UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Detecção de Ataques de Negação de Serviço Utilizando Aprendizado de Máquina

Eduardo da Costa da Silva

Orientador

Sidney Cunha de Lucena

RIO DE JANEIRO, RJ - BRASIL
SETEMBRO de 2016

Detecção de Ataques de Negação de Serviço Utilizando Aprendizado de Máquina

Eduardo da Costa da Silva

DISSERTAÇÃO APRESENTADA COMO REQUISITO PARCIAL PARA OBTENÇÃO DO TÍTULO DE MESTRE PELO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA DA UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO (UNIRIO). APROVADA PELA COMISSÃO EXAMINADORA ABAIXO ASSINADA.

Aprovada por:

Sidney Cunha de Lucena, D.Sc. - UNIRIO

Carlos Alberto Campos, D.Sc. - UNIRIO

Antônio Augusto de Aragão Rocha, D.Sc. - UFF

Guilherme de Melo Baptista Domingues, D.Sc. - UERJ

RIO DE JANEIRO, RJ - BRASIL
SETEMBRO de 2016

Silva, Eduardo da Costa da.
S586 Detecção de ataques de negação de serviço utilizando
aprendizado de máquina / Eduardo da Costa da Silva, 2016.
128 f.; 30 cm

Orientador: Sidney Cunha de Lucena
Dissertação (Mestrado em Informática) - Universidade Federal do
Estado do Rio de Janeiro, Rio de Janeiro, 2016.

1. Distributed Denial of Service (DDoS). 2. Aprendizado do
computador. 3. Entropia. 4. Ataques de negação de serviço.
5. Arquitetura lambda. I. Lucena, Sidney Cunha de.
II. Universidade Federal do Estado do Rio de Janeiro. Centro
de Ciências Exatas e Tecnológicas. Curso de Mestrado em Informática.
III. Título.

CDD – 006.31

Dedico esta dissertação à minha mãe, Maria de Lourdes da Costa da Silva.
A luta dela para que eu estudasse é a minha maior motivação para superar todos os meus desafios atuais e futuros.
A minha vitória, em qualquer sentido, é mais dela do que minha!

Agradecimentos

Primeiramente, ao Programa de Pós-Graduação em Informática da Universidade Federal do Estado do Rio de Janeiro (PPGI UNIRIO) por oferecer, na medida do possível, em meio a crise em que o Brasil atualmente atravessa, boas condições de infraestrutura para um melhor aprendizado. E também a todo o corpo docente, o pessoal da secretária e a coordenação do mesmo.

Ao Observatório Nacional através do Bruno Bazzanela, chefe da Divisão de Tecnologia e Informática, pelo amplo apoio em relação à necessidade de ausência nas atividades laborais devido ao comparecimento nas aulas do PPGI UNIRIO. Além, é claro, de todos os meus amigos da equipe de trabalho pelo suporte dado durante os meus vários períodos de afastamento.

Emito meus agradecimentos, também, para os colegas de mestrado por constituírem um ambiente de benefício mútuo onde um apoiava o outro. Em destaque, o Fellipe Wood, que acabou virando meu amigo pessoal, sempre com palavras de incentivo e compartilhamento de ideias, receios e conquistas.

Aos amigos pessoais e a minha família por entenderem o fato de não estar presente no futebol dominical, festas e outros eventos sociais para realizar os trabalhos acadêmicos exigidos.

A minha primeira orientadora, a profª Morgana Carmen . E ao meu atual orientador, o profº Sidney Lucena pelos seus ensinamentos e olhar crítico para explorar a minha melhor condição de produção.

E por último, porém não menos importante, gostaria de agradecer a Deus por me dar saúde, perseverança e força de vontade para enfrentar e superar todos os desafios pessoais, acadêmicos e profissionais que surgiram durante esses trinta meses desta dura, mas aprazível, jornada de aprendizado que foi o mestrado.

Silva, Eduardo da Costa da. **Detecção de Ataques de Negação de Serviço Utilizando Aprendizado de Máquina**. UNIRIO, 2016. 128 páginas. Dissertação de Mestrado. Departamento de Informática Aplicada, UNIRIO.

RESUMO

Identificar ataques distribuídos de negação de serviço (DDoS - *Distributed Denial of Service*) é considerada uma tarefa complexa, pois os tráfegos gerados por esses tipos de ataques possuem características dinâmicas que dificultam a obtenção de modelos comportamentais estáticos. Tais peculiaridades dos ataques DDoS dão a possibilidade de camuflagem dos fluxos originados por eles, fazendo com que sejam confundidos com fluxos lícitos, livres de ataques DDoS, provenientes de tráfego legítimo de dados. Mesmo assim, o tráfego malicioso pode apresentar padrões comportamentais diferenciados que permitem ser capturados utilizando métodos ou técnicas adequados. Dado esse contexto, a presente dissertação propõe um sistema baseado em aprendizado de máquina para detectar fluxos de ataques DDoS, incorporando em seu aprendizado o estado global, obtido pelo uso do conceito de entropia, das características de rede presentes nos fluxos IPs (*Internet Protocol*) num dado período de tempo. A arquitetura proposta segue a premissa da escalabilidade e foi baseada na arquitetura *lambda*, integrando três soluções conhecidas e de código aberto para uso em cenários do tipo *Big Data*: o Apache SAMOA, Apache Storm e o Apache Hadoop. A utilização do Apache SAMOA foi caracterizada por fornecer os algorítimos e métodos utilizados neste trabalho, tais como: o VHT (*Vertical Hoeffding Tree*), o Bagging e AdaptiveBagging para a criação da tarefa de classificação dos fluxos. O Apache Storm foi o mecanismo responsável pelo processamento dos fluxos. Já o Apache Hadoop teve a incumbência de armazenar quaisquer dados utilizados pelo sistema. A validação do sistema proposto usou *datasets* realísticos contendo tráfego legítimo e ataques DDoS obtidos em ambiente controlado. Em adição, foram utilizados três cenários contento ataques DDoS do *dataset* CTU-13. Os resultados para a acurácia, superior a 94% no pior caso e tendendo a 100% nos casos restantes, a baixa quantidade de falsos positivos e negativos bem como as suas respectivas taxas, demonstram a eficiência do sistema proposto.

Palavras-chave: DDoS, detecção de anomalia, arquitetura lambda, aprendizado de máquina, entropia.

ABSTRACT

The identification of Distributed Denial of Service (DDoS) attacks is considered a complex task given that traffic generated by this type of attack has dynamic characteristics, making it difficult to obtain statistical behavioral models. These peculiarities of DDoS attacks give the possibility of camouflaging their flows, causing them to be misunderstood as legitimate traffic. Still, malicious traffic may have different behavioral patterns, allowing them to be caught using suitable methods or techniques. Given this context, this dissertation proposes a machine learning-based system to detect DDoS attack flows, incorporating in their learning a global network state represented by the entropy of the characteristics of IP (Internet Protocol) flows present in a given period of time. The proposed architecture follows the premise of scalability and was based on the Lambda architecture, integrating three known open source solutions for use in Big Data scenarios: Apache SAMOA, Apache Storm and Apache Hadoop. The Apache SAMOA provided the algorithms and methods used in this study, such as VHT (Vertical Hoeffding Tree), Bagging and Adaptive Bagging, used to create flow classification tasks. Apache Storm was the mechanism responsible for processing flows and Apache Hadoop has the responsibility to store any data used by the system. The validation of the system used realistic datasets containing legitimate traffic and DDoS attacks obtained in a controlled environment. In addition, experiments used three scenarios with DDoS attacks from CTU-13 dataset. Results shown an accuracy of more than 94% in worst case, tending to 100% in all other cases. In addition, the low number of false positives and negatives, as well as their respective rates, demonstrates the effectiveness of the proposed system.

Keywords: DDoS, anomaly detection, lambda architecture, machine learning, entropy.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Nomenclaturas

Acur	Acurácia
DDoS	*Distributed Denial of Service*
F-M	F-*Measure*
FN	Falsos Negativos
FP	Falsos Positivos
GB	*Gigabyte*
HDFS	*Hadoop Distributed File System*
ICMP	*Internet Control Message Protocol*
IP	*Internet Protocol*
PI	*Processing Item*
Prec	Precisão
ROC	*Receiver Operating Characteristics*
SAMOA	*Scalable Advanced Massive Online Analysis*
Tfn	Taxa de Falsos Negativos
Tfp	Taxa de Falsos Positivos
Tvn	Taxa de Verdadeiros Negativos
Tvp	Taxa de Verdadeiros Positivos
UDP	*User Datagram Protocol*
VFDT	*Very Fast Decision Tree*
VHT	*Vertical Hoeffding Tree*
VN	Verdadeiros Negativos
VP	Verdadeiros Positivos

1. Introdução

1.1 Considerações Iniciais e Motivação para a Pesquisa

Durante os últimos anos, o tráfego de dados na Internet tem apresentado um crescimento exponencial. Em 2010, por exemplo, foram gerados 1 bilhão de dados a cada 2 dias e espera-se que até 2020 sejam criados e manipulados mais de 40 ZB [1]. Uma grande quantidade de serviços de Internet, tais como busca, e-mails, mapas e outras aplicações são providas ao público de forma gratuita [2] e a massa de dados transferida vem crescendo gradualmente devido ao incremento da quantidade de usuários de Internet e da largura de banda consumida por aplicativos disponibilizados de forma *on-line* [3]. Paralelamente, devido à vasta proliferação de novos serviços e o incremento da importância dos mesmos no cenário tecnológico atual, as redes de computadores se tornam alvos cada vez mais atraentes para ataques distribuídos de negação de serviço (DDoS - Distributed Denial of Service). Os Ataques DDoS são aqueles que visam esgotar os recursos computacionais do *host* vítima para que ele não consiga responder a solicitações legítimas de conexão [4]. A exploração de vulnerabilidades de alguns protocolos de rede e a aplicação de novas técnicas de ataque, como por exemplo o NTP (*Network Time Protocol*) *Amplification* [5], amplificam o poder de ataque a ponto de atingir taxas de até 400 Gbits/s de dados para a rede da *host* vítima.

As redes de computadores de grandes instituições devem eficientemente comportar uma grande gama de serviços e/ou aplicações para que seus usuários possam acessá-los da forma mais produtível possível. Portanto, garantir que a rede local de computadores esteja operando satisfatoriamente tornou-se uma atividade primordial. No entanto, há dificuldades inerentes a este gerenciamento, principalmente no que tange à análise do tráfego de rede para a construção de padrões comportamentais de normalidade e de anormalidade da rede, de modo a permitir uma correta

identificação de ataques DDoS, caracterizados como sendo uma anomalia no tráfego. Estas dificuldades são impostas pela semelhança estatística que existe entre tráfego malicioso e certos padrões de tráfego normal (livre de ataques), semelhanças estas ocasionadas pelo novo perfil dos serviços e aplicações de rede e também pelo avanço das técnicas empregadas nos ataques DDoS [6]. Assim sendo, a solução destinada à análise do tráfego deverá considerar o atual cenário dinâmico de transmissão de dados cuja velocidade, variedade e volume são itens basilares do mesmo. Em outras palavras, a solução deverá conseguir trabalhar usando técnicas de *Big Data* [1] [7] para gerar respostas em tempo hábil, conforme exigido para uma adequada reação a eventos de segurança.

Nas arquiteturas contemporâneas, com características centralizadas, dos sistemas de identificação de ataques, verificar todos os pacotes de dados que constituem o tráfego de rede é uma atividade custosa do ponto de vista computacional. Ademais, efetuá-la dentro de um período de tempo relativamente curto sem que informações sobre as características de rede sejam perdidas é um grande desafio. A análise em tempo real é de suma importância para a segurança de redes de computadores e sistemas que fazem esta análise de forma centralizada terem dificuldade de lidar com uma grande quantidade de dados [8].

Uma abordagem que fizesse, em tempo real ou em um tempo relativamente baixo, uma inspeção de forma paralela e/ou distribuída dos fluxos de pacotes com mesmas características de rede (endereço IP de origem, endereço IP de destino, porta de origem, porta de destino e protocolo), dado um certo intervalo de tempo, contribuiria para diminuir o custo computacional e o tempo de análise do tráfego. Em adição, esta solução seria capaz de superar várias limitações, inerentes ao cenário de inspeção de fluxos de dados, para a extração de padrões comportamentais. Como principais limitações, podem ser indicadas [9]:

- A impossibilidade de armazenar todos os fluxos, sendo que somente porções dos mesmos podem ser analisados e armazenados;
- A chegada dos dados em alta velocidade, pois o tempo de processamento tem que ser mínimo e, após realizado, descartar os dados;
- Os fluxos de dados podem se alterar com o passar do tempo, logo dados antigos

[1] As técnicas de *Big Data* compreendem a interligação de um grupo de disciplinas para o processamento de grandes volumes de dados, tais como: estatística, mineração de dados, aprendizado de máquina, redes neurais, análise de redes sociais, processamento de sinal, reconhecimento de padrões, métodos de otimização e abordagens para visualização

tornam-se irrelevantes ou, até mesmo, prejudiciais à atual predição.

Nesse sentido, a utilização de ferramentas de *Big Data*, tais como o Apache Hadoop [2], o Apache Storm [3] e o Apache SAMOA (*Scalable Advanced Massive Online Analysis*) [4] seriam apropriadas. Essas plataformas têm a capacidade de tratar uma grande quantidade de dados de maneira paralela e/ou distribuída. O *software* Apache Hadoop computa as entradas de dados em lote, o Apache Storm efetua as suas tarefas em tempo real, a partir de um fluxo de entrada, e o Apache SAMOA faz a mineração de volumosos fluxos de dados. Ambas as ferramentas são livres e podem trabalhar conjuntamente com técnicas sofisticadas como aprendizado de máquina e inteligência artificial, assim como técnicas mais tradicionais de estatística. Além disso, elas ainda podem ser usadas na coleta, processamento e armazenamento dos fluxos de dados.

A proposta desta dissertação é a implementação de um sistema de detecção de ataques integrando ferramentas de *Big Data* para efetuar a detecção de ataques DDoS através da classificação supervisionada de fluxos de dados com algoritmo de aprendizado de máquina baseado em árvores de decisão, incorporando em seu aprendizado o estado global da rede num determinado intervalo de tempo através do cálculo das entropias das características de rede e suas diferenças de valores para os quatro últimos intervalos de tempo. Também foi considerada, facilidade de expansão arquitetural de modo a aumentar a capacidade de processamento e armazenamento de dados sob demanda, desde que haja recursos físicos disponíveis. A avaliação do sistema desenvolvido foi realizada utilizando um *dataset* realístico mesclando dados reais legítimos, ataques DDoS gerados a partir de um ambiente controlado e três cenários (4,10 e 11) com ataques DDoS do *dataset* CTU-13.

Apesar da imensa variedade de tipos de ataques, o presente estudo abordará os ataques DDoS. Além disso, se destinará a fazer a análise do tráfego sem considerar a ação que será tomada após a detecção de fluxos de dados oriundos de ataque.

1.2 Contribuições do Trabalho

Durante toda a fase de elaboração da presente dissertação foram identificadas algumas contribuições relevantes, dentre as quais é possível destacar:

[2] http://hadoop.apache.org/
[3] http://storm.apache.org/
[4] https://samoa.incubator.apache.org/

- Uma nova abordagem de implementação da arquitetura *lambda*, integrando *softwares* livres capazes de oferecer processamento paralelo, confiabilidade e eficiência na análise de fluxos de dados para a identificação de ataques DDoS;
- A incorporação, no processo de classificação individual de um fluxo, do estado global de um conjuntos de fluxos reunidos num determinado intervalo de tempo, que neste trabalho foi de 30 segundos, finalidade atingida com a construção de um módulo denominado de *holístico*, que considerava a relação temporal das distribuições das características de rede (endereços IPs e portas de origens e destinos) ao longo de intervalos de 30 segundos;
- A coleta de um *dataset* de uma rede de computadores real com fluxos IP livres de ataques distribuídos de negação de serviço (DDoS);
- A criação de um *dataset* contendo fluxos IPs provenientes de ataques DDoS emulados em ambiente controlado, mas guiado por características realísticas de ataques DDoS reportados nos últimos anos, para mesclagem com um *dataset* de fluxos lícitos, ou seja, livres de ataques.
- Resultados bastante expressivos para o sistema proposto, tendendo a 100% de acurácia e de precisão, alcançando porcentagem máxima de acertos na maioria dos casos estudados.

1.3 Estrutura da Dissertação

A estruturação desta dissertação se dará da seguinte forma: o Capítulo 2 abordará os conceitos básicos que envolvem os componentes que farão parte da arquitetura do sistema de detecção de ataques. No Capítulo 3, é apresentado o sistema proposto, descrevendo sua arquitetura, seus componentes e o seu funcionamento. O cenário experimental no qual o sistema foi validado será explicado no Capítulo 4. Já na Capítulo 5, os resultados obtidos são mostrados e discutidos. Por último, tem-se o Capítulo 6 com as conclusões, considerações finais e trabalhos futuros.

2. Fundamentação Teórica

Aqui será abordada a teoria necessária para uma melhor compreensão do presente trabalho. Dessa maneira, serão descritos o que vem a ser uma anomalia no tráfego, o processo de descobrimento de conhecimento em bases de dados, o aprendizado de máquina, a classificação supervisionada, árvores de decisão, árvores de Hoeffding, VFDT, VHT, definição de fluxos IP e, finalizando, com o que atualmente vem sendo pesquisado - os trabalhos relacionados.

2.1 Anomalia de Tráfego em Redes de Computadores

Anomalias de tráfego em redes de computadores referem-se à existência de padrões de tráfego detectáveis em uma dada rede e que não seguem uma normalidade comportamental conhecida para a rede em questão [10]. As anomalias podem aparecer no tráfego de dados por uma ampla gama de motivos, como por exemplo, queda de um sistema, erro de configuração de algum equipamento de rede, intrusões cibernéticas, atividade terrorista, atividades de *softwares maliciosos* e etc. Este trabalho destina-se a detecção de anomalias provocadas por ataques cibernéticos do tipo DDoS (*Distributed Denial of Service*).

A definição de anomalia é simples. No entanto, na prática, a identificação de uma anomalia decorrente um ataque DDoS torna-se uma tarefa desafiadora. Há desafios, principalmente, no que tange os seguintes pontos:

- Necessidade de uma definição do padrão normal de comportamento da rede: em muitos ambientes o padrão de comportamento normal da rede varia conforme o tempo, fazendo com que o comportamento normal atual possa não ser relevante no futuro;

- Disponibilidade de bons *datasets* para a validação dos modelos de detecção utilizados.

Diante desse complexo cenário, alguns métodos, com seus pontos fortes e fracos, vêm sendo empregados para a detecção de anomalias provocadas por ataques.

2.1.1 Ataques DDoS

Ataques em Redes de Computadores têm evoluído a cada dia que passa e, devido a isso, muitos pesquisadores têm proposto novas técnicas para evitar, mitigar ou impedir esses ataques.

Um ataque DDoS (*Distributed Denial of Service*) é um ataque de negação de serviço DoS (*Denial of Service*) cujas origens do tráfego de ataque são distribuídas. Funciona de forma análoga a um ataque DoS, ou seja, tenta tornar um determinado serviço ou servidor indisponível para os usuários ao onerar os recursos existentes no *host* vítima [11]. Muito embora as tecnologias tenham avançado consideravelmente, esse tipo de ataque ainda é uma grande ameaça nos dias atuais. Isso se deve ao fato principalmente de ser difícil distinguir entre um ataque do tipo DDoS e uma grande quantidade de acessos por parte de usuários comuns em determinado horário de pico.

Um ataque DDoS consiste normalmente no envio de diversos tipos de pacotes de dados por várias máquinas de origem para uma única máquina destino. Muitas vezes essas máquinas, os *Bots*, que realizam o ataque são controladas remotamente por um outro computador, o *Master*, formando uma *botnet* [12] - grupo interligado de *Bots* que executam, coordenadamente, ações maliciosas definidas pelo *Master*.

Essas conexões e essa gerência remota das *botnets* são efetuadas através da colaboração de sistemas da informação. Para que diversos computadores possam enviar dados simultaneamente para um *host* vítima é necessário que exista uma ferramenta que seja capaz de efetuar tal operação, e nesse contexto se enquadram os sistemas da informação que coordenam os ataques.

