Forest* híbridos são técnicas alternativas viáveis de mecanismo híbrido de detecção de ataque, principalmente o *Random Forest* híbrido que apresentou ótimos resultados em todas as janelas por fluxo e por tempo nos dois ambientes Emulado e Real, se mostrando como uma ótima alternativa híbrida para ambientes onde a detecção fora do Plano de Dados é indicada.

A ML-Entropia, conseguiu melhorar não somente os resultados da Entropia tradicional sozinha, mas também os resultados do SVR sozinho, quando comparamos os resultados, demonstrando a melhoria não apenas de uma técnicas (Entropia) mas das duas técnicas (Entropia e SVR) que foram combinadas. Além disso apresentou resultados ótimos no ambiente Emulado e quase ótimos no ambiente Real, apresentando-se como uma ótima alternativa de implementação, não somente fora do Plano de Dados, mas também para ambiente onde a detecção no Plano de Dados é necessária. Recomendamos para este caso, que o treino/teste de ML rodando fora do Plano de Dados e fornecendo à entropia o coeficiente, e ML-Entropia no Plano de Dados, uma vez que apresentamos aqui abordagens de entropia na literatura rodando no Plano de Dados. Comparando aos trabalhos semelhantes no estado da arte fomos superiores tanto no ambiente Emulado quanto no ambiente Real, conforme já apresentado na Subseção 6.2.3. A avaliação dos resultados obtidos e já comentados segue com as seguintes considerações:

- Os resultados obtidos no Ambiente Emulado (MiniNet) são coerentes com os resultados obtidos no Ambiente Real;

- A abordagem aqui apresentada melhorou os coeficientes do Limiar-X, consequentemente melhorou a separação dos dados espúrios e legítimos, com redução das Taxas de Erros em comparação com Entropia tradicional. Comprovado com os resultados obtidos tanto no ambiente Emulado quanto no Ambiente Real, demonstrando a viabilidade de uso da proposta de solução em Ambientes Reais;
- ML-Entropia melhorou a detecção em relação Entropia e SVR sozinhos, indicando que quanto melhores forem os resultados de Entropia e do algoritmo de Regressão escolhido sozinhos, melhores serão os resultados finais na abordagem de ML-Entropia. Vale ressaltar que outros algoritmos de regressão podem ser utilizados na abordagem uma vez que é característica desses algoritmos a possibilidade de retornar o coeficiente de previsão. Da mesma forma Entropia de Renyis e φ-Entropia;
- O método permitiu um limiar dinâmico e ajustável ao estado atual da rede;
- A abordagem proposta obteve resultados superiores aos três trabalhos de referência na literatura em Ambientes Emulado e Real;
- A coleta de fluxos de dados e análise fora dos Planos de Dados e Controle não impactou no desempenho geral da rede e desonerou o Plano de Controle;
- Os Limiares-X foram escolhidos com foco no melhor desempenho geral na detecção do Ataque com menor redução possível da Taxa de Erro (FP e FN) para Entropia sozinha, em caso de impasse os FN's foram priorizados. Por acreditar que os FN's são mais prejudiciais à rede do que as FP's, pois se for bloqueado um usuário legítimo (FP) como se fosse um ataque o prejuízo será apenas relativo àquela atividade que deixou de ser executada, mas se for liberado um verdadeiro atacante (FN) como se fosse um usuário legítimo, ele pode causar danos à toda a rede em um único ataque.

  Comentário: Isto justificaria a diferença de ML-Entropia e Entropia nos dados sem ataque, pois os FP's influenciam na Precisão, enquanto os FN's influenciam no *Recall*, mas se for observada a média harmônica das duas métricas que é dada pela *F1-Score* os resultados de ML-Entropia foram superiores em todos, mas ela não pode ser calculada nos dados sem ataque por que não existem FN's, ainda assim os percentuais deste experimento foram altos e promissores o que não tira o mérito da solução, mas aponta para um possível ajuste que é encontrar uma maior redução, na análise dos Limiares-X, que melhor reduzam os índices de FP;
- É percebido que a diferença entre os resultados de ML-Entropia e Entropia sozinha diminuem à medida que os tamanhos das janelas vão aumentando. Além disso a Entropia sozinha precisa de janelas maiores para melhorar seu desempenho geral, enquanto ML-Entropia além melhorar os resultados de Entropia sozinha traz bons

resultados com janelas menores. Isso é considerado uma vantagem de ML-Entropia em relação à Entropia, pois com janelas menores, principalmente por tempo, implica em dizer que um ataque pode ser detectado mais cedo;