Existem diversas variações de ataques DDoS, de acordo com o protocolo utilizado. Muitas vezes esse tipo de ataque utiliza o protocolo TCP na camada de transporte, pois este protocolo possui mecanismos que consomem mais recursos da vítima do que o protocolo UDP. No entanto, pela dificuldade de obtenção de *datasets* públicos atuais e que espelhassem uma operação realística de uma rede de

computadores, foram empregados nesta dissertação os ataques DDoS do tipo UDP e ICMP. Pois, os mesmos eram os tipos de ataques existentes no *dataset* CTU-13 - detalhado na subseção 4.3.2 - base de dados com fluxos IPs rotulados que melhor reunia os quesitos mencionados, segundo a pesquisa efetuada ao estado da arte à época da escrita deste trabalho.

2.1.2 Métodos de Detecção de Anomalias

Segundo revisão literária realizada para o presente trabalho, esta subseção apresentará alguns métodos de detecção de anomalias comumente encontrados na literatura, tais como: estatístico, baseado em classificação, baseado em agrupamento e desvios, *soft computing*, baseado em conhecimento e combinação de classificadores.

2.1.2.1 Estatístico

Normalmente, métodos estatísticos [13] tentam ajustar os dados a um ou mais modelos estatísticos, usualmente construídos para caracterizar o comportamento normal. Então, aplicam um teste de inferência para determinar se uma instância desconhecida pertence ao modelo de comportamento normal. As instâncias que têm baixa probabilidade de terem sido geradas a partir do modelo aprendido são declaradas como anomalias. Para gerar tais modelos, podem ser aplicadas duas técnicas: paramétrica e não paramétrica. Na paramétrica, leva-se em consideração pressupostos acerca da distribuição em questão e estima-se os parâmetros em relação aos dados inseridos como entrada do modelo. Já na não paramétrica, não se assume quaisquer pressupostos sobre a distribuição considerada.

2.1.2.2 Baseado em classificação

A classificação é o problema de identificar quais conjuntos de categorias uma nova observação pertence, fundamentado em dados de treinamento contendo observações cujas as suas respectivas categorias são conhecidas [14]. Portanto, as técnicas de classificação são baseadas no estabelecimento de um modelo implícito ou explícito que é capaz de categorizar os padrões comportamentais da rede de computadores em várias classes. Muitas vezes, a aplicabilidade dos princípios do aprendizado de máquina - como, por exemplo, a classificação - coincide com as técnicas estatísticas, embora a classificação refine o desempenho do seu modelo considerando resultados anteriores, ou seja, considerando experiências anteriores para a construção de um modelo de classificação mais ajustado ao dados do que um modelo produto de técnica puramente estatística.

2.1.2.3 Baseado em agrupamentos e desvios

Agrupamento é a tarefa de separar objetos em grupos [15]. Aqueles objetos que possuam algum tipo de semelhança ficam num determinado grupo. Outros objetos com características similares, mas diferentes das características de um grupo já existente são colocados em um outro grupo. Aquele objeto que não se encaixa em nenhum grupo, por ter características altamente diferentes das do restante e que, provavelmente, não podem ser originados a partir desse modelo de dados, é chamado de desvio [16].

2.1.2.4 Soft computing

Soft Computing [17] é uma abordagem para a computação que se assemelha à notável capacidade do cérebro humano de racionar e aprender em um ambiente de incerteza e imprecisão. Na tentativa de encontrar soluções razoavelmente úteis, os métodos baseados em *soft computing* levam em consideração a presença de incerteza e imprecisão na formulação de seus modelos. Como exemplos de tais métodos, é possível citar a Lógica Fuzzy, a Rede Neural, os Algoritmos Genéticos, o Algoritmo de Colônia de Formigas e os Sistemas Imunológicos Artificiais.

2.1.2.5 Baseado em conhecimento

Nos métodos baseados em conhecimento [18], eventos da rede de computadores ou dos *hosts* são confrontados contra regras ou padrões de ataques pré-definidos. A meta desse tipo de método é representar um ataque previamente conhecido de uma forma generalizada, de modo que tratar uma atual ocorrência do mesmo se torne uma atividade trivial. Os sistemas especialistas, os baseados em regras, os baseados em ontologias, os baseados em lógica e os de análise de transição de estado são exemplos de aplicabilidade desse tipo de método.

2.1.2.6 Combinação de classificadores

A combinação de classificadores é um método cuja estratégia é reunir técnicas de várias naturezas com o intuito de combinar, de alguma forma, o resultado da predição de seus modelos para maximizar o acerto na detecção [19] de uma anomalia no tráfego de dados da rede de computadores provocada por um ataque. O aumento da acurácia na identificação dos ataques é devido ao fato da mistura de técnicas efetuar uma relação mútua de benefício entre as técnicas envolvidas, isto é, a qualidade de uma estratégia supre a deficiência de uma outra técnica. Como

exemplo desse método, e que foram utilizados nesta dissertação, é possível apontar o Bagging e o Adaptive Bagging. O Bagging faz exatamente o que foi descrito, aumenta a acurácia da classificação criando um classificador resultante da combinação dos resultados individuais dos outros métodos. Já o Adaptive Bagging, tem o funcionamento análogo ao Bagging, exceto pelo fato de efetuar o monitoramento da acurácia e refinar o modelo de classificação, com os últimos itens analisados, caso detecte diminuição em seu valor.

2.2 Entropia de Shannon e Desvio Padrão

A entropia empregada neste trabalho foi a de Shannon [20], enquanto métrica estatística. É usada para descrever o grau de dispersão ou concentração das distribuições de probabilidades das características de rede (IPs e portas de origem e destino) [21]. Essa métrica é uma das mais adotadas na literatura para identificação de tráfego malicioso, dando uma visão de estado global das características de rede dentro de um intervalo de tempo. Por isso, foi selecionada. Optou-se pelo uso da forma normalizada da entropia de Shannon, onde os valores calculados podem variar entre 0 e 1. A equação usada foi a seguinte:

$$E = \| \frac{-\sum_{i=1}^{N} p_i log_2(p_i)}{log_2(N)} \| \tag{2.1}$$

As variáveis utilizadas na equação são:

- N - Quantidade de diferentes ocorrências no espaço amostral;
- p_i - A probabilidade associada a cada ocorrência i.

Neste artigo, N é a quantidade de fluxos contidos em 30 segundos e p é a frequência da característica de rede observada.

O desvio padrão é uma métrica de dispersão usada para indicar a variação de uma medida em relação a sua média [22]. Ela foi usada para indicar as variações nos tamanhos dos pacotes de dados e nas durações dos fluxos de dados. É calculada com a seguinte fórmula pelo módulo holístico:

$$S = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \overline{x})^2}{N}} \tag{2.2}$$

As variáveis utilizadas na equação são:

- x_i - O valor do $atributo_i$;
- $\overline{x}$ - A média dos valores dos N atributos;
- N - Quantidade de fluxos;

2.3 Descoberta de Conhecimento em Base de Dados

A Descoberta de Conhecimento em Bases de dados é um processo geral de descoberta de conhecimentos úteis, previamente desconhecidos, a partir de grandes bancos de dados [23].

A Figura 2.1 ilustra o funcionamento desse tipo de processo. A primeira etapa é a seleção dos dados que serão tratados para que seja possível extrair algum tipo de conhecimento. Assim, os dados que serão analisados são agrupados e organizados. Essa etapa é uma das mais importantes do processo de descoberta de conhecimento em bases de dados, visto que os dados são selecionados de acordo com o domínio a ser estudado. A etapa de seleção pode ser bem complexa, principalmente quando os dados são oriundos de diversas bases, como nessa pesquisa.

A etapa seguinte é a de pré-processamento, onde os dados são adequados ao formato necessário, eliminando redundâncias e inconsistências. Essa fase também é conhecida como limpeza dos dados, pois muitas informações não necessárias para a investigação do problema devem ser eliminadas e a base de dados ajustada somente com as informações relevantes para a pesquisa. É importante que essa etapa seja realizada com atenção e que tenha auxílio de um especialista que possa informar se um dado é relevante ou é um *outlier*.

Na etapa seguinte é realizada a transformação dos dados. Essa fase antecede a mineração de dados. Nessa etapa os dados são formatados para que as técnicas selecionadas consigam ser executadas. Também é possível nessa etapa encontrar dados não expostos explicitamente. São os chamados dados derivados, que podem ser transformados a partir de outros dados. Por exemplo, o atributo “idade” pode ser obtido através da informação de data de nascimento.

Todas as etapas são extremamente importantes para que o processo de descoberta de conhecimento em bases de dados funcione adequadamente, entretanto a

literatura fornece maior destaque para a etapa de *Data Mining*. *Data Mining* é a exploração e análise, de forma automática ou semi-automática, de grandes bases de dados com objetivo de descobrir padrões e regras [23]. Em outras palavras, o objetivo da mineração de dados é encontrar padrões em grandes quantidades de dados que possibilitem as organizações a encontrar a melhor forma de lidar com essas informações que encontravam-se ocultas. Nessa etapa, técnicas de aprendizados de maquina [24] são executadas, permitindo que os padrões sejam encontrados.

A última etapa da descoberta de conhecimento em bases de dados é a interpretação da informação obtida. Essa informação gera conhecimento para o usuário que a analisa. Como muitas das técnicas aplicadas nos dados geram informações distintas, é importante que haja um especialista específico para a análise do conhecimento.

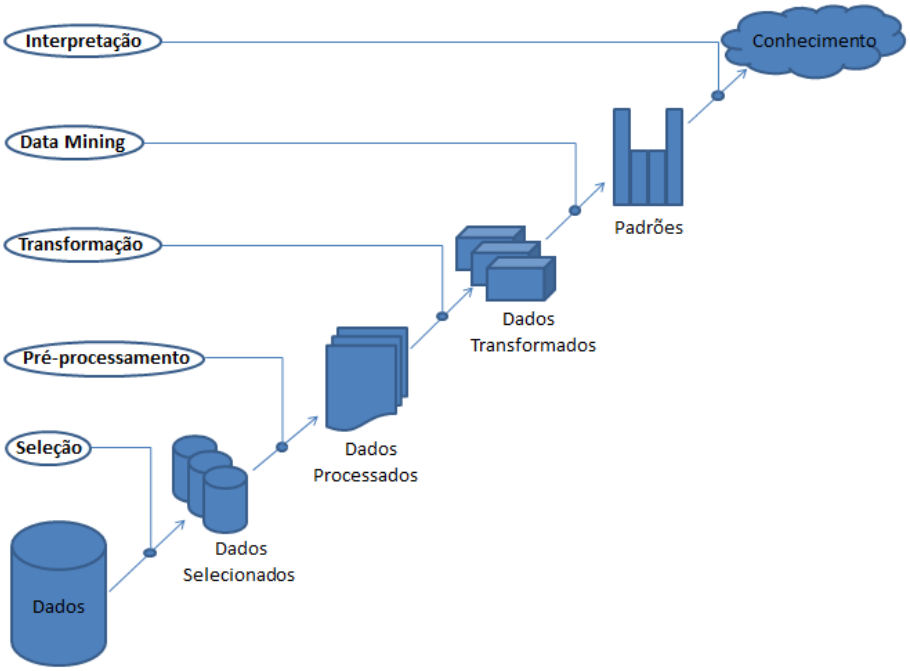

Figura 2.1: Descoberta de conhecimento em bases de dados

2.3.1 Mineração de fluxos de dados

A mineração de fluxos de dados é uma técnica utilizada para extrair informações úteis, ao caso estudado, de infinitos fluxos de dados, representando-as em estruturas de conhecimento conhecidas como padrões e modelos. Ela é utilizada em virtude de deficiências inerentes na forma de funcionamento das técnicas tradicionais (em *batch*) de extração de conhecimento em base de dados. Pois, as técnicas que fazem a análise dos dados em *batch*, efetuam múltiplas re-leituras dos dados de treinamento para capturar as informações necessárias para a confecção de seu modelo de predi-

ção, tornando-as inapropriadas nos ambientes que necessitam manipular fluxos de dados [25], pois os fluxos passam uma única vez e numa velocidade maior do que elas conseguem tratá-los. Além disso, é necessário que os fluxos de dados estejam armazenados permanentemente para que os mesmos sejam consultados durante o processo de re-leitura para refinamento do modelo de classificação - o que é inviável em função da não limitação na quantidade de fluxos de dados.

A mineração de fluxos de dados considerara e superara as limitações de memória, armazenamento e velocidade quanto ao tratamento dos longos fluxos de dados. Utiliza, para isso, estruturas de dados e estratégias de processamento que capturam as informações ocultas dos fluxos na única vez que eles passam [26]. Dessa forma, não existe a necessidade de armazená-los permanentemente e efetuar várias leituras dos mesmos após seu armazenamento. Isso permite analisar mais fluxos de dados utilizando menos recursos computacionais.

2.4 Aprendizado de Máquina

Aprendizado de máquina ou *machine learning* consiste num conjunto de procedimentos computacionais alicerçados na utilização de rotinas lógicas ou binárias que aprendem uma determinada tarefa de uma série de exemplos [27], chamados de dados de treinamento. Após a fase de aprendizado, a tarefa é validada com o uso de outro conjunto de dados, agora denominado dados de teste. A tarefa a ser aprendida neste estudo será a classificação dos fluxos de dados em *normal* ou *ataque* usando classificação supervisionada - onde os dados de treinamentos devem estar etiquetados com as suas respectivas classes - para a construção de um modelo de classificação baseado em árvore de decisão [28]. A abordagem dessa tarefa foi selecionada porque não exige intervenção humana no processo de classificação e fornece a possibilidade do modelo capturar complexos padrões existentes entre os dados analisados.

A avaliação do desempenho desse classificador é feita através da tarefa *prequential or interleaved-test-then-train*. Na *prequential*, cada atributo pode ser usado primeiro para testar e depois para treinar o modelo. Quando a operação é feita desta forma, é garantido que o modelo seja sempre testado com novas instâncias e a acurácia do sistema cresça gradualmente ao longo do tempo, dando mais importância à média global do que a um individual atributo de teste [29].

2.5 Classificação Supervisionada

Na classificação supervisionada, é utilizado um *dataset* onde as suas instâncias são etiquetadas - no caso do presente trabalho, este seria etiquetado como normal ou como ataque. Uma abordagem típica, em tais casos, é a construção de um modelo preditivo para classificação de instâncias normais e instâncias de ataques. Qualquer instância ainda não conhecida é comparada contra o modelo preditivo para determinar a que classe essa instância pertence.

Existem dois principais problemas que afetam a classificação supervisionada: o ajuste exagerado aos dados de treinamento e a falta de instâncias devidamente etiquetadas. O primeiro caso vem do fato das instâncias normais se apresentarem em maior quantidade do que as instâncias de ataque. Isso pode provocar um ajuste exagerado do modelo de predição às instâncias normais, chegando até a capturar os padrões de seus erros e desvios, podendo ocasionar um baixo desempenho na classificação de instâncias desconhecidas - o que naturalmente ocorrerá. O segundo é explicado pela falta de *datasets* realísticos livremente disponibilizados para *download*.

Como principais vantagens, pode-se apontar:

- flexibilidade no treinamento e teste;
- adaptabilidade, pois permitem a atualização de sua estratégia de execução com a incorporação de novas informações;
- apresentam alta taxa de detecção para ataques conhecidos.

Apesar das boas vantagens, ainda apresentam limitações, tais como:

- consumo de muitos recursos computacionais quando comparado com outras técnicas, como as estatísticas;
- altamente dependente dos pressupostos feitos pelos classificadores, e;
- impossibilidade de detecção de ataques para os quais não houve treinamento, ou seja, sem que antes seja alimentado com dados de treinamento para um dado tipo de ataque.

2.6 Árvore de Decisão

Uma árvore de decisão é determinada como um processo de classificação que faz a divisão recursiva de um conjunto de dados em conjuntos menores baseada em regras simples contidas em cada "nó interno" (etapa de divisão), com intuito de associar um dado observado a um determinado valor. E como a quantidade de classes, igual a 2 (normal ou ataque), dos fluxos de treinamento para a construção do modelo de classificação é previamente conhecida, o processo de classificação constitui um método supervisionado de classificação.

A estrutura de uma árvore de decisão é composta por um nó raiz, nó(s) de decisão, nó(s) internos ou nó(s) de divisão, aresta(s) ou ramo(s), e folha(s) ou nó(s) terminal(is). O nó raiz é o objeto mais importante do conjunto, em função de ter a melhor qualidade de informação. O nó de decisão contém as regras que testam um atributo particular em relação a uma constante, a aresta conduz o dado a um possível valor ou a um outro teste - dependendo da resposta ao teste anterior - e a folha possui o valor correspondente à classe do objeto.

O percurso percorrido por um dado da raiz até uma folha configura uma regra de classificação utilizada para classificá-lo. A Figura 2.2 exemplifica a estrutura de uma árvore de decisão, sendo NR o nó raiz, ND o nó de decisão, NF o nó folha, e as interligações entre os nós são representadas pelas arestas.

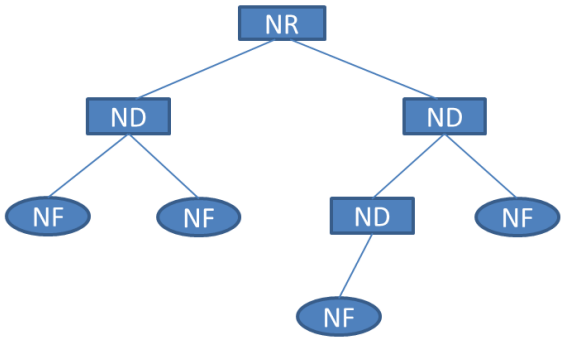

Figura 2.2: Componentes da estrutura de uma árvore de decisão

As árvores de decisão possuem relevantes características, como, por exemplo, o fato delas serem estritamente não paramétricas, não necessitarem de premissas quanto à natureza das distribuições dos dados de entrada e poderem trabalhar com relações não lineares existentes nos vínculos entre os atributos de um *dataset* e as classes dos mesmos. Esse tipo de estrutura de dados permite, também, a manipulação de dados numéricos e/ou categóricos de forma natural. Ademais, a interpretação

do modelo de classificação gerado por uma árvore de decisão se torna fácil por ser intuitivo. Tais peculiaridades caracterizam as árvores de decisão como ferramentas vantajosas em relação a outros métodos de classificação supervisionada [30].

Neste trabalho serão utilizados métodos de classificação baseados no classificador VHT explicado na seção 2.9, que por sua vez, é derivado de uma estrutura de árvore de decisão.

2.6.1 Critérios de divisão ou crescimento

A construção de uma árvore de decisão tem com base o conceito "dividir e conquistar". Isto é, conforme já explicado na subseção anterior, se dá a partir de divisões sucessivas até que se chegue a um valor para um determinado objeto observado, valor este correspondente a um percurso de visitação de nós desde a raiz até uma folha. Para tal, é preciso selecionar quais serão os nós - que nesta dissertação são os atributos dos *datasets* - a partir dos quais a árvore se subdividirá ou, como muitos autores denominam, crescerá. A seleção dos nós é dirigida considerando a qualidade de informação provida pelo nó, onde, normalmente, essa qualidade é calculada através do cálculo de métricas como: ganho de informação, taxa de ganho de informação, índice Gini ou ambos. O nó que apresenta maior qualidade informação será o nó selecionado.

O cálculo do ganho de informação [28] é efetuado com o auxílio do conceito de entropia - que, neste caso, quantifica diferença entre pureza ou organização dos dados em relação a um atributo [31] antes da divisão da árvore com o seu valor após essa divisão, com a meta de diminuir a dificuldade de previsão ocasionada pela aleatoriedade dos valores que definem uma classe. No entanto, essa métrica apresenta a tendência de selecionar nós com grande quantidade de arestas. Uma consequência disso é que pode ocorrer um ajuste exagerado do modelo de classificação ao conjunto de dados de treinamento quando seus atributos têm grande quantidade de valores, chegando até a se ajustar aos erros e desvios dele, ocasionando baixo desempenho no momento da classificação de dados novos, ou seja, dados não pertencentes ao conjunto de dados de treinamento.

Uma técnica básica para mitigar o ajuste exagerado de um modelo aos seus dados de treinamento é a utilização de uma outra métrica denominada de "taxa do ganho de informação" [32]. O cálculo da taxa do ganho de informação, ao contrário do cálculo do ganho de informação, leva em consideração o tamanho e o número de filhos deste nó, capturando assim informações implícitas associadas ao crescimento

da árvore.

Já o índice Gini, também conhecido como coeficiente de Gini [33] é uma métrica de desigualdade usada em muitos campos da ciência para caracterizar distribuições. E, assim como o ganho de informação, também é aplicado levando em consideração a diferença dos valores de um atributo antes e depois do crescimento da árvore de decisão.

2.6.2 Aprendizado

A fase de aprendizado, ou indução, de uma árvore de decisão é etapa na qual um modelo de classificação é construído. A construção do modelo está diretamente associado à escolha dos melhores atributos, pois os mesmos levarão à folha na qual existe a resposta da pergunta que se deseja responder ao efetuar uma classificação: a que classe pertence esse objeto ou instância? Essas folhas são os nós com maior significância em termos de informações a respeito de uma classe.

A criação de um modelo de classificação exige um conjunto de dados previamente conhecidos. Cada objeto ou instância do conjunto de dados é marcada com a sua respectiva classe. Dele, então, são extraídos os padrões ocultos ao olho humano, que vão fazer parte do modelo de classificação usado para predizer a classe de uma instância nova com base nos padrões capturados e aprendidos dos dados de treinamento. Esse confronto acontece individualmente a cada passagem de instância nova que adentra ao modelo de classificação. O grupo de instâncias novas é chamado de dados de teste, pois são dados não utilizados na fase de treinamento e, portanto, desconhecidos pelo modelo de classificação aprendido.

O processo de aprendizado se inicia com a seleção do melhor atributo ou nó do conjunto de dados - o nó *root*. Depois, um ou mais nós internos filhos ou folhas são criados para cada valor que o tributo pode assumir. Esse comportamento é repetido até que alguma condição de parada seja alcançada [34].

2.7 Árvore de Hoeffding

Uma árvore de Hoefding [35] é a denominação dada ao modelo de classificação originado de uma árvore de decisão que utiliza a métrica estatística conhecida como limiar de Hoeffding [36] - que é independente da distribuição da variável considerada - para selecionar a quantidade mínima de observações necessárias de instâncias de

treinamento de modo a escolher o melhor atributo no qual a árvore se subdividirá. O limiar de Hoeffding afirma que com probabilidade igual a 1 - δ, a média real de uma variável aleatória num intervalo R não será diferente da média estimada calculada a partir de n independentes observações se essa média estimada calculada for menor do que ϵ - calculado pela equação 2.3. Essa particularidade das árvores de Hoeffding garante que a utilização de n observações, que é a menor quantidade possível de observações dentro de um intervalo de valores R, para escolher o melhor atributo seria o mesmo que se utilizasse infinitas observações para a indicação do mesmo atributo.

$$\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}} \tag{2.3}$$

As variáveis utilizadas na equação são:

- R - faixa de valores de uma variável aleatória;
- δ - é derivado da diferença entre 1 e a probabilidade de acerto na escolha de um nó para o crescimento da árvore de decisão;
- n - quantidade de observações independentes;

Seja $G(B_i)$ o critério de divisão da árvore de decisão, B_a o atributo com o maior valor para o critério de divisão da árvore de decisão e B_b o segundo atributo com o maior valor para o critério de divisão da árvore de decisão e ϵ, calculado com δ definido pelo usuário. E seja $\Delta\overline{G} = \overline{G}(B_a) - \overline{G}(B_b) \geq 0$ uma nova variável aleatória com a diferença entre os valores do critério de divisão da árvore de decisão em n independentes observações. Após aplicar o limiar de Hoeffding em $\Delta\overline{G}$ e verificar que é maior que ϵ, o atributo B_a é selecionado como o melhor atributo para que partir dele se faça a divisão da árvore de decisão, pois é possível atestar que $\Delta G \geq \Delta\overline{G} - \epsilon > 0$ em função de $\overline{G}$ ser uma média dos valores de G. Como resultado, o nó precisa acumular atributos até que o ϵ seja menor do que $\Delta\overline{G}$. Então, neste ponto, a árvore estará pronta para se dividir utilizando o melhor atributo, encaminhando os itens observados seguintes para novas folhas.

2.8 VFDT

O Very Fast Decision Tree (VFDT) [35] é um dos algoritmos inspirados nas árvores de Hoeffding. Tal algoritmo de Domingos e Hulten é projetado com melhorias que dão a possibilidade de capturar rapidamente os padrões ocultos de *datasets* com grandes volumes de elementos, possivelmente infinitos. O algoritmo permite a utilização de duas métricas para a avaliação de atributos: o ganho de informação e o índice Gini. Além disso, o mesmo tem uma natureza incremental em seu aprendizado, ou seja, à medida que novos dados alimentam o algoritmo, o mesmo vai aprimorando o seu modelo de classificação através da seleção do melhor atributo de crescimento da árvore de decisão, utilizando o mecanismo da árvore de Hoeffding nessa tarefa.

O VFDT inclui uma série de elementos de melhoria que aumentam o seu desempenho quando comparado a outros tipos de árvores de decisão. No âmbito da igualdade na métrica de escolha de um melhor atributo entre dois atributos, pode selecionar, de acordo com o padrão configurado, qualquer um deles. Como os valores da métrica ganho de informação são similares, não fará diferença optar por um em detrimento de outro. No cálculo do ganho de informação, este se dá com o acúmulo de uma quantidade mínima de exemplos para efetuar a operação, pois seria muito custoso do ponto de vista temporal o cálculo individual dessa métrica para cada instância. A economia de memória também foi contemplada com a eliminação de atributos que pouco contribuirão para a formação do modelo de classificação. Isso acontece quando a diferença entre o ganho de informação entre atributos é maior que o limiar de Hoeffding. Então, esses atributos são removidos e o montante de memória por eles consumida é disponibilizada ao uso.

2.9 VHT

O Vertical Hoeffding Tree (VHT) é um classificador distribuído e escalável que utiliza paralelismo vertical sobre um classificador baseado em uma árvore de Hoeffding ou Very Fast Decision Tree (VFDT).

Tanto a árvore de Hoeffding quanto a sua variante melhorada, a VFDT, não utilizam paralelismo em suas abordagens. O paralelismo vertical implementado no VHT permite que as instâncias que compõem um *dataset* sejam divididas, em termos de atributos, em grupos para que sejam processados paralelamente. Então, esses N

grupos são disponibilizados ao mesmo tempo às N unidades verticais de processamento para cálculos em paralelo das métricas de critério de divisão da árvore de decisão, como, por exemplo, o ganho de informação. De posse dessas métricas calculadas, chamadas de estatísticas locais, elas são combinadas para o cálculo de uma estatística global que decidirá em que nó ou em que atributo a árvore de decisão será dividida e, consequentemente, crescerá. O emprego dessa arquitetura paralela facilita a escalabilidade, sendo muito eficiente para o processamento paralelo de *datasets* com muitas dimensões [37]. A Figura 2.3 exibe o diagrama da implementação do paralelismo vertical do VHT no Apache SAMOA.

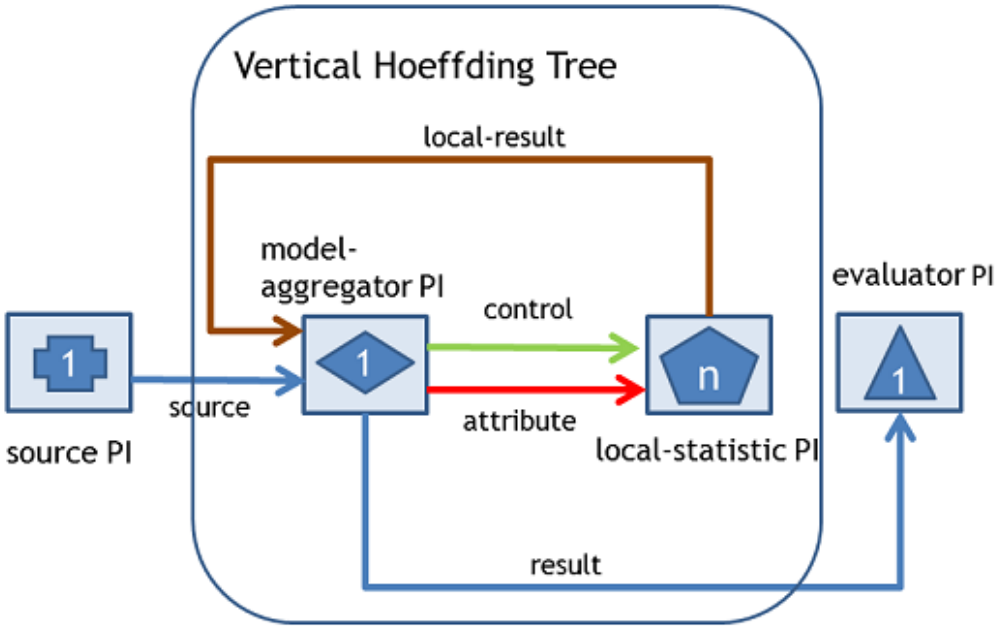

Figura 2.3: Diagrama do classificador VHT [38]

Os componentes referenciados na Figura 2.3 são:

- source PI e evaluator PI: são componentes da tarefa de avalização *prequantial*;
- local-statistic PI: unidades de processamento verticais que computam as métricas referentes aos atributos recebidos;
- model-aggregator PI: unidade que divide os atributos em N grupos e envia para as unidades de processamento verticais. Além disso, é o responsável por sumarizar as métricas locais enviadas pelos *local-statistic PIs* e construir o modelo de classificação;

- control stream: utilizado pelo *model-aggregator* para enviar mensagens de controle a um *local-statistic PI*;
- attribute stream: utilizado pelo *model-aggregator* para enviar atributos a um *local-statistic PI*;
- local-result stream: utilizado por um *local-statistic PI* para enviar o resultado do seu cálculo;
- evaluator PI: efetua a avaliação da classificação, exibindo a acurácia do modelo de classificação;
- result stream: utilizado pelo *model-aggregator* para enviar o resultado da classificação para o *evaluator PI*;

2.10 Fluxos IP

De acordo com [39], o fluxo IP é definido como uma sequência unidirecional de pacotes que compartilham algumas propriedades, tais como endereço IP de origem, endereço IP de destino, porta de origem, porta de destino e o protocolo. Um fluxo é considerado encerrado ou inativo quando: (i) não existe a passagem, pelo equipamento responsável pela medição dos fluxos, de pacotes pertencentes a este fluxo durante um determinado intervalo de tempo; (ii) o fluxo de dados atinge o seu tempo máximo de vida ou atividade; (iii) há o recebimento, no caso de fluxos TCP, de pacotes de dados indicando que o mesmo foi finalizado normalmente com a *flag* FIN ativa ou foi reiniciado com a *flag* RST ativa; ou ainda, (iv) por limitações internas, levando o fluxo a ser terminado prematuramente como, por exemplo, devido a baixa quantidade de memória. Tanto o tempo máximo que deve ser aguardado entre as chegadas de pacotes de um mesmo fluxo quanto o tempo máximo de vida ou de atividade de um fluxo possuem valores padrões que variam conforme o fabricante do equipamento. Ademais, oferecem a possibilidade de serem configurados ao bel-prazer do administrador de rede.

2.10.1 Análise do tráfego utilizando fluxos IP

A análise de tráfego no nível dos fluxos IP oferece a condição de capturar, de forma eficiente, diversos tipos de comportamentos anormais da rede de computadores provocados, por exemplo, por ataques DDoS, erros na configuração de equipamentos, falhas e etc. Um outro benefício da utilização de fluxos IPs é que estes geram

muito menos dados a serem analisados do que numa abordagem realizando a mesma inspeção a nível de pacotes [40], o que, na maioria dos ambientes, pode significar um problema em vista da quantidade de tráfego transmitido.

2.10.2 NetFlow

Segundo [41], o NetFlow é um recurso disponibilizado inicialmente nos roteadores e *switches* da empresa CISCO para coleta do tráfego de rede, atuando como um caracterizador de tráfego e oferecendo informações sumarizadas em nível de fluxos IP aos administradores da rede acerca dos pacotes transmitidos. Esses fluxos IP sumarizados reportam dados importantes para o conhecimento da operação da rede de computadores. Essas informações são disponibilizadas em forma de registros NetFlow que, por sua vez, são formados por 7 atributos principais: endereço IP de origem, endereço IP de destino, porta de origem, porta de destino, protocolo, tipo de serviço (ToS - Type of Service) e interface lógica de entrada. Se um pacote apresenta diferença em qualquer um desses atributos principais, ele pertencerá a outro fluxo, portanto, será exibido em outro registro. O processo de criação de um fluxo IP é exemplificado na Figura 2.4.

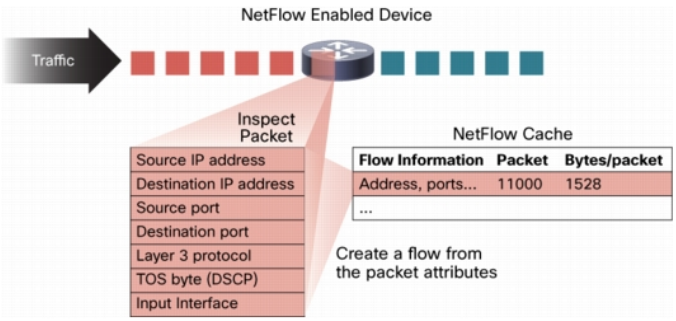

Figura 2.4: Criação de fluxo IP NetFlow [39]

A exportação dos dados é efetuada utilizando o protocolo UDP. Atualmente existem 5 formatos para o NetFlow: Versão 1, Versão 5, Versão 7, Versão 8 e Versão 9. Todos estes formatos podem ser usados para realizar a exportação dos fluxos, porém os formatos mais usados são os seguintes:

- Versão 5: versão que adicionou as informações BGP dos sistemas autônomos e os números de sequência dos fluxos;
- Versão 9: versão que trabalha com *templates*, que é o seu diferencial. Ela possui um *template* básico que pode ser modificado para personalizar o registro

NetFlow exportado. Foi o formato que inspirou os engenheiros do IPFIX (*Internet Protocol Information Export*).

Como ilustrado na Figura 2.5, a operação básica da exportação dos fluxos IP acontece com a interação entre os componentes "Exportador NetFlow" e "Coletor NetFlow".

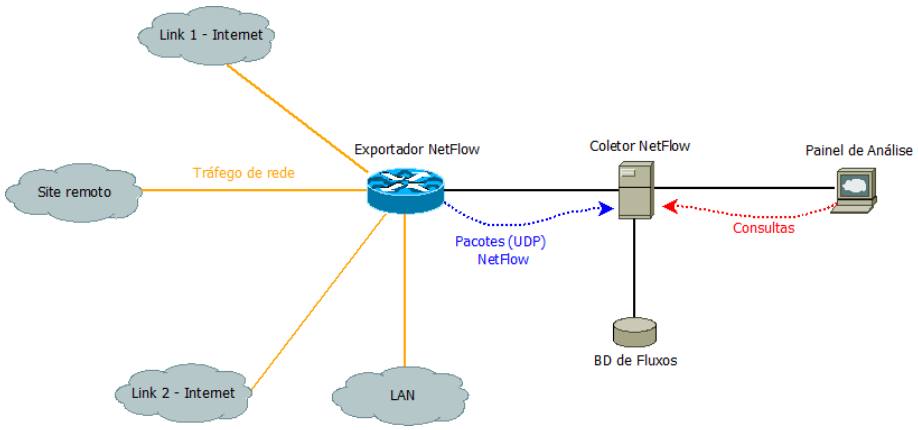

Figura 2.5: Exemplo de uma arquitetura NetFlow

Existe ainda o NetFlow *cache*, que é o encarregado de criar registros, guiado por um determinado formato, referentes à cada fluxo IP ativo. Já o Exportador NetFlow tem a função de exportar os fluxos armazenados no NetFlow *cache* para um Coletor NetFlow de fluxos, que disponibiliza uma visualização de relatórios com os fluxos. A Figura 2.6 exibe o exemplo do conteúdo do NetFlow *cache*.

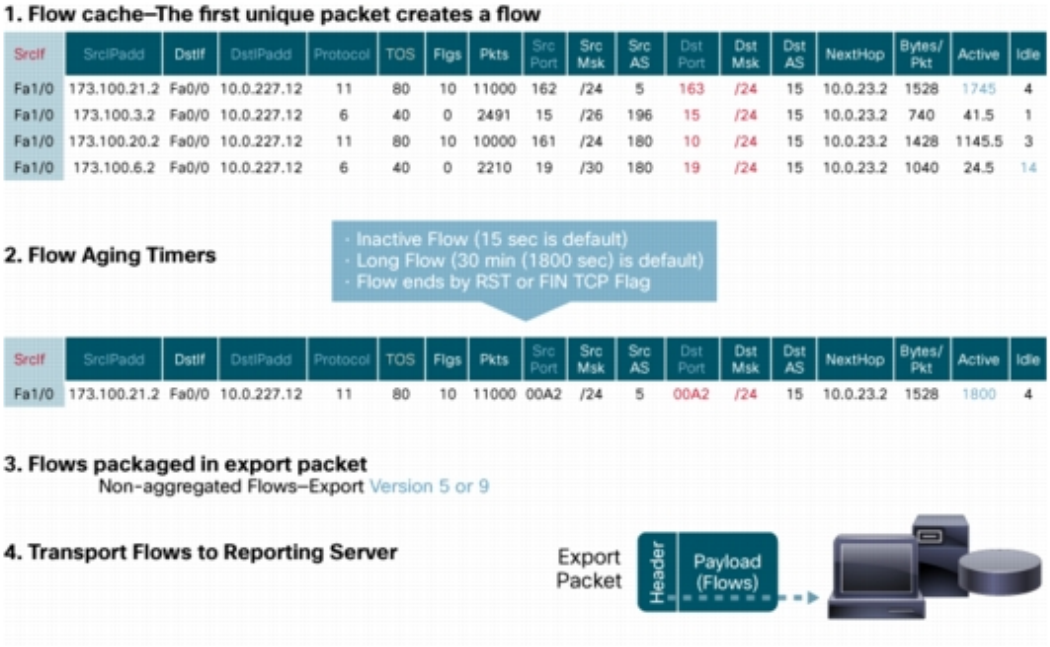

Figura 2.6: Exemplo de conteúdo do NetFlow *Cache* [39]

2.11 Trabalhos Relacionados

Em [18] é realizado um estudo estruturado e comparativo entre as várias técnicas para detecção de anomalias, como, por exemplo, aprendizado de máquina, mineração de dados e baseadas em estatística. Também foram analisados métodos utilizados nos sistemas de detecção de intrusões, fazendo a categorização dos mesmos em função dos mecanismos utilizados, apresentando os seus pontos positivos e negativos. Além disso, há a indicação de pontos que precisam ser mais explorados para melhorar o desempenho desses e de futuros sistemas detecção de anomalias. Apesar de dissertar sobre os diversos métodos e desafios em detecção de anomalias, como se tratava de um estudo, não houve nenhuma implementação do que foi exposto.

Os pesquisadores em [42] se utilizam de uma abordagem empregando aprendizado de máquina para a detecção de anomalias nos fluxos de dados, anomalias estas provocadas por atividades maliciosas de uma *Botnets* num sistema classificado como sistema “quase tempo real”, isto é, o sistema de detecção construído apresentava um tempo de execução próximo a um tempo de execução em tempo real para a detectar anomalias. No caso, tratava-se da detecção de *botnets* em redes P2P. Abordagem usada pode ser escalável e lidar com ampla quantidade de dados, sendo o Hadoop,

o Apache Hive [1] e o Apache Mahout [2] os principais componentes implantados na arquitetura projetada. No entanto, seus mecanismos de extração das características de rede e de classificação supervisionada não foram projetados para trabalhar com processamento de fluxos de dados que evoluem, mas, sim, para processamento em lote onde os dados são estacionários. Como resultado, não há menção à natureza evolutiva das características de rede ao longo do tempo, não considerando a atualização do modelo de classificação gerado. Ademais, o tempo levado pelas execuções das funções de mapeamento e redução que integram o paradigma de programação MapReduce do Apache Hadoop não são indicadas para processamento de fluxos de dados que se modificam com o decorrer do tempo, porque demoram para gerar o resultado.

Para identificar em tempo real as anomalias geradas por ataques que estão acontecendo no momento da análise do tráfego, e a origem das mesmas, os autores em [43] projetaram um sistema distribuído de computação de fluxos com o Apache Storm que monitora ininterruptamente o tráfego de dados na rede de computadores. Segundo os resultados, o sistema proposto provou ter bom desempenho e escalabilidade. Porém, uma desvantagem desse artigo, é que os autores consideram apenas o aumento volumétrico do tráfego de pacotes ou octetos para identificação de fluxos IPs que causam essa anomalia. Ademais, o artigo não deixa claro se o sistema de detecção projetado consegue detectar anomalias que são geradas a partir baixa variação volumétrica no tráfego da rede. E, também, não classificam a natureza do comportamento anômalo da rede, ou seja, se a anomalia é gerada por tráfego lícito ou malicioso.