- ML-Entropia obteve desempenho melhor nos Dados com Ataque que todos os outros com exceção de *Random Forest* híbrido que obteve ótimos resultados em todos os testes;
- Os resultados em janelas temporais não foram tão interessantes, para Entropia, ML-Entropia e SVR, quanto em janelas por fluxo, contudo vale ressaltar que os parâmetros iniciais de definição de Limiares-X foram feitos com base nas janelas por fluxo para preservar os parâmetros de comparação na avaliação de desempenho da solução proposta e garantir que os resultados são justos e válidos cientificamente. Ainda assim o SVC e *Random Forest* híbridos obtiveram resultados ótimos em janelas por tempo. Indica que o SVC é mais adequado para este tipo de janela.
- A redução das diferenças entre ML-Entropia e Entropia se observa em janelas temporais, nos dados sem Ataque, indicando que um aprimoramento nessas janelas podem refletir em resultados finais melhores ainda;
- ML-Entropia demonstrou ser uma alternativa viável dentro e fora do Plano de Dados com as considerações já feitas;
- *Random Forest* e SVC são alternativas viáveis fora dos Planos de Dados e Controle;
- A abordagem proposta serve não apenas para SDN mas também para Redes Tradicionais. Como diferencial da solução, ela já estará pronta e habilitada para explorar todo o potencial existente nas Redes SDN, em termos de mitigação e implementação em qualquer de seus planos, ao contrário de soluções que só visam Redes Tradicionais ou só Redes SDN;
- Por fim ressalta-se que com a técnica de aprendizado de máquinas quanto mais tempo a solução estiver rodando em ambiente de produção é esperado que melhores sejam seus resultados.

Capítulo 7

Considerações Finais

A SDN foi apresentada neste trabalho como uma realidade em expansão no mercado. Uma revisão do estado da arte foi feita, apresentando o desafio dos ataques DDoS como um problema relevante a ser mitigado, em SDN e Redes tradicionais. Diversas abordagens de pesquisas que trataram do tema DDoS foram apresentadas e uma questão é comum, o de estabelecer um limite para determinar melhor se um tráfego é espúrio ou não, com um limiar adaptativo ao estado da rede.

Foi proposto um método híbrido com quatro técnicas distintas que foram implementadas em Ambiente Emulado e Real. A primeira delas a ML-Entropia que faz uso dos cálculos de entropia como entrada para o SVR-RBF que por sua vez fornece o coeficiente de previsão à ML-Entropia. Além desta técnica três outras usando SVR, SVC e *Random Forest* também híbridas e viáveis de implementação no mundo real, comparando-as entre si e com o estado da arte. Todas melhoraram suas métricas de detecção em relação a Entropia tradicional e a literatura, ao mesmo tempo, entregaram uma solução com um limiar dinâmico e ajustável ao estado atual da rede. Contudo ressalta-se que este método não resolve outros desafios como de *Flash Events* [54] que são tráfegos legítimos que se assemelham ao ataque DDoS e nem foi testado com outros tipos de ataque, como de baixa taxa e lentos, para aferir sua efetividade na detecção.

Este trabalho implementou em SDN, mas a solução também é viável em Redes Tradicionais. Como diferencial a solução está pronta e habilitada para explorar todo o potencial existente nas Redes SDN, em termos de mitigação e implementação em qualquer um de seus planos, ao contrário de soluções que só visam Redes Tradicionais ou só SDN.

Em pesquisas futuras, pretende-se estender este trabalho considerando ataques dirigidos ao controlador e também estender o método considerando múltiplos controladores. Além disso, pretende-se testar o método proposto para detectar ataques DDoS de baixa taxa ou outros tipos de ataques e usar medidas de entropia de informação generalizada, como a Entropia de Renyis e φ-Entropia, para verificar os ganhos.