Analisando os fluxos de dados da rede, um sistema de detecção de ameaças em tempo real foi proposto em [44]. Os ataques DoS e de varredura são detectados com a utilização dos algoritmos C4.5, rede neural e máquina de suporte de vetores via classificação supervisionada dos fluxos. Além disso, para a detecção de novos comportamentos anômalos na rede - aqueles que ainda não possuem dados de treinamento para serem classificados, chamados de *zero-day attacks* - foi usada uma distribuição Gaussiana para a escolha empírica de ”limiares”para os valores assumidos pelos atributos, de maneira a sinalizar um novo comportamento anômalo para um dado atributo, necessitando intervenção humana para a arbitragem destes limiares. Ambas as técnicas são utilizadas depois da seleção dos atributos mais relevantes do *dataset*, obtidos pelo algoritmo de Análise dos Componentes Principais (Principal

[1] https://hive.apache.org/
[2] http://mahout.apache.org/

Component Analysis - PCA). O *dataset* usado foi construído pelos próprios autores a partir de coletas numa rede real com parte dos sistemas propositalmente infectados. Apesar do sistema proposto alcançar mais de 95% de acurácia, a classificação dos fluxos em normais, ataques DoS e de varredura não leva em consideração a forma com a qual esses fluxos estão distribuídos dentro de um intervalo de tempo para ambas as classes. Em adição, a detecção de novas anomalias exige a configuração manual do valor da variável "limiar", de acordo com o tráfego de dados da rede e a anomalia que se deseja detectar.

Os autores em [45] tratam da classificação de fluxos de tráfego malicioso através da identificação das melhores características para classificar um fluxo como sendo originado por um ataque. Para tal, eles usaram o algoritmo de seleção de características SBS (*Sequential Backward Selection*) e o algoritmo de classificação não supervisionada K-means. A avaliação foi efetuada com a utilização de três cenários onde fluxos de ataques DDoS foram misturados aos outros tipos de fluxos nas proporções de 10%, 30% e 50%. Os resultados obtidos variaram de 97,2% a 99,04%. No entanto, não foi encontrado nesse trabalho menção a capacidade de processamento e armazenamento da solução que desse condições de incremento, configurando possíveis problemas de sobrecarga de processamento e armazenamento a medida que aumentar a quantidade de fluxos transmitidos e demandar maiores esforços computacionais para tratá-los.

Os trabalhos apresentados nessa subseção transcorreram desde os estudos de técnicas, métodos e sistemas de detecção de anomalias passando pela análise de tráfego, finalizando com implementações de sistemas de detecção. Tal visão dos trabalhos reflete o cenário do estado da arte, na época da escrita desta dissertação, em termos de detecção de anomalias com um maior foco nas anomalias provocadas por atividades maliciosas como no caso de ataques DDoS. Como pode ser observado, os trabalhos, de uma forma geral, falham em um ou outro aspecto, como por exemplo: tempo de resposta inadequado para geração de alarmes em função de latência em funções que analisam o tráfego; estratégias, como a verificação volumétrica da quantidade de pacotes e octetos, que podem ser facilmente burladas; quando utilizado aprendizado de máquina, falta a consideração do estado da distribuição dos fluxos IPs em um determinado intervalo de tempo pelo fato do modelo de classificação examinar apenas o fluxo corrente e, também, questões relacionadas à capacidade, fácil e prática, de escalabilidade do sistema em termos computacionais para suportar a análise de uma maior demanda de fluxos IPs. Dado esse contexto do estado da arte, a presente dissertação propõe um sistema de detecção que visa superar tais dificuldades cujo

os detalhes serão destacados no capítulo 3.

3. Sistema Proposto

Este capítulo apresentará com maiores detalhes o sistema proposto por esta dissertação, dando uma visão geral da arquitetura projetada e o funcionamento de seus componentes.

3.1 Apresentação do Sistema Proposto

O processamento em lotes (*batch*) é uma técnica normalmente utilizada para a manipulação de grandes volumes de dados, porém a geração de suas respostas apresenta latências. Essas latências, num cenário onde os dados trafegam com velocidade elevada, como no caso de fluxos de dados em redes de alta capacidade, tornam inapropriada a aplicação dessa técnica. Nesse sentido, uma técnica chamada de processamento de fluxos de dados (*streaming processing*) foi desenvolvida, permitindo o processamento de fluxos de dados dentro de um intervalo de tempo próximo ao tempo real.

Dado o cenário de técnicas supracitadas, em [7] foi construída a arquitetura *lambda*, que faz a junção dessas duas formas de processamento para que volumosos fluxos de dados sejam analisados. A arquitetura lambda, ilustrada na Figura 3.1, é constituída pela composição de três camadas: a camada de processamento em lote, a camada de processamento em velocidade e a camada de serviços. A camada de processamento em lote é a responsável por armazenar grande quantidade de dados e processá-los em lote. A camada de processamento em velocidade é designada para a manipulação dos dados que transcorrem em alta velocidade. Já a camada de serviço tem a função de juntar os resultados dos processamentos das outras duas camadas para disponibilizar visualizações dos dados analisados.

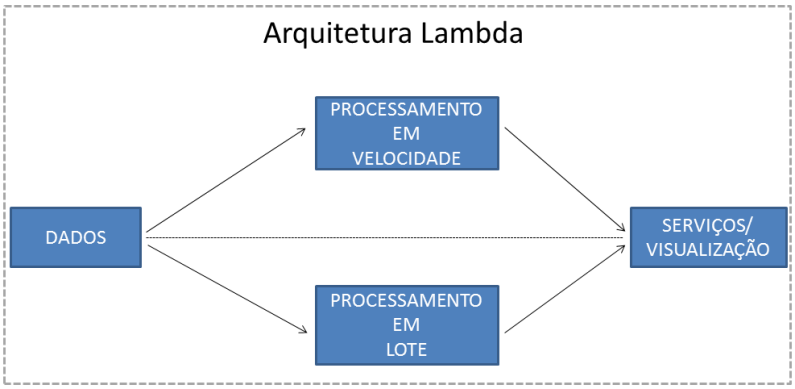

Figura 3.1: Arquitetura Lambda

A proposta de solução deste trabalho é o desenvolvimento de um sistema elástico, em termos de processamento e armazenamento, de detecção de ataques DDoS usando aprendizado de máquina via classificação supervisionada para a construção de modelos comportamentais dos fluxos em condições normais de operação da rede e sob ataque. Esses modelos serão construídos com a utilização de árvores de decisão de Hoeffding em sua versão paralela, a VHT.

A arquitetura do sistema proposto foi baseada na arquitetura lambda para proporcionara capacidade de manipulação de grande quantidade fluxos IPs nos aspectos de armazenamento, processamento e visualização de informações pertinentes ao processo de obtenção dos padrões comportamentais do tráfego. Como mostra a Figura 3.2, o sistema integrará as ferramentas de *Big Data* SAMOA, Storm e Hadoop. Apesar de ter elencado nominalmente tais ferramentas, o interessante delas é suporte tecnológico que elas fornecem ao sistema proposto. O SAMOA tem como principal incumbência a disponibilização de algoritmos de aprendizado de máquina, classificadores ou métodos (VHT, Bagging e AdaptiveBagging) para o sistema. O STORM é o motor de processamento da solução. Já o HADOOP, módulo HDFS, é o local de armazenamento dos *datasets*. Além disso, haverá um módulo holístico cuja principal tarefa será incorporar o processo de aprendizado de máquina uma visão do estado global das características de rede, referente aos fluxos ativos ao longo do tempo no *dataset*. Istó é, fornecerá o quão dispersas ou concentradas os endereço IPs e portas de origem e destino estarão ao longo do tempo.

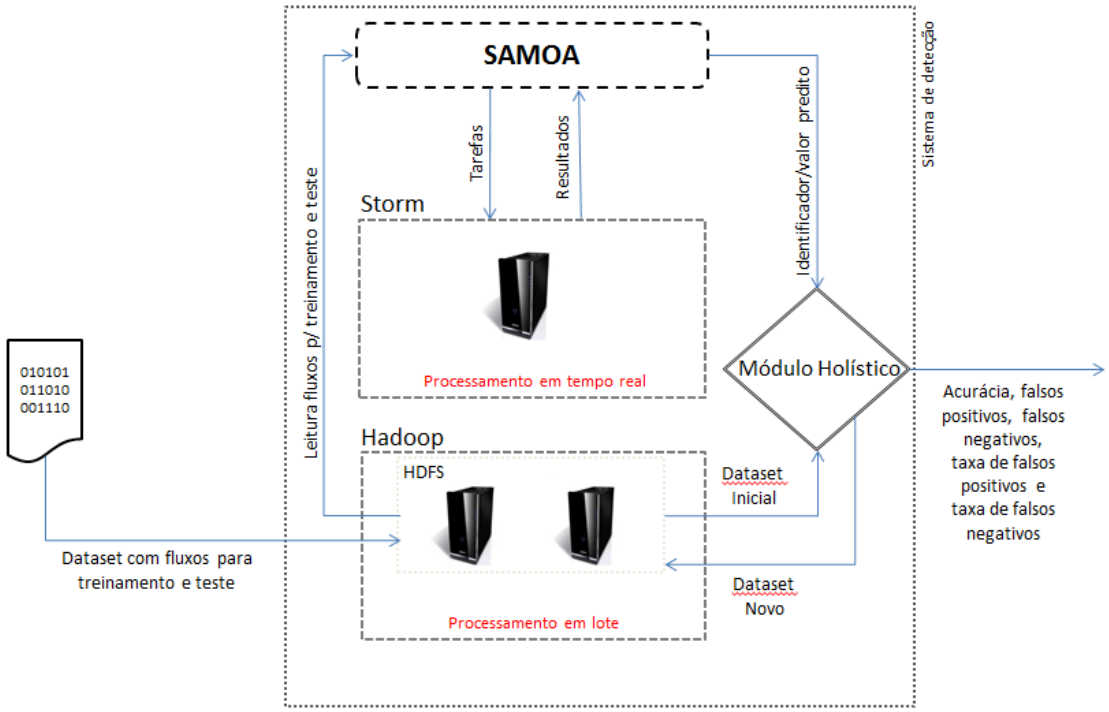

Figura 3.2: Arquitetura do sistema proposto

3.2 Componentes do Sistema Proposto

Nesta seção os elementos integrantes da arquitetura do sistema proposto serão explicados, bem como as suas interligações para que o sistema fosse posto em operação num cenário experimental.

3.2.1 Apache Samoa

O SAMOA (*Scalable Advanced Massive On-line Analysis*) [46] é uma plataforma de *software* livre projetada para minerar grandes volumes de fluxos de dados, isto é, para extrair padrões de informações de fluxos de *Big Data*. O SAMOA pode oferecer flexibilidade para o desenvolvimento de novos algoritmos, ou aproveitamento de algoritmos de outras soluções de aprendizado de máquina, facilidade de integração com outro mecanismos de processamento de fluxos de dados e escalabilidade em caso de crescimento dos dados tratados.

A utilização de interfaces para integração com outros mecanismos de processamento de fluxos, chamadas de DSPEs (*Distributed Stream Processing Engines*),

como por exemplo o Apache Storm, utilizado neste trabalho, permite que o SAMOA seja executado no "topo"de tais mecanismos, sendo considerada uma arquitetura plugável, de fácil integração com esses e novos DSPEs a serem desenvolvidos.

A alimentação do SAMOA se dá pela leitura de arquivos no formato ARFF (*Attribute-Relation File Format*), podendo estes serem lidos diretamente do sistema de arquivos comum ou do sistema de arquivos distribuído do Apache Hadoop, o HDFS. Neste trabalho é usada a segunda opção em virtude da possibilidade de expansão do sistema e pela alta taxa de transferência do mesmo.

Quanto à arquitetura, esta é constituída de 4 camadas: algoritmo, API, ML-adapter e SPE-adapter, como pode ser observado na Figura 3.3. Seguem as descrições de cada camada:

- Camada algoritmo: fornece os algoritmos já implementados no SAMOA;
- Camada API (*Application Programming Interface*): responsável por facilitar a implementação de novos algoritmos;
- Camada ML-adapter: tem como função permitir a integração de novos *frameworks* com o SAMOA, e;
- Camada SPE-adapter: suporta a integração com mecanismos de processamento distribuídos de fluxos ou DSPEs, tal como o Apache Storm.

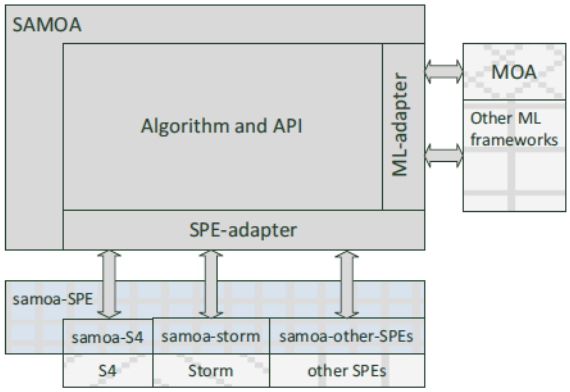

Figura 3.3: Arquitetura do SAMOA [47]

Assim como no Apache Storm, o Apache SAMOA considera um algoritmo como uma topologia - ilustrados na Figura 3.4, isto é, um grafo direcionado de nós que se comunicam entre eles via troca de mensagens.

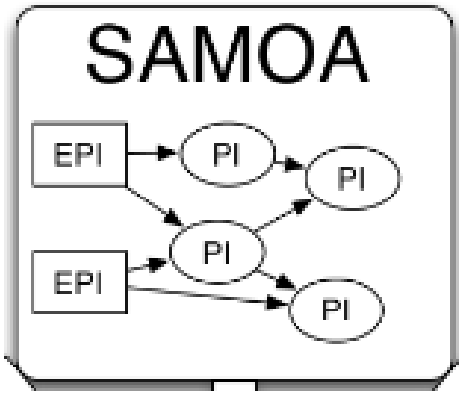

Figura 3.4: Exemplo de topologia no SAMOA [48]

Como pode ser visualizado na Figura 3.5, uma típica topologia no Apache SAMOA contém os seguintes componentes básicos:

- *Processing Item*: é um componente oculto que envolve um *Processor* com a propriedade de conectar *Processors*, porém não tem código. Pode ser de dois tipos:
  - *Simple Processing Item (PI)*: é um componente executável dentro da topologia, e;
  - Entrance Processing Item (EntrancePI): é um PI que aceita itens de entrada que podem gerar o seu próprio fluxo de dados. A sua utilização é baseada na interligação com fontes externas de fluxos de dados para fazer a alimentação da topologia.
- *Processor*: é uma interface que contém toda a lógica escrita para a execução de uma tarefa;
- *Stream*: é um canal de comunicação entre *Processors*, podendo ser do tipo 1 para N, ou seja, uma origem para vários destinos;

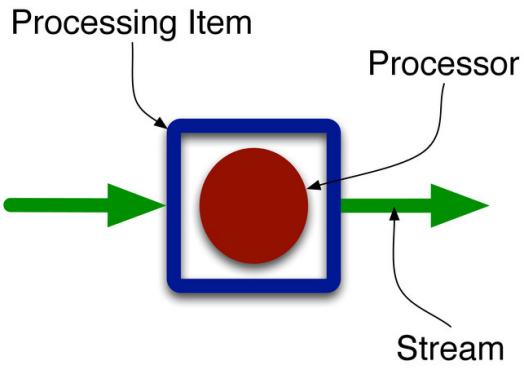

Figura 3.5: Componentes básicos de uma topologia no SAMOA [49]

Além de ser uma plataforma, o SAMOA também é considerado uma biblioteca por oferecer a possibilidade de utilização de diversos algoritmos de aprendizado de máquina para as tarefas de regressão, clusterização e classificação. Exemplos desses algoritmos são o HAMR, que é uma implementação distribuída de *Adaptive Model Rules* [50] para regressão, os algoritmos VHT (*Vertical Hoeffding Tree*) [35] para classificação e uma implementação do algoritmo CluStream [51] para a tarefa de clusterização.

3.2.2 Apache Storm

O Apache Storm é um sistema baseado em *software livre* para computação distribuída em tempo real. Segundo [52], o Storm foi inicialmente criado por Nathan Marz quando trabalhava na empresa BackType, que foi englobada pela empresa Twitter em 2011. Atualmente, o Apache Storm é mantido e suportado pela Apache Software Foundation.

O projeto foi concebido de modo a fazer o processamento distribuído em tempo real de ilimitados fluxos de dados de maneira rápida, escalável, tolerante à falhas, confiável e fácil de operar e administrar ao longo de um grupo de computadores (*cluster*) que executam partes diferentes da tarefa em execução. O intuito é garantir o processamento mesmo que alguns desses computadores falhem ou seus recursos se tornem inadequados para um determinado montante de dados. No entanto, para isso deseja-se que a solução não seja de difícil configuração quando se deseja aumentar o seu poder de processamento. Em função dessas características, podem ser empregadas quaisquer linguagens de programação para trabalhar em conjunto com ele, além de tecnologias atuais de banco de dados para armazenamento de dados resultantes de seu processamento.

3.2.3 Apache Hadoop

O Apache Hadoop é uma plataforma implementada em *software* livre destinada ao armazenamento e processamento em lote de grandes volumes de dados distribuídos ao longo de um conjunto de computadores de baixo custo que formam um *cluster*. A transmissão dos dados entres os nós do *cluster* é realizada em alta velocidade, usando um modelo simples de programação chamado *MapReduce* para a execução paralela das tarefas programadas [53]. Esse *software* vem sendo adotado, rapidamente, pelo meio acadêmico e industrial devido à sua potencialidade e facilidade de uso, sendo cada vez mais desenvolvido e utilizado [54]. Foi projetado para ser tolerante a falhas, escalável e não necessitar de componentes especiais de *hardware*. É fundamentado em 4 módulos:

- Hadoop Common: utilitários comuns que suportam os outros módulos do Hadoop;
- Hadoop Distributed File System (HDFS): sistema de arquivos distribuídos que permite alto *throughput* de acesso aos dados;
- Hadoop YARN: responsável pelo agendamento de tarefas e o gerenciamento do cluster, e;
- Hadoop MapReduce: um sistema baseado no YARN para processamento paralelo de grande conjunto de dados.

Neste trabalho, em função da alta latência entre as funções de mapeamento e redução [55], é somente utilizado o HDFS, que serve como um repositório dos dados utilizados pelo sistema de detecção de ataques desenvolvido. Repositório no qual o SAMOA lerá os fluxos para fins de classificação e, também, estrutura na qual o módulo holístico (a ser descrito a seguir) lerá esses mesmos fluxos para efetuar o cálculo de métricas que visam caracterizar o estado global da rede num dado intervalo de tempo.

3.2.4 Módulo Holístico

O módulo holístico é um importante componente do sistema proposto para detecção de ataques DDoS. Esse módulo tem a responsabilidade de ler todos os fluxos do *dataset* armazenado no HDFS e dele extrair características temporais relacionadas a estes fluxos, criando assim um novo *dataset* enriquecido com tais informações.

As características temporais são diretamente associadas ao estado global dos fluxos num determinado intervalo de tempo que, neste trabalho, foi de 30 segundos, em função de ser um intervalo de tempo usado para fins de detecção em outros trabalhos da literatura [56]. Os estados globais dos fluxos dizem respeito aos graus de dispersão ou concentração das distribuições das características de rede (endereços IPs e portas de origem e destino) dos fluxos IP compreendidos entre o tempo inicial e final da observação. Como indicador dos estados globais dos fluxos, as entropias dessas características foram calculadas para cada intervalo de 30 segundos existente no *dataset*. Além disso, os quatro últimos valores de entropia foram usados para se calcular as diferenças entre estes e o valor da entropia do intervalo de tempo corrente. As diferenças entre as entropias fornecem uma tendência de concentração ou dispersão nas características de redes observadas, cuja tendência é aprendida pelo processo de aprendizado de máquina. Os resultados destas diferenças foram acrescentados ao *dataset*, formando um novo *dataset* capaz de indicar a variação dos valores de entropia ao longo de uma janela saltante de tempo (últimos 2 minutos). Em adição, também foram calculadas outras métricas, tais como o desvio-padrão dos tamanhos dos pacotes e das durações dos fluxos, que também foram adicionadas ao *dataset* novo. A Figura 3.6 ilustra o processo de transformação do *dataset* inicial para *dataset* novo, com novos atributos.

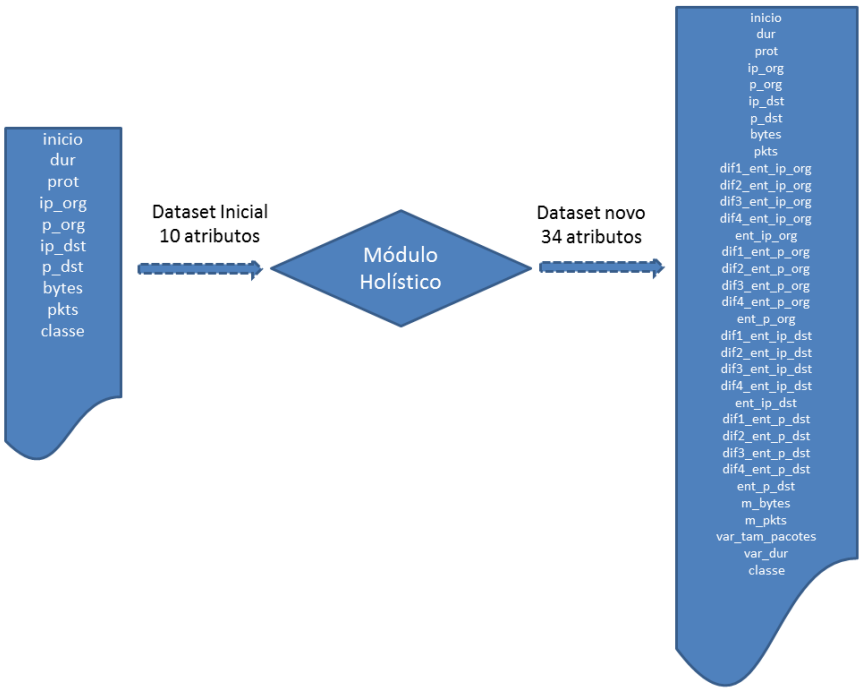

Figura 3.6: Processo de transformação dos datasets

E o Algoritmo 3.1 exibe, de forma simplificada, a programação efetuada no

módulo holístico para calcular os atributos avançados incorporados ao *dataset* novo.

Algoritmo 3.1: Cálculo dos atributos avançados

Entrada: *dataset* inicial

Saída: *dataset* novo

```
1  início
2      ler fluxos do dataset inicial;
3      se (intervalo_tempo = 30 segundos) então
4          incrementar num_intervalo_tempo;
5          calcular média da quantidade de bytes;
6          calcular média da quantidade de pacotes;
7          calcular o desvio-padrão do tamanho dos pacotes;
8          calcular o desvio-padrão da duração dos fluxos;
9          calcular entropia do endereço IP de origem;
10         calcular entropia da porta de origem;
11         calcular entropia do endereço IP de destino;
12         calcular entropia da porta de destino;
13         se (num_intervalo_tempo ≥ 5) então
14             calcular as diferenças entre os valores das entropias do endereço IP
                de origem, porta de origem, endereço IP de destino e porta de
                destino do intervalo de tempo corrente e seus respectivos valores
                nos 4 últimos intervalos de tempo;
15         fim
16         senão
17             retornar 0 para os resultados das diferenças entre os valores das
                entropias do endereço IP de origem, porta de origem, endereço IP
                de destino e porta de destino do intervalo de tempo corrente e seus
                respectivos valores nos 4 últimos intervalos de tempo;
18         fim
19         intervalo_tempo ←0;
20         mesclar os atributos primários dos fluxos do dataset inicial com os
            atributos avançados calculados;
21     fim
22 fim
```

Uma outra função desse módulo é o de calcular métricas como acurácia, precisão e os números de falsos positivos e falsos negativos, bem como as suas respectivas taxas, métricas estas que serão descritas no Capítulo 5. Os cálculos são feitos através

da chamada de um *script*, que pode ser interpretado como um submódulo, abstraído na Figura 3.2. Em adição, ele terá a tarefa de ficar verificando, num determinado intervalo de tempo que pode ser ajustado conforme a taxa com que novos fluxos chegam à rede de um dado ambiente de produção, a existência de novos conjuntos de fluxos para classificação. No surgimento de um novo conjunto de fluxos de dados, o módulo holístico poderá sinalizar ao SAMOA para que este dê início à classificação desses novos fluxos. Dessa maneira, o sistema estará continuamente sendo alimentado.

3.3 Orquestração do Sistema Proposto

Para que as etapas de funcionamento do sistema fossem viáveis, houve um estudo acerca dos componentes integrantes com o intuito de conhecer as suas características mais importantes e a possibilidade de integração entre elas. Os parágrafos seguintes serão dedicados à descrição da orquestração das etapas de operação do sistema de detecção proposto sintetizada na Figura 3.7.

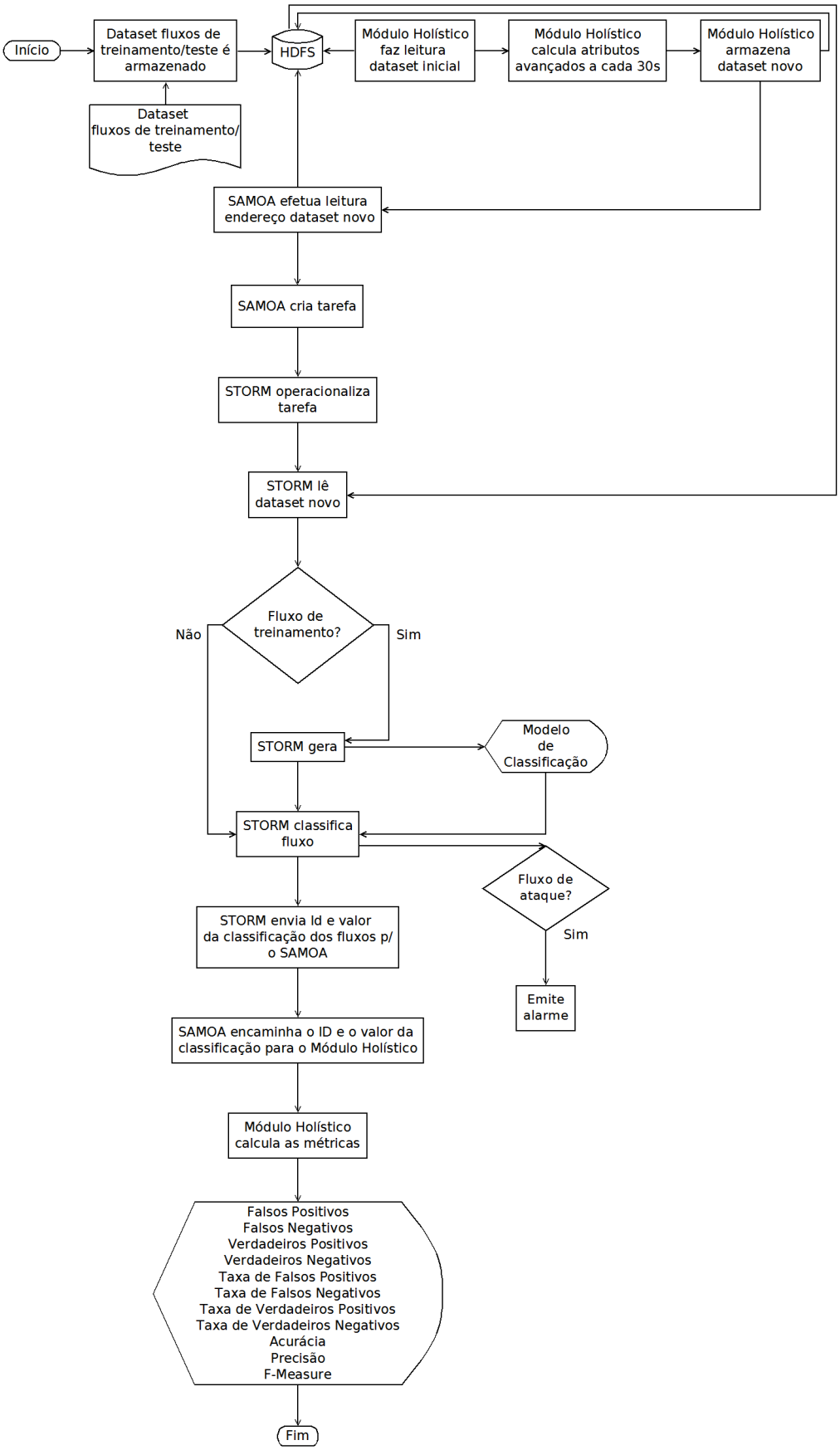

Figura 3.7: Fluxograma do processo de orquestração do sistema proposto

- 1. Depois da coleta de um *dataset* contendo fluxos para treinamento/teste do sistema, há o armazenamento dele no sistema distribuído de arquivos do Hadoop, o HDFS. A partir desse momento, ele passa a ser chamado de *dataset* inicial é já estará disponível para ser lido pelo módulo holístico.

- 2. Após o armazenamento do *dataset* inicial, o módulo holístico realiza a leitura dos fluxos do *dataset* inicial, agrupando-os em intervalos de tempo de 30 segundos baseados em seus *timestamps* para calcular novos atributos. Esses novos atributos são denominados atributos avançados: as entropias das características de rede (endereço IPs de origem, porta de origem, endereço IP de destino e porta de destino) e as suas respectivas diferenças para os valores das entropias dessas mesmas características de rede nos quatro intervalos de tempo anteriores ao intervalo de tempo corrente, o tamanho médio dos fluxos em *bytes*, número médio de pacotes dos fluxos, o desvio-padrão dos tamanhos dos pacotes e das durações dos fluxos existentes no intervalo de tempo mencionado.

- 3. Depois do cálculo dos atributos avançados, o módulo holístico realiza o armazenamento do *dataset* novo no HDFS, com os novos atributos incorporados, tornando-o disponível para que o SAMOA inicie o seu processo de leitura dos fluxos de treinamento e dos fluxos de teste para obtenção do endereço do *dataset* novo para iniciar a elaboração da tarefa de classificação.

- 4. Antes do início da explicação desta etapa, é importante salientar que a execução de leitura do SAMOA no HDFS só foi possível com uma alteração no código da interface de leitura de fluxos do SAMOA. Essa alteração foi indicada pelos integrantes da lista de desenvolvimento do SAMOA - o que mais tarde acabou virando um *patch* oficial do SAMOA. Neste ponto o SAMOA cria a tarefa a ser enviada ao STORM para fins de processamento. A configuração da tarefa inclui o seu grau de paralelismo e o método usado que, se for o Bagging, ainda é possível indicar a quantidade de modelos de classificação que irão fazer parte do processo de classificação a ser atribuído a determinado fluxo.

- 5. Ao receber a tarefa, o STORM inicia o processo de operacionalização da mesma: descompactação da tarefa para leitura de parâmetros e a execução dela, de acordo com as configurações lidas.

- 6. O STORM inicia o processo de leitura do *dataset* novo para a obtenção dos fluxos de treinamento e de teste. Se o fluxo lido for para treinamento do

sistema, ou seja, contiver uma marcação indicando a qual classe o mesmo pertence, o STORM o utilizará para a geração do modelo de classificação. Senão, o fluxo será classificado pelo STORM - que usará o modelo de classificação gerado na fase de treinamento do sistema. Nesta parte do código, caso um fluxo fosse classificado com um fluxo originado de um ataque DDoS, um alarme poderia ser emitido conforme a Figura 3.7 ilustra, porém essa funcionalidade não foi ativada por existir um conhecimento prévio do valor da classificação do fluxo.

- 7. A medida que os fluxos vão sendo analisados, os resultados das classificações dos fluxos vão sendo enviados para o SAMOA de modo a permitir sumarização dos resultados dos fluxos de treinamento. As classificações dos fluxos de teste também são enviadas. Exatamente nesse ponto houve mais uma alteração no código do SAMOA. Essa alteração foi necessária para interceptar os identificadores dos fluxos de teste com os seus respectivos valores de sua classificação, para que fossem enviados para o módulo holístico de modo a propiciar o calculo da acurácia, dos falsos positivos, dos falsos negativos e suas taxas.

- 8. O módulo holístico, então, recebe os pares de valores referentes ao identificador do fluxo e o seu valor de classificação. Alicerçado em marcações dos fluxos anteriores ao processo de classificação, são calculadas as métricas: falsos positivos, falsos negativos, verdadeiros positivos, verdadeiros negativos, taxa de falsos positivos, taxa de falsos negativos, taxa de verdadeiros positivos (também chamada de *Recall* ou Sensitividade), taxa de verdadeiros negativos, acurácia, precisão e F-*Measure*.

- 9. Nessa etapa, o módulo holístico já calculou os resultados e os torna disponíveis mediante a escrita dos mesmos em arquivo.

No que diz respeito às fases de treinamento e teste que, neste caso, pode ser entendida como fase de operação do sistema, elas são delimitadas em função da existência ou não da marcação de um determinado fluxo e de sua posição no *dataset*. Mais detalhadamente, os fluxos de treinamento foram marcados e colocados no início do *datastet*. Os fluxos de teste (para fase de operação do sistema) foram colocados posteriormente aos fluxos de treinamento. Assim, foram separadas as fases treinamento/operação do protótipo do sistema proposto.

Conforme explanação dos componentes e de como eles se interligam, foi montado um protótipo que foi posto em operação no cenário experimental apresentado no próximo capítulo.

4. Cenário Experimental

Neste capítulo será discutido o cenário onde o protótipo do sistema proposto no Capítulo 3 foi construído. Sua estrutura engloba o ambiente de criação do sistema, as ferramentas que auxiliaram a execução de alguma atividade relacionada ao sistema, os *datasets* utilizados e a metodologia empregada para a avaliação do protótipo do sistema.

4.1 Ambiente de Criação do Protótipo do Sistema Proposto

O protótipo do sistema de detecção de ataques DDoS proposto foi implementado usando máquinas virtuais criadas com o VirtualBox. O VirtualBox [1] é uma ferramenta originalmente desenvolvida pela empresa Innotek mas que atualmente pertence à Oracle. Destina-se a efetuar a virtualização de sistemas operacionais, podendo ser instalada em quase todos as plataformas atuais de sistemas operacionais, como Windows, Linux e Mac. Ela permite que em um computador comum sejam instalados e operados diversos sistemas operacionais ao mesmo tempo, dando a possibilidade de construir excelentes laboratórios virtuais para os mais diversos experimentos. Além do que foi supracitado, é um *software* livre regido pelos termos da GPL (*General Public License*) em sua segunda versão. É de fácil instalação e leve operação [57].

O VirtualBox foi instalado num servidor do Observatório Nacional com dois processadores Intel(R) Xeon(R) CPU E5-2643 v2 @ 3.50GHz com doze núcleos cada, 64GB RAM e Debian 8 como sistema operacional. Apesar das quantidades razoáveis de núcleos e memória, não foi possível utilizar os recursos totais dessa máquina, pois ela também era usada para experimentos dos alunos de pós-graduação do instituto.

[1] https://www.virtualbox.org/

A camada de processamento em lote, cujo o representante é o HDFS do Hadoop, que no sistema proposto propicia armazenamento, utilizou um conjunto de três máquinas virtuais em *cluster*: uma *master* e duas *slaves*. Já a camada de processamento em velocidade, composta pelo Storm e pelo SAMOA, foi configurada em uma única máquina virtual, a mesma máquina usada como *master* do Hadoop, para economizar recursos computacionais da máquina física que estava sendo compartilhada com as atividades acadêmicas do Observatório Nacional. Em ambas as máquinas virtuais, foram instalados o sistema operacional Debian 7 e dedicados quatro núcleos do processador Xeon(R) CPU E5-2643 v2 @ 3.50GHz. A máquina denominada *master*, por comportar maior quantidade de aplicações, recebeu 20 GB de RAM e as outras duas 6 GB. Quanto às versões dos componentes do sistemas, temos o Apache Hadoop 2.6.4, o Apache Storm 0.9.6 e o Apache SAMOA 0.3.0-incubating - estas eram as versões mais atuais no momento do desenvolvimento do sistema de detecção proposto.

4.2 Ferramentas de Apoio

Nesta seção serão apresentadas as ferramentas que não fizeram parte dos componentes do sistema proposto. No entanto, estas ferramentas auxiliaram diretamente na operação do mesmo, seja convertendo fluxos, como no caso do nProbe, ou no tratamento de *traces* de rede e na confecção de um importante módulo do sistema proposto, o módulo holístico, que foi implementado na linguagem AWK.

4.2.1 NProbe

O NetFlow Probe [2] (nProbe) é um extensível analisador de tráfego de redes de computadores disponibilizado em *software* livre. Foi projetado com a intenção de ser leve, eficiente e fácil de implantar. Ele inspeciona os pacotes de dados que trafegam pela rede de computadores e os sumariza em fluxos nos formatos NetFlow versão 5, NetFlow versão 9 ou IPFIX, dando a possibilidade dos administradores selecionarem quais dos atributos computados devem ser exportados para uma base de dados.

Dentre suas principais funcionalidades, destaca-se a coleta ou exportação de fluxos gerados pelos mais diversos equipamentos de rede, como, por exemplo, *gateways* de borda, *swithes* e roteadores. O nProbe opera tanto com IPv4 quanto com IPv6, possui uma vasta diversidade de *plug-ins* para diferentes cenários de uso, além de in-

[2] http://www.ntop.org/products/netflow/nprobe/

tegração com outras ferramentas de gerenciamento de redes. Em adição, o nProbe é projetado para analisar redes multi-Gbit em velocidade máxima, inspecionando centenas de milhares de pacotes por segundo [58]. O nProbe oferece ainda a possibilidade de enviar os fluxos monitorados para um outro coletor de fluxos ou para um servidor de banco de dados. O nProbe pode também ler arquivos de captura de pacotes, em formato pcap, e fazer a conversão para fluxos.

4.2.2 AWK

O AWK [59] é uma linguagem de programação construída para fazer a manipulação de arquivos de texto de modo a extrair informações do mesmo de forma fácil. A sua operação básica consiste em ler sucessivamente em ordem as linhas dos arquivos de entrada procurando por linhas cujo padrão seja igual ao determinado pelo usuário. Cada padrão definido pelo usuário dever ser acompanhado de uma ação que deve ser executada a cada ocorrência desse padrão.

A alimentação de um programa escrito em AWK é dividida em *records*, que podem ser compreendidas em linhas, terminados por um separador de *records*. Um *record*, por sua vez, é dividido em *fields*, cujo separador pode ser alterado pelo usuário. Na prática, a linguagem awk se encaixa em duas áreas:

- Gerador de relatórios: pela facilidade de extrair e calcular informações através do processamento dos arquivos de entrada; e
- Conversor de dados: pois, também pela característica de seu processamento, pode transformar os dados gerados pela saída de programas em um formato adequado para alimentar um outro sistema.

Devido ao seu potencial supracitado, se tornou uma ferramenta bastante útil ao trabalho, utilizada para a formatação dos diversos *traces* de rede e na programação do módulo holístico.

4.2.3 Hping3 e TCPDUMP

O hping3 [3] é um *software* livre gerador e analisador de pacotes para o protocolo TCP/IP. É comumente usado pelos administradores de rede de computadores com foco em segurança para testar e auditar as suas redes e aplicações de segurança.

[3] http://www.hping.org/hping3.html

O TCPDUMP [4] é um utilitário amplamente conhecido pelos profissionais de rede computadores. É considerada uma das mais completas ferramentas para o monitoramento de pacotes transmitidos na rede, mostrando desde o cabeçalho dos pacotes até conteúdo dos mesmos.

As duas ferramentas foram usadas de maneira complementar, explorando seus potenciais para auxiliar na geração dos fluxos do *dataset* de treinamento, base de dados inicialmente fornecida ao sistema de detecção de ataques proposto, para fins de construção e ajustes dos modelos que serão usados na classificação dos fluxos de rede.

4.3 *Datasets* Usados

Como insumo alimentador do sistema proposto foram usados dois *datasets*. O primeiro, formado a partir da captura de fluxos de tráfego real, livres de ataques DDoS, obtidos da rede do Observatório Nacional - representando fluxos normais de rede - misturado com fluxos de ataque simulados em ambiente controlado. Já o segundo *dataset* foi o CTU-13, publicamente disponível em [60]. As subseções seguintes versam a respeito desses dois *datasets*

4.3.1 *Datatet* de Fluxos Normais + Fluxos de Ataque Simulado

Nesta subseção serão abordados os processos de captura dos fluxos normais, geração e captura dos fluxos de ataque, conversão dos arquivos de pacotes para arquivos de texto contendo fluxos IP e os critérios adotados para a composição dos fluxos de treinamento.