Referências

[1] Ching-Hao, C. e Dr. Y. Lin: *Openflow version roadmap*. IEEE, 2015. x, 10

[2] Fayyad, U., G. Piatetsky-Shapiro e P. Smyth: *The kdd process for extracting useful knowledge from volumes of data*. Commun. ACM, 39(11):27–34, novembro 1996, ISSN 0001-0782. `http://doi.acm.org/10.1145/240455.240464`. x, 16, 17

[3] A. C. Müller, S. Guido: *Introduction to Machine Learning with Python*, volume Released October 2016. O'Reilly Media, Inc., oct 2016. ISBN: 9781449369415. x, 18, 20, 21, 52

[4] Filgueiras, P. R.: *Regressão por vetores de suporte aplicado na determinação de propriedades físico-químicas de petróleo e biocombustíveis*. Tese de Doutoramento, Repositório da Produção Científica e Intelectual da Unicamp, 2014. `http://repositorio.unicamp.br/handle/REPOSIP/249325`, [Online; accessed 16-Maio-2019]. x, 19, 20, 52

[5] Cortes, C. e V. Vapnik: *Support-vector networks*. Machine Learning, 20(3):273–297, Sep 1995, ISSN 1573-0565. `https://doi.org/10.1007/BF00994018`. x, 20, 52

[6] Dehkordi, A. B., M. Soltanaghaei e F. Z. Boroujeni: *The ddos attacks detection through machine learning and statistical methods in sdn*. The Journal of Supercomputing, páginas 1–33, 2020. x, 1, 23, 24, 32, 83, 84

[7] Aung, K. M. e N. M. Htaik: *Anomaly detection in sdn's control plane using combining entropy with svm*. Em *2020 17th IEEE Conference, Computer, Telecommunications and Information Technology (ECTI-CON)*, páginas 122–126, June 2020. x, 1, 23, 26, 27, 32, 83, 84

[8] Agarwal, B. e N. Mittal: *Hybrid approach for detection of anomaly network traffic using data mining techniques*. Procedia Technology, 6, dezembro 2012. x, 1, 3, 23, 25, 30, 32, 83, 84

[9] Sezer, S., S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller e N. Rao: *Are we ready for sdn? implementation challenges for software-defined networks*. IEEE Communications Magazine, 51(7):36–43, July 2013, ISSN 0163-6804. 1

[10] Liu, Y., B. Zhao, P. Zhao, P. Fan e H. Liu: *A survey: Typical security issues of software-defined networking*. China Communications, 16(7):13–31, July 2019. 1, 9, 10, 12, 14, 31

[11] Mousavi, S. M. e M. St-Hilaire: *Early detection of ddos attacks against sdn controllers.* Em *2015 International Conference on Computing, Networking and Communications (ICNC)*, páginas 77–81, Feb 2015. 1, 3, 11, 13, 15, 25, 31, 32, 38

[12] EdicaoMS.com.br: *Maior ataque ddos da história chega a 1,35 tbps e tem como alvo github.* http://www.edicaoms.com.br/imprimir/tecnologia/maior-ataque-ddos-da-historia-chega-a-1-35-tbps-e-tem-como-alvo-o-github/. 1, 3, 13

[13] Sambandam, N., M. Hussein, N. Siddiqi e C. Lung: *Network security for iot using sdn: Timely ddos detection.* 2018 IEEE Conference on Dependable and Secure Computing (DSC), páginas 1–2, Dec 2018. 1, 3, 13, 15, 26, 31, 32

[14] Braga, R., E. de Souza Mota e A. Passito: *Lightweight ddos flooding attack detection using nox/openflow.* Em *LCN*, páginas 408–415. IEEE Computer Society, 2010, ISBN 978-1-4244-8387-7. `http://dblp.uni-trier.de/db/conf/lcn/lcn2010.html#BragaMP10`. 1, 3, 27, 32

[15] Prasetiawan, D., M. Abdurohman e F. A. Yulianto: *Improving distributed denial of service (ddos) detection using entropy method in software defined network (sdn).* ComTech: Computer, Mathematics and Engineering Applications, 8:215, dezembro 2017. 1, 3, 15, 25, 31, 32

[16] Singh, Jagdeep e Sunny Behal: *Detection and mitigation of ddos attacks in sdn: A comprehensive review, research challenges and future directions.* Computer Science Review, 37:100279, 2020, ISSN 1574-0137. `http://www.sciencedirect.com/science/article/pii/S1574013720301647`. 1, 3, 4, 15, 22

[17] Caraka, R., H. Yasin e A. Waridi: *Peramalan crude palm oil (cpo) menggunakan support vector regression kernel radial basis.* MATEMATIKA, 1:43–57, junho 2017. 1, 19, 39

[18] Koponen, T., K. Amidon, P. Balland, M. Casado, A. Chanda, B. Fulton, I. Ganichev, J. Gross, P. Ingram, E. Jackson, A. Lambeth, R. Lenglet, S. Li, A. Padmanabhan, J. Pettit, B. Pfaff, R. Ramanathan, S. Shenker, A. Shieh, J. Stribling, P. Thakkar, D. Wendlandt, A. Yip e R. Zhang: *Network virtualization in multi-tenant datacenters.* Em *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, páginas 203–216, Seattle, WA, 2014. USENIX Association, ISBN 978-1-931971-09-6. `https://www.usenix.org/conference/nsdi14/technical-sessions/presentation/koponen`. 3, 7

[19] Nunes, B. A. A., M. Mendonca, X. Nguyen, K. Obraczka e T. Turletti: *A survey of software-defined networking: Past, present, and future of programmable networks.* IEEE Communications Surveys Tutorials, 16(3):1617–1634, Third 2014, ISSN 1553-877X. 3, 8, 9, 10

[20] Shin, S. e G. Gu: *Attacking software-defined networks: A first feasibility study.* HotSDN 2013 - Proceedings of the 2013 ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, páginas 165–166, agosto 2013. 3

[21] Kreutz, D., F. M.V. Ramos e P. Verissimo: *Towards secure and dependable software-defined networks*. Em *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, páginas 55–60, New York, NY, USA, 2013. ACM, ISBN 978-1-4503-2178-5. http://doi.acm.org/10.1145/2491185.2491199. 3

[22] Vishwakarma, R. e A. K. Jain: *A survey of ddos attacking techniques and defence mechanisms in the iot network.* Telecommunication Systems, 73(1):3–25, 2020, ISSN 10184864. http://search-ebscohost-com.ez54.periodicos.capes.gov.br/login.aspx?direct=true&db=iih&AN=141078329&lang=pt-br&site=ehost-live. 3, 13

[23] Magazine, Security: *570% increase in bit-and-piece ddos attacks in 2020.* https://www.securitymagazine.com/articles/93456-increase-in-bit-and-piece-ddos-attacks-in-2020, 2021. acessado em janeiro de 2021. 3, 13

[24] Mansfield-Devine, S.: *The evolution of ddos.* Computer Fraud Security, 2014, outubro 2014. 3, 13

[25] Carvalho, L. F., T. Abrão, L. S. Mendes e M. L. Proença: *An ecosystem for anomaly detection and mitigation in software-defined networking.* Expert Systems with Applications, 104:121 – 133, 2018, ISSN 0957-4174. http://www.sciencedirect.com/science/article/pii/S0957417418301726. 3, 21, 24, 32

[26] Dey, S., Md Mahbubur R. e Md R. Uddin: *Detection of flow based anomaly in openflow controller: Machine learning approach in software defined networking.* 4th International Conference on Eletrical Engineering and Information Communication Technology, páginas 416–421, setembro 2018. 3, 26, 32

[27] Giotis, K., C. Argyropoulos, G. Androulidakis, D. Kalogeras e V. Maglaris: *Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments.* Computer Networks, 62:122 – 136, 2014, ISSN 1389-1286. http://www.sciencedirect.com/science/article/pii/S1389128613004003. 4

[28] Kreutz, D., F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky e S. Uhlig: *Software-defined networking: A comprehensive survey.* IEEE, 2015. 7, 8, 9

[29] McKeown, N., T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, S. Shenker J. Rexford e J. Turner: *Openflow: Enabling innovation in campus networks.* ACM SIGCOMM Computer Communication, 38, 2008. 9, 10

[30] EnterpriseNetworkingPlanet, By:Sean Michael Kerner: *ONF-Announces-Stratum-Project-to-Redefine-SDN.* http://www.enterprisenetworkingplanet.com/netsp/openflow-is-the-past-as-onf-announcesstratum-project-to-redefine-sdn.html, March 13, 2018. 10, 11

[31] Estinet: *Estinet - the network simulator and emulator that supports sdn/openflow.* http://www.estinet.com/ns/, acesso em 2019-09-11. 11