4.3.1.1 Captura dos Fluxos Normais

A geração dos arquivos de captura de tráfego legítimo consistiu na interceptação, com o utilitário TCPDUMP, durante 7 dias, do tráfego de dados da rede de computadores do Observatório Nacional, a partir de um servidor conectado numa porta do *switch* principal do *datacenter*. A técnica de espelhamento de porta estava configurada nessa porta, espelhando todo o tráfego passante nas portas de entrada/-saída para a Internet (*Outside)*, rede interna (*Inside*) e DMZ (*Desmilitarized Zone* ou Zona Desmilitarizada) como mostra a Figura 4.1. O espelhamento de porta de *switches* é empregado quando se necessita que o tráfego de dados passante numa ou

[4] http://www.tcpdump.org/

mais portas do dispositivo seja copiado para uma outra porta específica para fins de segurança contra ameaças, resolução de problemas ou para fins de análise [61].

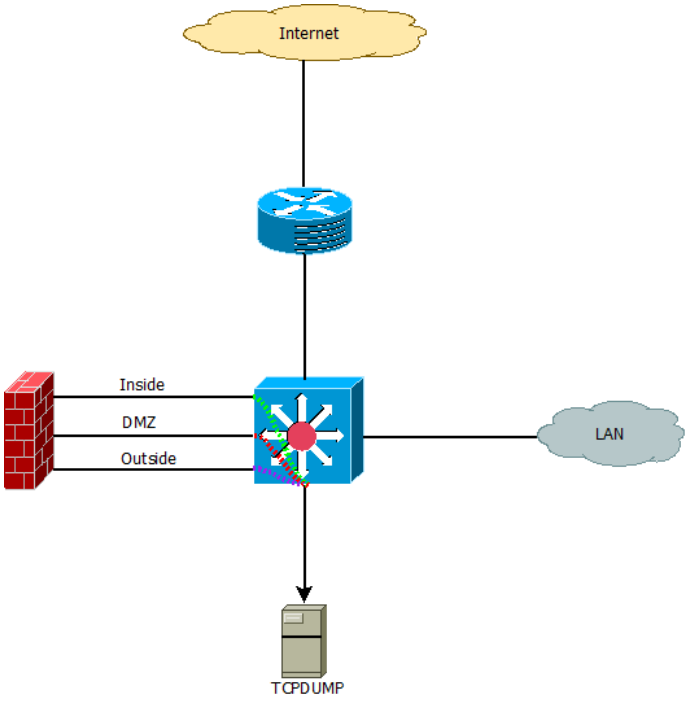

Figura 4.1: Captura do tráfefo de dados da rede do Observatório Nacional, onde as linhas tracejadas indicam o espelhamento de portas

Os pacotes de dados foram considerados legítimos ou normais porque os sistemas de segurança da instituição não indicaram quaisquer alterações nos indicadores de anomalias costumeiramente associados ao comportamento do tráfego, alterações estas que poderiam ter sido ocasionadas por ataques DDoS durante o período de captura do tráfego de dados. Os arquivos de captura de pacotes, conhecidos como arquivos ".pcap", foram gerados a cada 5 minutos, como é comumente utilizado pela maioria dos *softwares* que fazem a análise do tráfego de rede. Ademais, utilizando-se esse tempo para as gerações dos arquivos, adicionava-se uma facilidade na leitura dos mesmos, pois não ficavam com tamanho demasiadamente grande, o que exigiria maior capacidade de leitura dos computadores utilizados para conferência de seus conteúdos.

Antes da definição das proporções entre os fluxos normais e de ataques, e depois da conversão para fluxos IP - ambas explicadas em seções mais a frente -, o tráfego legítimo foi caracterizado de modo a quantificar e conhecer os fluxos nele existentes. Os valores obtidos foram de, aproximadamente 17.000.000 de fluxos em 7 dias, mais de 16.000 fluxos/min, mais de 24.000 fluxos/dia, e mais de 150 protocolos de-

tectados, totalizando aproximadamente 596 GB de arquivos de captura de pacotes. A Figura 4.2 exibe os 15 protocolos mais frequentes. Nela é possível observar que na sétima posição há um protocolo desconhecido (*"unknown"*), devido ao fato dos fluxos utilizarem portas com valores altos, diferentes das portas mais conhecidas e que utilizam valores mais baixos.

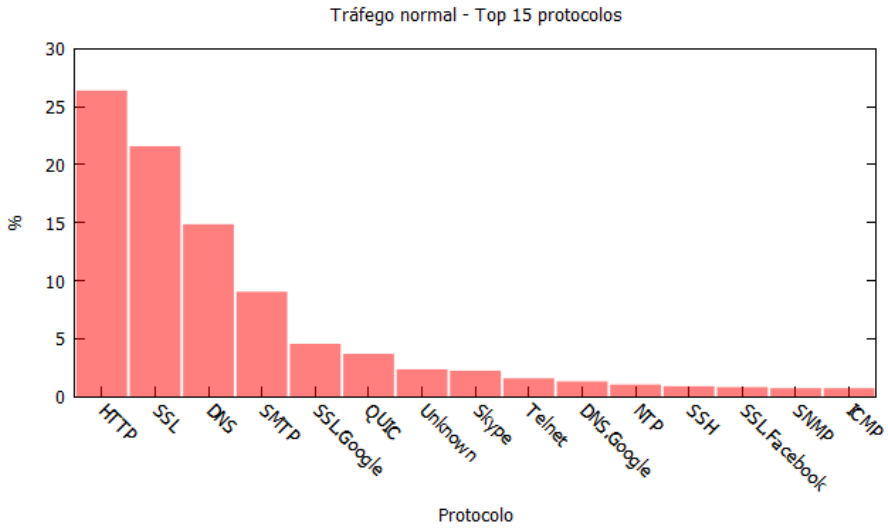

Figura 4.2: Os 15 protocolos mais frequentes nos fluxos IPs do tráfego normal

4.3.1.2 Criação e Captura dos Fluxos de Ataque

O hping3 foi o utilitário responsável pela simulação em ambiente controlado dos ataques DDoS, forjando diversos endereços de IPs de origem atacando um determindo IP de destino, isto é, diversos *hosts* de origem atacando um *host* vítima. No *host* vítima os pacotes foram capturados, também, com o auxílio do TCPDUMP. Os ataques DDoS selecionados para o experimento foram o ICMP flood e o UDP flood, porque são apontados como sendo as maiores frequências de ataques DDoS no Volume XI do relatório mundial de segurança em infraestrutura da empresa ARBOR Networks [5] - empresa mundialmente reconhecida por sua atuação nos campos pesquisa, identificação e mitigação de ataques DDoS. E além disso, são os tipos de ataques DDoS encontrados nos cenários do *dataset* CTU-13 que foram selecionados para teste do sistema proposto. Abaixo, a Figura 4.3 exibe a síntese do processo citado.

[5]https://www.arbornetworks.com/

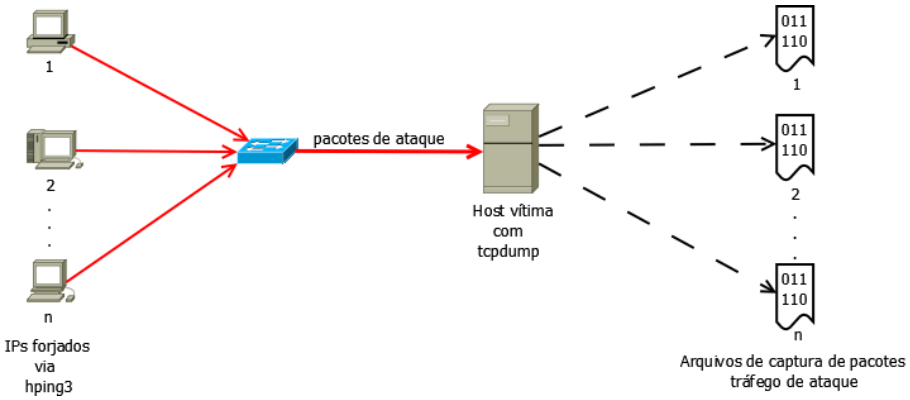

Figura 4.3: Ataque DDoS simulado e geração dos arquivos de captura de pacotes do tráfego de ataque

4.3.1.3 Conversão dos Arquivos de Pacotes em Fluxos IP

De posse dos arquivos de captura de pacotes, referentes ao tráfego de pacotes de dados livres de ataque e ao tráfego de pacotes de dados contendo ataque, usando o nProbe [6], os arquivos de pacotes com o seu respectivo tipo de tráfego foram convertidos para fluxos IPs. O processo de conversão foi configurado para considerar fluxos com duração máxima de 30 segundos, para que o mesmo não demorasse a ser criado no arquivo de exportação, permitindo que fluxos de ataques com longa duração pudessem ser tratados em tempo hábil. Caso o fluxo tivesse duração superior a 30 segundos, um novo fluxo era gerado. O nProbe, que utiliza o recurso NetFlow - explicado na subseção 2.10.2, foi selecionado em virtude de exportar os fluxos para um arquivo de texto no formato NetFlow, no qual os usuários têm liberdade de indicar quais atributos constituirão os fluxos exportados. Em adição, tem o fato da licença do nProbe, na sua versão com maiores recursos, ser livre para instituições sem fins lucrativos, dando maior flexibilidade de utilização do mesmo. A Figura 4.4 ilustra a questão da conversão dos arquivos de pacotes com o tráfego de dados da rede.

[6] http://www.ntop.org/products/netflow/nprobe/

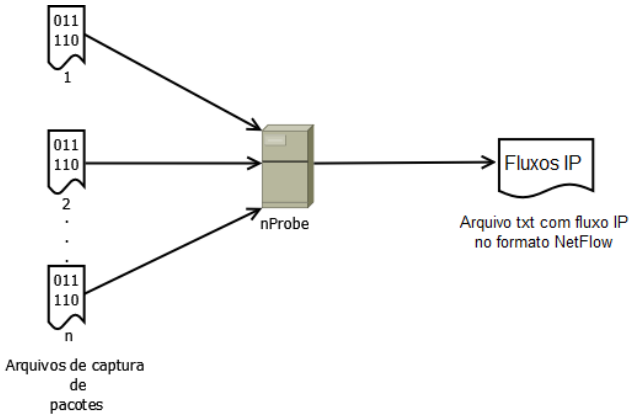

Figura 4.4: Processo de conversão dos arquivos de captura de pacotes para fluxos de dados no formato NetFlow

4.3.1.4 Composição dos Fluxos de Treinamento

Passa a fase de captura e conversão dos fluxos de normais e de ataque, pode-se, então, dar início à definição da proporção dados de treinamento e dados de teste, isto é, determinar a quantidade de fluxos de treinamento que iriam fazer parte do *dataset* inicial que o sistema usaria. No entanto, na literatura não foi encontrada referência alguma à exata proporção entre a quantidade de fluxos de treinamento e quantidade de fluxos de teste. Dado esse cenário, foi adotada uma estratégia para definir a quantidade de fluxos de treinamento que não facilitasse a classificação e nem causasse um super ajuste aos fluxos de treinamento. Para tanto, foi tomado como ponto referencial a quantidade de fluxos de ataques DDoS dos cenários (4, 10 e 11) do *dataset* CTU-13 com fluxos bidirecionais, usando 80% dessa quantidade para compor os fluxos de treinamento. Esse critério foi adotado porque os cenários com fluxos bidirecionais apresentam menor quantidade de fluxos.

4.3.2 *Dataset* CTU-13

O *dataset* CTU-13 [60] foi criado na Universidade CTU da Repúplica Tcheca no ano de 2011 em virtude da necessidade dos seus autores validarem o seus trabalhos quanto à detecção de *botnets*. Porém, não havia bons *datasets* públicos realísticos disponíveis, pois deveriam conter tráfego normal, de fundo (ou *background*) e de *botnets*, cujo balanceamento entre esses tráfegos representassem o comportamento real de uma rede de computadores. Inicialmente, foi criado um *dataset* com fluxos unidirecionais - o que foi publicado no artigo [60]. Mais tarde, foi elaborado um novo *dataset* com fluxos bidirecionais. Neste trabalho foram usados os dois *datasets*.

A criação desses *datasets* teve como premissas conter ataques reais de diferentes *botnets*, tráfego desconhecido de uma grande rede de computadores, fluxos corretamente etiquetados para treinamento e validação, muitas *botnets* infectadas ao mesmo tempo, para poder capturar padrões de sincronismo, e arquivos de fluxos no formato NetFlow que mantenham a privacidade de seus usuários. Dadas as condições iniciais, foi montada uma topologia usando máquinas virtuais com os sistemas operacionais mais usados na época (Windows XP e Debian) para criar os 13 cenários integrantes dos *datasets*. A Tabela 4.1 mostra as características dos 13 cenários criados para ambas as orientações dos fluxos (entrada ou saída), com os seguintes significados para as siglas usadas: Cen = cenário, CF = Click Fraud, PS = Port Scan, FF = Fast Flux, e AU = compilado e controlado pelos autores.

Tabela 4.1: Características dos cenários: unidirecional e bidirecional

Cen	**IRC**	**SPAM**	**CF**	**PS**	**DDoS**	**FF**	**P2P**	**AU**	**HTTP**	**Observação**
1	X	X	X							
2	X	X	X							
3	X			X				X		
4	X				X			X		UDP e ICMP DDoS.
5		X		X					X	Scan web proxies.
6				X						C&C proprietário. RDP.
7									X	Hosts chineses.
8				X						C&C proprietário. Net-BIOS,STUN.
9	X	X	X	X						
10	X				X			X		UDP DDoS.
11	X				X			X		ICMP DDoS.
12							X			Sincronização.
13		X		X					X	Captcha. Web mail.

Cada um dos 13 cenários gerados originou um arquivo de captura de pacotes com quantidade de fluxos, pacotes, *botnets*, tamanhos e duração diferentes. A Tabela 4.2 quantifica essas particularidades de cada arquivo de captura de pacotes para ambas as orientações direcionais dos *datasets*.

Tabela 4.2: Características dos arquivos de captura de pacotes: unidirecional e bidirecional

Cen	Duração(hs)	Qtd Pacotes	Qtd NetFlows	Tamanho	Botnet	Qtd Botnets
1	6,15	71.971.482	11.231.035	52 GB	Neris	1
2	4,21	71.851.300	7.037.972	60 GB	Neris	1
3	66,85	167.730.395	15.202.061	121 GB	Rbot	1
4	4,21	62.089.135	4.238.045	53 GB	Rbot	1
5	11,63	4.481.167	7.710.910	37,6 GB	Virut	1
6	2,18	38.764.357	2.579.105	30 GB	Menti	1
7	0,38	7.467.139	454.175	5,8 GB	Sogou	1
8	19,5	155.207.799	11.993.935	123 GB	Murlo	1
9	5,18	115.415.321	8.087.513	94 GB	Neris	10
10	4,75	90.389.782	5.180.852	73 GB	Rbot	10
11	0,26	6.337.202	40.836	5,2 GB	Rbot	3
12	1,21	13.212.268	1.262.790	8,3 GB	NSIS.ay	3
13	16,36	50.888.256	6.425.345	34 GB	Virut	1

Os arquivos de captura de pacotes criados passaram por um processo de conversão em fluxos para serem devidamente especificados com as suas respectivas marcações à respeito de suas natureza: normal, *background* ou fluxos de comando e controle (CC). A Tabela 4.3 exibe a distribuição dos fluxos dos 13 cenários criados para o *dataset* com orientação bidirecional.

Tabela 4.3: Distribuição da quantidade de fluxos bidirecionais

Cen	Total	Botnet	Normal	C&C	Background
1	2.824.636	39.933 (1,41%)	30.387 (1,07%)	1.026 (0,03%)	2.753.290 (97,47%)
2	1.808.122	18.839 (1,04%)	9.120 (0,5%)	2.102 (0,11%)	1.778.061 (98,33%)
3	4.710.638	26.759 (0,56%)	116.887 (2,48%)	63 (0,001%)	4.566.929 (96,94%)
4	1.121.076	1.719 (0,15%)	25.268 (2,25%)	49 (0,004%)	1.094.040 (97,58%)
5	129.832	695 (0,53%)	4.679 (3,6%)	206 (1,15%)	124.252 (95,7%)
6	558.919	4.431 (0,79%)	7.494 (1,34%)	199 (0,03%)	546.795 (97,83%)
7	114.077	37 (0,03%)	1.677 (1,47%)	26 (0,02%)	112.337 (98,47%)
8	2.954.230	5.052 (0,17%)	72.822 (2,46%)	1.074 (2,4%)	2.875.282 (97,32%)
9	2.753.884	179.880 (6,5%)	43.340 (1,57%)	5.099 (0,18%)	2.525.565 (91,7%)
10	1.309.791	106.315 (8,11%)	15.847 (1,2%)	37 (0,002%)	1.187.592 (90,67%)
11	107.251	8.161 (7,6%)	2.718 (2,53%)	3 (0,002%)	96.369 (89,85%)
12	325.471	2.143 (0,65%)	7.628 (2,34%)	25 (0,007%)	315.675 (96,99%)
13	1.925.149	38.791 (2,01%)	31.939 (1,65%)	1.202 (0,06%)	1.853.217 (96,26%)

A relação das distribuições dos fluxos nos 13 cenários do *dataset* com orientação unidirecional é mostrada na Tabela 4.4

Tabela 4.4: Distribuição da quantidade de fluxos unidirecionais

Cen	Total	Background	Botnet	Normal
1	10.612.259	10.124.854 (95,40%)	94.972 (0,89%)	392.433 (3,69%)
2	6.351.188	6.071.419 (95,59%)	54.433 (0,85%)	225.336 (3,54%)
3	15.202.060	14.381.899 (94,60%)	75.891 (0,49%)	744.270 (4,89%)
4	4.238.038	3.895.469 (91,91%)	6.466 (0,15%)	336.103 (7,93%)
5	455.540	416.267 (91,37%)	2.129 (0,46%)	37.144 (8,15%)
6	2.158.748	2.031.967 (94,12%)	4.927 (0,22%)	121.854 (5,64%)
7	454.174	425.611 (93,71%)	293 (0,06%)	28.270 (6,22%)
8	11.993.934	11.451.205 (95,47%)	12.063 (0,10%)	530.666 (4,42%)
9	7.627.037	6.881.228 (90,22%)	383.215 (5,02%)	362.594 (4,75%)
10	5.180.851	4.535.493 (87,54%)	323.441 (6,24%)	321.917 (6,21%)
11	408.835	119.933 (29,33%)	277.892 (67,97%)	11.010 (2,69%)
12	408.835	119.933 (29,33%)	277.892 (67,97%)	11.010 (2,69%)
13	1.299.090	1.218.140 (93,76%)	21.760 (1,67%)	59.190 (4,55%)

4.4 Avaliação do Sistema Proposto

A avaliação do sistema proposto deu-se pela alimentação do sistema com subconjuntos de *datasets* realísticos oriundos do *dataset* CTU-13, tendo seus fluxos marcados como sendo normal ou ataque DDoS. Ou seja, o *dataset* CTU-13 foi, via *scripts*, tratado de modo a conter apenas esses dois tipos de classe. Houve também a coleta de tráfego lícito da rede de computadores do Observatório Nacional, com fluxos de dados livres de ataque segundo as ferramentas de segurança da instituição, e emulação de ataques DDoS em ambiente controlado para formar um conjunto de fluxos de treinamento. Estes fluxos de treinamento foram posteriormente mesclados com os fluxos oriundos do *dataset* CTU-13 para compor o chamado "*dataset* inicial", *dataset* este usado para alimentar o sistema de detecção proposto e ilustrado na Figura 4.5.

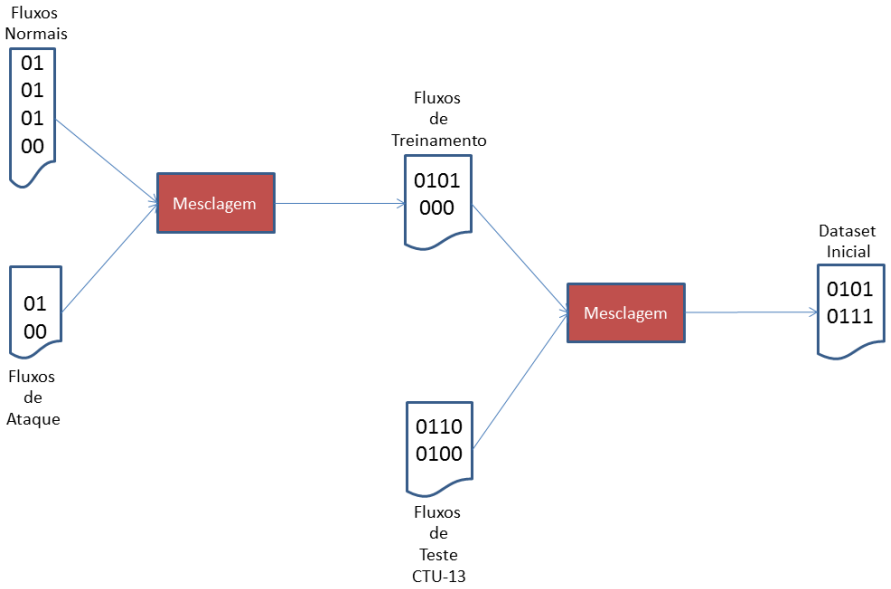

Figura 4.5: Criacão do dataset inicial para alimentar o sistema proposto

Ao contrário do processamento em lote de dados, que separa fisicamente os dados de treinamento e os dados de teste, na operação do processamento de fluxos de dados essa separação física não existe. Os fluxos vão todos juntos no mesmo *dataset*, sendo que a diferença é que os fluxos de treinamentos vêm marcados com o valor da classe a qual pertence indicada no último campo. Já nos fluxos de teste, o último campo que indicaria a qual classe o fluxo pertenceria fica sem valor, vazio. De posse dos subconjuntos de *datasets* CTU-13 e do *dataset* formado a partir da coleta do tráfego real e dos ataques emulados, foram criados três grupos de *datasets*:

- Grupo 1: tanto os fluxos para treinamento quanto os fluxos para teste são oriundos do *dataset* CTU-13 com fluxos bidirecionais. Sua seleção, ao invés do *dataset* CTU-13 com fluxos unidirecionais, se justifica em função desse apresentar, segundo seus autores, mais vantagens, tais como maiores detalhes relacionados às marcações dos fluxos, maiores detalhes quanto aos fluxos e melhor diferenciação entre clientes e servidores.

- Grupo 2: fluxos normais de treinamento provenientes do tráfego da rede do ON e fluxos de ataques oriundos de emulação em ambiente controlado. Quanto aos fluxos de teste, tanto os fluxos normais quantos os de ataque vêm do *dataset* CTU-13 com fluxos unidirecionais.

- Grupo 3: fluxos normais de treinamento provenientes do tráfego da rede do ON e fluxos de ataques oriundos de emulação em ambiente controlado. Quanto aos fluxos de teste, tanto os fluxos normais quantos os de ataque vêm do *dataset* CTU-13 com fluxos bidirecionais.

A razão de se dividir em três grupos fluxos IPs, deve-se ao fato de se desejar confrontar uma situação mais próxima da realidade. Situação esta, na qual os fluxos disponíveis para treinamento são diferentes daqueles fluxos que serão provenientes de ataques (Grupo 2 e Grupo 3) contra uma prática costumeiramente, utilizada nos estudos de detecção de ataques DDoS utilizando aprendizado de máquina, onde tanto os fluxos de treinamento e de teste se originam da mesma fonte (Grupo 1).

Com a função de evitar qualquer fator tendencioso na criação dos grupos, um grupo apenas com fluxos coletados na rede do Observatório Nacional e com ataques emulados em ambiente controlado não foi criado. A geração desses grupos visou avaliar a eficiência do sistema de detecção proposto em condições severas e próximas da realidade da operação de uma rede computadores, onde os fluxos disponíveis para treinamento são normalmente menores e com características de rede diferentes dos fluxos de ataque.

Cada um desses três grupos foi dividido em quatro diferentes *datasets*, cada qual contendo um diferente conjunto de atributos gerados pelo módulo holístico para cada fluxo, cujas características são:

- 1º conjunto - chamado de *dataset* completo: atributos dos fluxos incluem medidas de entropia e suas diferenças ao longo de uma janela de tempo, tamanhos médios em bytes e pacotes, desvios-padrões da duração do fluxo e do tamanho dos pacotes;

- 2º conjunto - chamado de *dataset* com entropias: atributos dos fluxos incluem medidas de entropia e suas diferenças ao longo de uma janela de tempo, mas sem os tamanhos médios em bytes e pacotes, e sem os desvios-padrões da duração do fluxo e do tamanho dos pacotes;

- 3º conjunto - chamado de *dataset* com estatísticas: atributos dos fluxos incluem tamanhos médios em bytes e pacotes e desvios-padrões da duração do fluxo e do tamanho dos pacotes, mas sem as medidas de entropia e suas diferenças ao longo de uma janela de tempo;

- 4º conjunto - chamado de *dataset* puro: atributos dos fluxos não incluem as medidas geradas pelo módulo holístico.

O objetivo da elaboração desses quatro conjuntos de atributos foi averiguar a contribuição das medidas de entropias e suas diferenças ao longo de uma janela de tempo, bem como as outras métricas estatísticas calculadas pelo módulo holístico.

O cenários supracitados partem de um *dataset* completo cujos os atributos foram divididos em duas categorias em termos de natureza: primários e avançados. Os atributos primários são aqueles obtidos diretamente pelo próprio processo de conversão dos arquivos de captura de pacotes em fluxos IP, tais como: inicio, dur, prot, ip_org, p_org, ip_dst, p_dst, bytes e pkts. Já os atributos avançados são aqueles derivados - calculados via módulo holístico - a partir dos atributos primários, como: dif1_ent_ip_org, dif2_ent_ip_org, dif3_ent_ip_org, dif4_ent_ip_org, ent_ip_org, dif1_ent_p_org, dif2_ent_p_org, dif3_ent_p_org, dif4_ent_p_org, ent_p_org, dif1_ent_ip_dst, dif2_ent_ip_dst, dif_ent_ip_dst, dif4_ent_ip_dst, ent_ip_dst, dif1_ent_p_dst, dif2_ent_p_dst, dif3_ent_p_dst, dif4_ent_p_dst, ent_p_dst, m_bytes, m_pkts, var_tam_pacotes, var_dur, visando oferecer um maior entendimento do comportamento holístico dos conjuntos de fluxos a cada intervalo de tempo de 30 segundos. Os atributos avançados têm a intenção de complementar a formulação do modelo de classificação dos fluxos, uma vez que os atributos primários fornecem uma informação individualizada para cada fluxo, sem considerar o comportamento geral do tráfego agregado num dado momento. No Apêndice A são exibidos os atributos do *dataset* completo com suas respectivas descrições.

Após a finalização da confecção de tais conjuntos e a conversão dos fluxos para o formato de arquivo "arff"(Attribute-Relation File Format), foram executadas classificações utilizando os mesmos. As classificações utilizadas pelo sistema usaram os

métodos VHT, *Bagging* e *Adaptive Bagging* - todos oferecidos pelo SAMOA. Apesar destes métodos utilizarem árvore de decisão para predizer a classificação do fluxo, eles utilizam técnicas diferentes para tal atividade. O VHT usa árvores de decisão conforme explicado na Seção 2. O modelo de classificação formado pelo *Bagging* usa conjuntos de modelos de classificação baseados no VHT. Já o *Adaptive Bagging* usa um detector de mudanças para decidir se vai "retreinar" o modelo - ideal para fluxos de dados cujas características mudam conforme o passar do tempo.

É importante destacar que o módulo holístico do protótipo testado do sistema não usou a função de detecção de novos conjuntos de fluxos de dados, deixando a ativação da mesma para uma futura implementação em ambiente de produção. Como consequência, todos os fluxos utilizados pelo sistema, sejam fluxos de treinamento ou fluxos de teste, foram obtidos a partir dos *datasets* montados.

5. Resultados

Este capítulo abordará os resultados obtidos após a avaliação do sistema proposto conforme tratado na seção 4.4. Primeiramente, as métricas utilizadas serão apresentadas com as devidas justificativas para terem sido usadas. Depois, os resultados obtidos para os arranjos grupais criados serão expostos. Então, os haverá uma análise dos resultados. E para encerrar o capítulo, algumas considerações de natureza operacional do protótipo do sistema proposto serão realizadas.

5.1 Apresentação das Métricas Utilizadas

As métricas utilizadas neste trabalho são aquelas normalmente encontradas na literatura para a aferição da eficiência de um sistema de detecção de ataques DDoS. Dentre os diversos indicadores de qualidades existentes [18], foram selecionados os seguintes: quantidade de falsos positivos, quantidade de falsos negativos, quantidade de verdadeiros positivos, quantidade de verdadeiros negativos, taxa de falsos positivos, taxa de falsos negativos, taxa de verdadeiros positivos, taxa de verdadeiros negativos, acurácia, precisão, F-*Measure*. Os falsos positivos, falsos negativos, verdadeiros positivos e verdadeiros negativos, assim como as suas respectivas taxas, são itens extremamente importantes, pois a perseguição da otimização deles é um objetivo constante dos sistemas de detecção de ataques. Através da acurácia é possível saber quão correto o sistema está desempenhando a sua tarefa de identificação de ataques, ao passo que contempla a porcentagem de acertos e falhas e, também, a quantidade de alarmes falsos. A meta da utilização da métrica precisão é definir em termos percentuais como o sistema identifica fluxos normais ou fluxos de ataques. E por último, a F-*Measure*, que através do uso de um recurso matemático denominado média harmônica, considera as propriedades da precisão e da taxa de verdadeiros positivos, ou *recall*, em seu cálculo.

- Falsos positivos (FP): é a quantidade de fluxos normais classificados como fluxos de ataque;
- Falsos negativos (FN): é a quantidade de fluxos de ataque classificados como fluxos normais;
- Verdadeiros positivos (VP): é a quantidade de fluxos de ataque classificados como fluxos de ataques;
- Verdadeiros negativos (VN): é a quantidade de fluxos de normais classificados como fluxos normais;
- Taxa de falsos positivos (Tfp): é a razão percentual entre o número de falsos positivos (FP) e a quantidade de fluxos normais. É definida pela seguinte equação:

$$\text{Tfp} = \frac{\text{FP}}{\text{qtd total de fluxos normais}} * 100 \tag{5.1}$$

- Taxa de falsos negativos (Tfn): é a razão percentual entre a quantidade de falsos negativos (FN) e a quantidade de fluxos de ataque. É definida pela seguinte equação:

$$\text{Tfn} = \frac{\text{FN}}{\text{qtd total de fluxos de ataque}} * 100 \tag{5.2}$$

- Taxa de verdadeiros positivos (Tvp) ou *Recall* ou Sensitividade: é a razão percentual entre o número de verdadeiros positivos (VP) e a quantidade de fluxos de ataque. É definida pela seguinte equação:

$$\text{Tvp} = \frac{\text{VP}}{\text{qtd total de fluxos de ataque}} * 100 \tag{5.3}$$

- Taxa de verdadeiros negativos (Tvn): é a razão percentual entre o número de verdadeiros negativos (VN) e a quantidade de fluxos normais. É definida pela seguinte equação:

$$\text{Tvn} = \frac{\text{VN}}{\text{qtd total de fluxos normais}} * 100 \tag{5.4}$$

- Acurácia (Acur): é a razão percentual entre a quantidade de fluxos corretamente classificados e a quantidade de fluxos analisados. É definida pela seguinte equação:

$$\text{Acur} = \frac{\text{qtd de fluxos analisados - (FP+FN)}}{\text{qtd total de fluxos analisados}} * 100 \tag{5.5}$$

- Precisão (Prec): é a razão percentual entre a quantidade de fluxos de ataque corretamente classificados (VP) e a quantidade de fluxos classificados como fluxos de ataque (VP + FP). É definida pela seguinte equação:

$$\text{Prec} = \frac{\text{VP}}{\text{VP} + \text{FP}} * 100 \tag{5.6}$$

- F-*Measure* (F-M): é a média harmônica entre precisão e *recall*. É definida pela seguinte equação:

$$\text{F-M} = \frac{2}{\frac{1}{\text{Prec}} + \frac{1}{Recall}} * 100 \tag{5.7}$$

5.2 Apresentação dos Resultados Obtidos

Nesta seção serão apresentados os resultados obtidos para o sistema de detecção proposto usando os métodos VHT, Bagging e Adaptive Bagging. Os resultados foram extraídos para três diferentes grupos de *datasets* envolvendo diferentes arranjos para treinamento e teste:

- Grupo 1: tanto os fluxos para treinamento quanto os fluxos para teste são oriundos do *dataset* CTU-13 com fluxos bidirecionais, sendo 80% usados para treinamento e 20% para teste;

- Grupo 2: os fluxos treinamento são provenientes do tráfego da rede do ON acrescido de fluxos de ataques oriundos de emulação em ambiente controlado, enquanto que os fluxos de teste são oriundos do *dataset* CTU-13 com fluxos unidirecionais;

- Grupo 3: os fluxos treinamento são provenientes do tráfego da rede do ON acrescido de fluxos de ataques oriundos de emulação em ambiente controlado, enquanto que os fluxos de teste são oriundos do *dataset* CTU-13 com fluxos bidirecionais.

Os experimentos do Grupo 1 seguem a prática tradicionalmente adotada na literatura, onde um mesmo *dataset* é usado para treinamento e teste. Os experimentos dos Grupos 2 e 3 procuram se aproximar de uma situação mais realista do ponto de vista operacional, onde possivelmente os *datasets* de treinamento são obtidos de fontes externas, com padrões e parâmetros de tráfego distintos com relação à rede gerenciada.

Os experimentos para cada grupo de *datasets* foram separados conforme os três cenários de ataques DDoS presentes no *dataset* CTU-13, seja com fluxos unidirecionais ou bidirecionais (cenários 4, 10 e 11). Em cada cenário, as métricas computadas consideraram quatro diferentes conjuntos de atributos com a intenção de representar as diferentes contribuições das medidas estatísticas provenientes do módulo holístico na detecção dos ataques. Os quatro conjuntos são:

- atributos **primários**: não usam as medidas do módulo holístico;

- atributos **primários + entropias**: usam apenas as medidas de entropia do módulo holístico;

- atributos **primários + estatísticos**: usam apenas as estatísticas básicas de fluxo do módulo holístico, sem considerar as medidas de entropia;

- e atributos **primários + entropias + estatísticos**: usam todas as medidas estatísticas do módulo holístico, incluindo as de entropia.

A seguir são apresentados os resultados para cada grupo de *datasets*.

5.2.1 Resultados para o Grupo 1

- Resultados usando o método VHT

  Como pode ser visualizado na Tabela 5.1, o método VHT proporcionou 100% de acurácia (Acur), de precisão (Prec), de taxa de verdadeiros positivos (Tvp), de taxa de verdadeiros negativos (Tvn) e valor máximo na F-*Measure* (F-M) não classificando erradamente nenhum fluxo normal como fluxo de ataque e nem fluxo de ataque como fluxo normal - ou seja, FP e FN iguais a 0 - o que ocasionou 0 nas taxas de falsos positivos e negativos.

Tabela 5.1: Resultados do grupo 1 usando o método VHT

CEN CTU-13	**Atributos**	**Acur(%)**	**Prec(%)**	***F-M***	**FP**	**FN**	**VP**	**VN**	**Tfp(%)**	**Tfn(%)**	**Tvp(%)**	**Tvn(%)**
4	primários	100,0	100,0	10000,0	0	0	515	223.698	0,0	0,0	100,0	100,0
	primários + entropias	100,0	100,0	10000,0	0	0	515	223.698	0,0	0,0	100,0	100,0
	primários + estatísticos	100,0	100,0	10000,0	0	0	515	223.698	0,0	0,0	100,0	100,0
	primários + entropias + estatísticos	100,0	100,0	10000,0	0	0	515	223.698	0,0	0,0	100,0	100,0
10	primários	100,0	100,0	10000,0	0	0	21.270	240.685	0,0	0,0	100,0	100,0
	primários + entropias	100,0	100,0	10000,0	0	0	21.270	240.685	0,0	0,0	100,0	100,0
	primários + estatísticos	100,0	100,0	10000,0	0	0	21.270	240.685	0,0	0,0	100,0	100,0
	primários + entropias + estatísticos	100,0	100,0	10000,0	0	0	21.270	240.685	0,0	0,0	100,0	100,0
11	primários	100,0	100,0	10000,0	0	0	1.632	19.815	0,0	0,0	100,0	100,0
	primários + entropias	100,0	100,0	10000,0	0	0	1.632	19.815	0,0	0,0	100,0	100,0
	primários + estatísticos	100,0	100,0	10000,0	0	0	1.632	19.815	0,0	0,0	100,0	100,0
	primários + entropias + estatísticos	100,0	100,0	10000,0	0	0	1.632	19.815	0,0	0,0	100,0	100,0

- Resultados usando o método Bagging

  O método Bagging também atingiu um resultado significativo para o Grupo 1, conseguindo alcançar o valor máximo de acurácia (Acur), de precisão (Prec), de taxa de verdadeiros positivos (Tvp), de taxa de verdadeiros negativos (Tvn) e na F-*Measure* (F-M). E valores mínimos para as outras métricas utilizadas: quantidade de falsos positivos (FP), quantidade de falsos negativos (FN), taxa de falsos positivos (Tfp) e taxa de falsos negativos (Tfn), onde tais valores são evidenciados na Tabela 5.2.

Tabela 5.2: Resultados do grupo 1 usando o método Bagging

CEN CTU-13	**Atributos**	**Acur(%)**	**Prec(%)**	***F-M***	**FP**	**FN**	**VP**	**VN**	**Tfp(%)**	**Tfn(%)**	**Tvp(%)**	**Tvn(%)**
4	primários	100,0	100,0	10000,0	0	0	515	223.698	0,0	0,0	100,0	100,0
	primários + entropias	100,0	100,0	10000,0	0	0	515	223.698	0,0	0,0	100,0	100,0
	primários + estatísticos	100,0	100,0	10000,0	0	0	515	223.698	0,0	0,0	100,0	100,0
	primários + entropias + estatísticos	100,0	100,0	10000,0	0	0	515	223.698	0,0	0,0	100,0	100,0
10	primários	100,0	100,0	10000,0	0	0	21.270	240.685	0,0	0,0	100,0	100,0
	primários + entropias	100,0	100,0	10000,0	0	0	21.270	240.685	0,0	0,0	100,0	100,0
	primários + estatísticos	100,0	100,0	10000,0	0	0	21.270	240.685	0,0	0,0	100,0	100,0
	primários + entropias + estatísticos	100,0	100,0	10000,0	0	0	21.270	240.685	0,0	0,0	100,0	100,0
11	primários	100,0	100,0	10000,0	0	0	1.632	19.815	0,0	0,0	100,0	100,0
	primários + entropias	100,0	100,0	10000,0	0	0	1.632	19.815	0,0	0,0	100,0	100,0
	primários + estatísticos	100,0	100,0	10000,0	0	0	1.632	19.815	0,0	0,0	100,0	100,0
	primários + entropias + estatísticos	100,0	100,0	10000,0	0	0	1.632	19.815	0,0	0,0	100,0	100,0

- Resultados usando o método Adaptive Bagging

  Assim como no caso da utilização dos outros métodos (VHT e Bagging), o Adaptive Bagging também não cometeu erro no seu processo de classificação, predizendo acertadamente todos os valores das classes às quais cada fluxo pertencia, situação expressada pelos valores de acurácia (Acur), de precisão (Prec), de taxa de verdadeiros positivos (Tvp), de taxa de verdadeiros negativos (Tvn) e na F-*Measure* (F-M), quantidade de falsos positivos (FP), quantidade de falsos negativos (FN), taxa de falsos positivos (Tfp) e taxa de falsos negativos (Tfn) mostrados na Tabela 5.3.