[32] Wang, S., C. Chou e C. Yang: *Estinet openflow network simulator and emulator*. IEEE Communications Magazine, 51(9):110–117, Sep. 2013. 11

[33] NSnam: *Ns-3 - network simulator*. https://www.nsnam.org/, acesso em 2019-09-11. 11

[34] Mininet: *Mininet - an instant virtual network on your laptop (or other pc)*. http://mininet.org/, acesso em 2019-09-11. 11

[35] Trabelsi, Z., S. Zeidan e M. M. Masud: *Hybrid mechanism towards network packet early acceptance and rejection for unified threat management*. IET Information Security, 11(2):104–113, 2017. 12

[36] Dzurenda, P., Z. Martinasek e L. Malina: *Network protection against ddos attacks*. International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems, 4, março 2015. 12

[37] Gondim, J. J. C., R. de Oliveira Albuquerque e A. L. Sandoval Orozco: *Mirror saturation in amplified reflection distributed denial of service: A case of study using snmp, ssdp, ntp and dns protocols*. Future Generation Computer Systems, 108:68 – 81, 2020, ISSN 0167-739X. http://www.sciencedirect.com/science/article/pii/S0167739X19322745. 13, 14

[38] Mirkovic, J. e P. Reiher: *A taxonomy of ddos attack and ddos defense mechanisms*. ACM SIGCOMM Computer Communication Review, 34, maio 2004. 13

[39] Alomari, E., S. Manickam, B. B. Gupta, S. Karuppayah e R. Alfaris: *Botnet-based distributed denial of service (ddos) attacks on web servers: Classification and art*. International Journal of Computer Applications, 49(7):24–32, Jul 2012, ISSN 0975-8887. http://dx.doi.org/10.5120/7640-0724. 14

[40] Pineda, J. O. C.: *A entropia segundo claude shannon: o desenvolvimento do conceito fundamental da teoria da informação*. https://tede2.pucsp.br/handle/handle/13330, acesso em 2019-09-10. 15

[41] Cui, J., M. Wang, Y. Luo e H. Zhong: *Ddos detection and defense mechanism based on cognitive-inspired computing in sdn*. Future Generation Computer Systems, 97:275 – 283, 2019, ISSN 0167-739X. http://www.sciencedirect.com/science/article/pii/S0167739X18324336. 15, 31

[42] David, J. e C. Thomas: *Detection of distributed denial of service attacks based on information theoretic approach in time series models*. Journal of Information Security and Applications, 55:102621, 2020, ISSN 2214-2126. http://www.sciencedirect.com/science/article/pii/S2214212620307869. 15

[43] A. Simon, H: *Why should machines learn? in michalski*. Machine Learning: An Artificial Intelligence Approach. Morgan Kaufmann, 1983. 16

[44] Nguyen, T. T. T. e G. Armitage: *A survey of techniques for internet traffic classification using machine learning*. IEEE Communications Surveys Tutorials, 10(4):56–76, Fourth 2008, ISSN 1553-877X. 16

[45] Ribeiro, V. G., S. R. Silveira, A. da Silveira, R. Atkinson e J. Zabadal: *O emprego de técnicas de mineração de dados para definição de estratégias em processos de divulgação científica em periódicos de design.* Strategic Design Research Journal, 6(2):85–94, 2013, ISSN 19842988. http://search-ebscohost-com.ez54.periodicos.capes.gov.br/login.aspx?direct=true&db=aph&AN=99468418&lang=pt-br&site=ehost-live. 16

[46] Antunes, J. L. F. e M. R. A. Cardoso: *Uso da análise de séries temporais em estudos epidemiológicos.* Epidemiologia e Serviços de Saúde, 24(3):565–576, sep 2015, ISSN 1679-4974. http://www.iec.pa.gov.br/template_doi_ess.php?doi=10.5123/S1679-49742015000300024&scielo=S2237-96222015000300565, acesso em 2018-09-17. 17

[47] Baydaroglu Y., Ozlem e K. Kocak: *Svr-based prediction of evaporation combined with chaotic approach.* Journal of Hydrology, 508:356–363, novembro 2013. 19, 51

[48] Ko, C. e C. Lee: *Short-term load forecasting using svr (support vector regression)-based radial basis function neural network with dual extended kalman filter.* Energy, 49:413–422, janeiro 2013. 19

[49] Omrani, E., B. Khoshnevisan, S. Shamshirband, H. Saboohi, N. B. Anuar e M. H. N. Md Nasir: *Potential of radial basis function-based support vector regression for apple disease detection.* Measurement, 55:512 – 519, 2014, ISSN 0263-2241. http://www.sciencedirect.com/science/article/pii/S0263224114002541. 19, 20

[50] Ramedani, Z., M. Omid, A. Keyhani, S. Shamshirband e B. Khoshnevisan: *Potential of radial basis function based support vector regression for global solar radiation prediction.* Renewable and Sustainable Energy Reviews, 39:1005 – 1011, 2014, ISSN 1364-0321. http://www.sciencedirect.com/science/article/pii/S1364032114005607. 19, 21, 31, 52

[51] Gitoee, A., A. Faridi e J. France: *Mathematical models for response to amino acids: estimating the response of broiler chickens to branched-chain amino acids using support vector regression and neural network models.* Neural Computing and Applications, 30(8):2499–2508, Oct 2018, ISSN 1433-3058. https://doi.org/10.1007/s00521-017-2842-x. 20, 31

[52] Shin, S., P. Porras, V. Yegneswaran, M. Fong, G. Gu e M. Tyson: *Fresco: Modular composable security services for software-defined networks.* ISOC Network and Distributed System Security Symposium, February 2013. 23, 32

[53] Xu, Y. e Y. Liu: *Ddos attack detection under sdn context.* Em *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, páginas 1–9, April 2016. 28, 32

[54] Sun, G., W. Jiang, Y. Gu, D. Ren e H. Li: *Ddos attacks and flash event detection based on flow characteristics in sdn.* 15th IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS 2018, Auckland, New Zealand, páginas 27–30, novembro 2018. 28, 32, 88

[55] Lapolli, Â. C., J. Adilson Marques e L. P. Gaspary: *Offloading real-time ddos attack detection to programmable data planes*. Em *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, páginas 19–27, April 2019. 29, 32

[56] Boutaba, R., M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano e O. M. Caicedo: *A comprehensive survey on machine learning for networking: evolution, applications and research opportunities*. Journal of Internet Services and Applications, 9(1):16, Jun 2018, ISSN 1869-0238. `https://doi.org/10.1186/s13174-018-0087-2`. 39

[57] Prat, N., I. Wattiau e J. Akoka: *Artifact evaluation in information systems design science research ? a holistic view*. Proceedings - Pacific Asia Conference on Information Systems, PACIS 2014, junho 2014. 40

[58] Sonnenberg, C. e J. V. Brocke: *Evaluation patterns for design science research artefacts*. Practical Aspects of Design Science, páginas 71–83, janeiro 2012. 40, 41

[59] Peffers, K., M. Rothenberger, T. Tuunanen e R. Vaezi: *Design science research evaluation*. Em Peffers, K., M. Rothenberger e B. Kuechler (editores): *Design Science Research in Information Systems. Advances in Theory and Practice*, páginas 398–410, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg, ISBN 978-3-642-29863-9. 40

[60] Jain, R.: *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley, first edição, apr 1991. 685 pages. 42

[61] Graylog, Inc.: *Graylog open source*. https://docs.graylog.org/en/3.0/, 04-20-2019. 55

[62] Mishra, O., P. Lodhi e S. Mehta: *Document oriented nosql databases: An empirical study*. Em Panda, Brajendra, Sudeep Sharma e Nihar Ranjan Roy (editores): *Data Science and Analytics*, páginas 126–136, Singapore, 2018. Springer Singapore, ISBN 978-981-10-8527-7. 55

[63] Sourceforge, Slashdot Media. By fredpissarra: *T50 very fast network stress tool version 5.8.7*. https://sourceforge.net/projects/t50/, 2020. 56, 63

[64] ©2020 GitHub, Inc. By Telekom Security: *T-pot 20.06 runs on debian (stable)*. https://github.com/telekom-security/tpotceconcept, 2020. 61

[65] (IANA), Internet Assigned Numbers Authority: *Ip flow information export (ipfix) entities*. https://www.iana.org/assignments/ipfix/ipfix.xhtml, 2020. 61