Tabela 5.3: Resultados do grupo 1 usando o método Adaptive Bagging

CEN CTU-13	**Atributos**	**Acur(%)**	**Prec(%)**	***F-M***	**FP**	**FN**	**VP**	**VN**	**Tfp(%)**	**Tfn(%)**	**Tvp(%)**	**Tvn(%)**
4	primários	100,0	100,0	10000,0	0	0	515	223.698	0,0	0,0	100,0	100,0
	primários + entropias	100,0	100,0	10000,0	0	0	515	223.698	0,0	0,0	100,0	100,0
	primários + estatísticos	100,0	100,0	10000,0	0	0	515	223.698	0,0	0,0	100,0	100,0
	primários + entropias + estatísticos	100,0	100,0	10000,0	0	0	515	223.698	0,0	0,0	100,0	100,0
10	primários	100,0	100,0	10000,0	0	0	21.270	240.685	0,0	0,0	100,0	100,0
	primários + entropias	100,0	100,0	10000,0	0	0	21.270	240.685	0,0	0,0	100,0	100,0
	primários + estatísticos	100,0	100,0	10000,0	0	0	21.270	240.685	0,0	0,0	100,0	100,0
	primários + entropias + estatísticos	100,0	100,0	10000,0	0	0	21.270	240.685	0,0	0,0	100,0	100,0
11	primários	100,0	100,0	10000,0	0	0	1.632	19.815	0,0	0,0	100,0	100,0
	primários + entropias	100,0	100,0	10000,0	0	0	1.632	19.815	0,0	0,0	100,0	100,0
	primários + estatísticos	100,0	100,0	10000,0	0	0	1.632	19.815	0,0	0,0	100,0	100,0
	primários + entropias + estatísticos	100,0	100,0	10000,0	0	0	1.632	19.815	0,0	0,0	100,0	100,0

5.2.2 Resultados para o Grupo 2

A Tabela 5.4, a Tabela 5.5 e a Tabela 5.6 mostram os resultados que os métodos VHT, Bagging e Adaptive Bagging, separados em tópicos, respectivamente, apresentaram ao final de suas execuções. Assim como nos cenários dos Grupo 1, os métodos atualizados conseguiram maximizar os acertos e minimizar os erros, exibindo os seguintes resultados: 100% de acurácia (Acur), de precisão (Prec), de taxa de verdadeiros positivos (Tvp), de taxa de verdadeiros negativos (Tvn) e valor máximo na F-*Measure* (F-M) e 0 na quantidade de falsos positivos (FP), quantidade de falsos negativos (FN), taxa de falsos positivos (Tfp) e taxa de falsos negativos (Tfn).

Tabela 5.4: Resultados do grupo 2 usando o método VHT

CEN CTU-13	**Atributos**	**Acur(%)**	**Prec(%)**	***F-M***	**FP**	**FN**	**VP**	**VN**	**Tfp(%)**	**Tfn(%)**	**Tvp(%)**	**Tvn(%)**
4	primários	100,0	100,0	10000,0	0	0	6.466	4.231.572	0,0	0,0	100,0	100,0
	primários + entropias	100,0	100,0	10000,0	0	0	6.466	4.231.572	0,0	0,0	100,0	100,0
	primários + estatísticos	100,0	100,0	10000,0	0	0	6.466	4.231.572	0,0	0,0	100,0	100,0
	primários + entropias + estatísticos	100,0	100,0	10000,0	0	0	6.466	4.231.572	0,0	0,0	100,0	100,0
10	primários	100,0	100,0	10000,0	0	0	323.441	4.857.410	0,0	0,0	100,0	100,0
	primários + entropias	100,0	100,0	10000,0	0	0	323.441	4.857.410	0,0	0,0	100,0	100,0
	primários + estatísticos	100,0	100,0	10000,0	0	0	323.441	4.857.410	0,0	0,0	100,0	100,0
	primários + entropias + estatísticos	100,0	100,0	10000,0	0	0	323.441	4.857.410	0,0	0,0	100,0	100,0
11	primários	100,0	100,0	10000,0	0	0	277.892	130.943	0,0	0,0	100,0	100,0
	primários + entropias	100,0	100,0	10000,0	0	0	277.892	130.943	0,0	0,0	100,0	100,0
	primários + estatísticos	100,0	100,0	10000,0	0	0	277.892	130.943	0,0	0,0	100,0	100,0
	primários + entropias + estatísticos	100,0	100,0	10000,0	0	0	277.892	130.943	0,0	0,0	100,0	100,0

Tabela 5.5: Resultados do grupo 2 usando o método Bagging

CEN CTU-13	**Atributos**	**Acur(%)**	**Prec(%)**	***F-M***	**FP**	**FN**	**VP**	**VN**	**Tfp(%)**	**Tfn(%)**	**Tvp(%)**	**Tvn(%)**
4	primários	100,0	100,0	10000,0	0	0	6.466	4.231.572	0,0	0,0	100,0	100,0
	primários + entropias	100,0	100,0	10000,0	0	0	6.466	4.231.572	0,0	0,0	100,0	100,0
	primários + estatísticos	100,0	100,0	10000,0	0	0	6.466	4.231.572	0,0	0,0	100,0	100,0
	primários + entropias + estatísticos	100,0	100,0	10000,0	0	0	6.466	4.231.572	0,0	0,0	100,0	100,0
10	primários	100,0	100,0	10000,0	0	0	323.441	4.857.410	0,0	0,0	100,0	100,0
	primários + entropias	100,0	100,0	10000,0	0	0	323.441	4.857.410	0,0	0,0	100,0	100,0
	primários + estatísticos	100,0	100,0	10000,0	0	0	323.441	4.857.410	0,0	0,0	100,0	100,0
	primários + entropias + estatísticos	100,0	100,0	10000,0	0	0	323.441	4.857.410	0,0	0,0	100,0	100,0
11	primários	100,0	100,0	10000,0	0	0	277.892	130.943	0,0	0,0	100,0	100,0
	primários + entropias	100,0	100,0	10000,0	0	0	277.892	130.943	0,0	0,0	100,0	100,0
	primários + estatísticos	100,0	100,0	10000,0	0	0	277.892	130.943	0,0	0,0	100,0	100,0
	primários + entropias + estatísticos	100,0	100,0	10000,0	0	0	277.892	130.943	0,0	0,0	100,0	100,0

Tabela 5.6: Resultados do grupo 2 usando o método Adaptive Bagging

CEN CTU-13	**Atributos**	**Acur(%)**	**Prec(%)**	***F-M***	**FP**	**FN**	**VP**	**VN**	**Tfp(%)**	**Tfn(%)**	**Tvp(%)**	**Tvn(%)**
4	primários	100,0	100,0	10000,0	0	0	6.466	4.231.572	0,0	0,0	100,0	100,0
	primários + entropias	100,0	100,0	10000,0	0	0	6.466	4.231.572	0,0	0,0	100,0	100,0
	primários + estatísticos	100,0	100,0	10000,0	0	0	6.466	4.231.572	0,0	0,0	100,0	100,0
	primários + entropias + estatísticos	100,0	100,0	10000,0	0	0	6.466	4.231.572	0,0	0,0	100,0	100,0
10	primários	100,0	100,0	10000,0	0	0	323.441	4.857.410	0,0	0,0	100,0	100,0
	primários + entropias	100,0	100,0	10000,0	0	0	323.441	4.857.410	0,0	0,0	100,0	100,0
	primários + estatísticos	100,0	100,0	10000,0	0	0	323.441	4.857.410	0,0	0,0	100,0	100,0
	primários + entropias + estatísticos	100,0	100,0	10000,0	0	0	323.441	4.857.410	0,0	0,0	100,0	100,0
11	primários	100,0	100,0	10000,0	0	0	277.892	130.943	0,0	0,0	100,0	100,0
	primários + entropias	100,0	100,0	10000,0	0	0	277.892	130.943	0,0	0,0	100,0	100,0
	primários + estatísticos	100,0	100,0	10000,0	0	0	277.892	130.943	0,0	0,0	100,0	100,0
	primários + entropias + estatísticos	100,0	100,0	10000,0	0	0	277.892	130.943	0,0	0,0	100,0	100,0

5.2.3 Resultados para o Grupo 3

- Resultados usando o método VHT

  Os resultados do método VHT mostram que, pela primeira vez, o acerto máximo não foi alcançado. Verificando a Tabela 5.7, em ambos os cenários (CEN CTU-13) os valores da acurácia (Acur), da precisão (Prec), F-*Measure* (F-M) e, naturalmente, a taxa de falsos positivos (Tfp) foram os mais influenciados pelo valor da métrica falsos positivos (FP), cuja interpretação refere-se ao fato do surgimento de fluxos normais classificados erroneamente como fluxos de ataques. A maior influência observada foi na precisão, pois essa medida considera todos os casos nos quais os fluxos são classificados como positivos para ataque, sendo ou não verdadeiramente positivos, resultando em precisão (Prec) de 59,16% para o cenário CTU-13/4, de 58,83% para os cenários CTU-13/10 e o CTU-13/11. A precisão (Prec), por sua vez, interferiu na F-*Measure* (F-M), pois é dos componentes da média harmônica calculada pela mesma. No entanto, não houve casos de fluxos de ataques classificados, de forma errada, como fluxos normais - os chamados falsos negativos (FN), ocasionando em 0 para o valor da taxa de falsos negativos (Tfn). Além disso, todos os fluxos de ataques foram corretamente identificados espelhados na quantidade de verdadeiros positivos (VP), resultando numa taxa de verdadeiros positivos (Tvp) igual a 100%. Ainda como consequência da quantidade de FP, foi obtida uma taxa de verdadeiros negativos (Tvn) igual a 99,84% para o cenário CTU-13/4, 93,82% para o cenário CTU-13/10 e 94,23% para o cenário CTU-13/11. Desta tabela, também é possível visualizar que a variação da natureza dos atributos não interferiu nos resultados. Qualquer que fosse a combinação, dentro do mesmo cenário, dos atributos primários com os atributos avançados (entropia e suas diferenças e as estatísticas), não houve variação nos valores das métricas empregadas.

Tabela 5.7: Resultados do grupo 3 usando o método VHT

CEN CTU-13	**Atributos**	**Acur(%)**	**Prec(%)**	***F-M***	**FP**	**FN**	**VP**	**VN**	**Tfp(%)**	**Tfn(%)**	**Tvp(%)**	**Tvn(%)**
4	primários	99,84	59,16	7434,09	1.781	0	2.580	1.116.715	0,16	0,0	100,0	99,84
	primários + entropias	99,84	59,16	7434,09	1.781	0	2.580	1.116.715	0,16	0,0	100,0	99,84
	primários + estatísticos	99,84	59,16	7434,09	1.781	0	2.580	1.116.715	0,16	0,0	100,0	99,84
	primários + entropias + estatísticos	99,84	59,16	7434,09	1.781	0	2.580	1.116.715	0,16	0,0	100,0	99,84
10	primários	94,32	58,83	7408,06	74.421	0	106.352	1.129.018	6,18	0,0	100,0	93,82
	primários + entropias	94,32	58,83	7408,06	74.421	0	106.352	1.129.018	6,18	0,0	100,0	93,82
	primários + estatísticos	94,32	58,83	7408,06	74.421	0	106.352	1.129.018	6,18	0,0	100,0	93,82
	primários + entropias + estatísticos	94,32	58,83	7408,06	74.421	0	106.352	1.129.018	6,18	0,0	100,0	93,82
11	primários	94,67	58,83	7408,01	5.713	0	8.164	93.374	5,77	0,0	100,0	94,23
	primários + entropias	94,67	58,83	7408,01	5.713	0	8.164	93.374	5,77	0,0	100,0	94,23
	primários + estatísticos	94,67	58,83	7408,01	5.713	0	8.164	93.374	5,77	0,0	100,0	94,23
	primários + entropias + estatísticos	94,67	58,83	7408,01	5.713	0	8.164	93.374	5,77	0,0	100,0	94,23

- Resultados usando o método Bagging

  Nos resultados do Grupo 3 utilizando o método Bagging, igualmente ao que aconteceu na utilização do método VHT para este mesmo grupo, a acurácia (Acur), a precisão (Prec), F-*Measure* (F-M), a taxa de verdadeiros negativos (Tvn) não atingiram o valor máximo. Em adição, variaram conforme a alteração dos cenários (CTU-13), exceto a precisão (Prec) para os cenários 10 e 11 do CTT-13 que se mantiveram com 58,83% conforme exibe a Tabela 5.8. Mais uma vez houve o surgimento de falsos positivos (FP) com valores diferentes para cada cenário (CTU-13). Entretanto, apesar das semelhanças nos resultados obtidos, há um resultado que nenhum outro cenário, dentre os três grupos, apresentou: melhoria na acurácia (Acur), na precisão (Prec), no F-*Measure* (F-M), na taxa de verdadeiros negativos (Tvn), na quantidade de falsos positivos (FP) e na taxa de verdadeiros negativos (Tvn) de um determinado cenário (CTU-13) em função da incorporação das métricas avançadas calculadas pelo módulo holístico do sistema de detecção proposto. Com a utilização das métricas avançadas, a acurácia do cenário 11 (CTU-13) passou gradualmente de 94,67% para 100%, representando uma melhoria superior a 5%, marca esta bastante relevante, principalmente quando se considera ambientes com um número elevado de fluxos IP trafegando pela rede. O mesmo progresso pode ser observado nas outras métricas: a precisão (Prec) passou de 58,83% para 99,96%; a F-*Measure* (F-M) foi de 7408,01 para 9998,16; a taxa de verdadeiros negativos (Tvn) de 94,23% para 100%; a quantidade de falsos positivos (FP) diminuiu de 5.713 para 3; a quantidade de verdadeiros negativos (VN) sofreu um aumento de 5.710 casos e a taxa de falsos positivos (Tfp) foi de 5,77% para 0,003%.

Tabela 5.8: Resultados do grupo 3 usando o método Bagging

CEN CTU-13	**Atributos**	**Acur(%)**	**Prec(%)**	***F-M***	**FP**	**FN**	**VP**	**VN**	**Tfp(%)**	**Tfn(%)**	**Tvp(%)**	**Tvn(%)**
4	primários	99,84	59,16	7434,09	1.781	0	2.580	1.116.715	0,16	0,0	100,0	99,84
	primários + entropias	99,84	59,16	7434,09	1.781	0	2.580	1.116.715	0,16	0,0	100,0	99,84
	primários + estatísticos	99,84	59,16	7434,09	1.781	0	2.580	1.116.715	0,16	0,0	100,0	99,84
	primários + entropias + estatísticos	99,84	59,16	7434,09	1.781	0	2.580	1.116.715	0,16	0,0	100,0	99,84
10	primários	94,32	58,83	7408,06	74.421	0	106.352	1.129.018	6,18	0,0	100,0	93,82
	primários + entropias	94,32	58,83	7408,06	74.421	0	106.352	1.129.018	6,18	0,0	100,0	93,82
	primários + estatísticos	94,32	58,83	7408,06	74.421	0	106.352	1.129.018	6,18	0,0	100,0	93,82
	primários + entropias + estatísticos	94,32	58,83	7408,06	74.421	0	106.352	1.129.018	6,18	0,0	100,0	93,82
11	primários	94,67	58,83	7408,01	5.713	0	8.164	93.374	5,77	0,0	100,0	94,23
	primários + entropias	94,67	58,83	7408,01	5.713	0	8.164	93.374	5,77	0,0	100,0	94,23
	primários + estatísticos	94,67	58,83	7408,01	5.713	0	8.164	93.374	5,77	0,0	100,0	94,23
	primários + entropias + estatísticos	100,00	99,96	9998,16	3	0	8.164	99.084	0,003	0,0	100,0	100,00

- Resultados usando o método Adaptive Bagging

  A Tabela 5.9 ilustra que, diferentemente dos métodos VHT e Bagging cujos os resultados foram apresentados nos dois itens anteriores (Tabela 5.7 e Tabela 5.8), o método Adaptive Bagging obteve desempenho máximo ao atingir 100% de acurácia (Acur), de precisão (Prec), de taxa de verdadeiros positivos (Tvp), de taxa de verdadeiros negativos (Tvn) e valor máximo na F-*Measure* (F-M). Ademais, apresentou os melhores valores mínimos nas quantidade de falsos positivos (FP), quantidade de falsos negativos (FN), taxa de falsos positivos (Tfp) e taxa de falsos negativos (Tfn). Esta tabela relata, também, que tais resultados são independentes de cenários, combinação de atributos ou ambos.

Tabela 5.9: Resultados do grupo 3 usando o método Adaptive Bagging

CEN CTU-13	**Atributos**	**Acur(%)**	**Prec(%)**	***F-M***	**FP**	**FN**	**VP**	**VN**	**Tfp(%)**	**Tfn(%)**	**Tvp(%)**	**Tvn(%)**
4	primários	100,0	100,0	10000,0	0	0	2.580	1.118.496	0,0	0,0	100,0	100,0
	primários + entropias	100,0	100,0	10000,0	0	0	2.580	1.118.496	0,0	0,0	100,0	100,0
	primários + estatísticos	100,0	100,0	10000,0	0	0	2.580	1.118.496	0,0	0,0	100,0	100,0
	primários + entropias + estatísticos	100,0	100,0	10000,0	0	0	2.580	1.118.496	0,0	0,0	100,0	100,0
10	primários	100,0	100,0	10000,0	0	0	106.352	1.203.439	0,0	0,0	100,0	100,0
	primários + entropias	100,0	100,0	10000,0	0	0	106.352	1.203.439	0,0	0,0	100,0	100,0
	primários + estatísticos	100,0	100,0	10000,0	0	0	106.352	1.203.439	0,0	0,0	100,0	100,0
	primários + entropias + estatísticos	100,0	100,0	10000,0	0	0	106.352	1.203.439	0,0	0,0	100,0	100,0
11	primários	100,0	100,0	10000,0	0	0	8.164	99.087	0,0	0,0	100,0	100,0
	primários + entropias	100,0	100,0	10000,0	0	0	8.164	99.087	0,0	0,0	100,0	100,0
	primários + estatísticos	100,0	100,0	10000,0	0	0	8.164	99.087	0,0	0,0	100,0	100,0
	primários + entropias + estatísticos	100,00	100,0	10000,0	0	0	8.164	99.087	0,0	0,0	100,0	100,00

5.3 Análise dos Resultados Obtidos

Os resultados máximos obtidos em todas as métricas calculadas para os três métodos de classificação usados para identificar os fluxos normais e de ataques no Grupo 1 estão relacionados ao fato desse grupo de *datasets* utilizar parte do mesmo *dataset* para treino (80%) e a parte restante para teste (20%). Tais conjuntos de fluxos (treinamento e teste) acabam por apresentar padrões iguais de comportamento e, por vezes, conter fluxos com exatamente as mesmas características de rede, tornando-os facilmente classificáveis.

No caso do Grupo 2, apesar dos fluxos de treinamento e fluxos de teste terem origens completamente diferentes, os padrões comportamentais entre os fluxos de testes unidirecionais do *dataset* CTU-13 apresentam pouca variação. Essa situação justifica o fato da acurácia (Acur), precisão (Prec), taxa de verdadeiros positivos (Tvp), taxa de verdadeiros negativos (Tvn) terem alcançado 100% e valor máximo para a F-*Measure* (F-M), e 0 na quantidade de falsos positivos (FP), quantidade de falsos negativos (FN), taxa de falsos positivos (Tfp) e taxa de falsos negativos (Tfn).

Nesses casos onde as medidas de desempenho atingiram os maiores valores possíveis nos *datasets* utilizados, não necessariamente teremos os mesmos valores usando outros *datasets* - o que pode ser evidenciado nos resultados do Grupo 3, analisados a seguir.

No Grupo 3, onde os experimentos foram realizados com fluxos de teste bidirecionais do *dataset* CTU-13, fluxos estes com mais diversidade de informações porque existem atributos que oferecem dados mais detalhados acerca do padrão comportamental do fluxo, há uma maior variação no comportamento do tráfego de dados que eles representam. Havendo uma maior variação de comportamento, consequentemente, existirá uma maior variação nos resultados de um classificador modelado a partir de um outro conjunto de fluxos de treinamento, com origens diferentes e, provavelmente, padrões comportamentais diferentes. Tal característica desse *dataset*, somado ao fato da grande diferença (ou desbalanceamento) na quantidade de fluxos entre as duas classes (normal e ataque), impactou diretamente a métrica precisão (Prec) que se manteve entre 58,83% e 99,96%, pois em seu cálculo são considerados apenas os fluxos classificados como ataque, isto é, somente de uma classe. Diferentemente, a métrica acurácia (Acur) considera as duas classes (normal e ataque) em seu cálculo, justificando a discrepância entre os valores dessas duas medidas de desempenho. A melhoria nos resultados que os atributos avançados (entropias e

demais medidas estatísticas) proporcionaram ao método Bagging vem do fato do método em questão usar mais modelos de classificação do que o método VHT, que usa apenas um. Isso faz com que o mesmo consiga capturar com maior exatidão a contribuição, em termos de ganho de informações, de cada conjunto de atributos assinalados aos seus modelos de classificação, de modo que no passo de decisão da classificação final de um fluxo, onde existe uma votação entre os modelos, fará diferença escolher aqueles atributos com maior contribuição para a formação do modelo de classificação. Já os resultados a partir da utilização do método Adaptive Bagging, para esse grupo, podem ser explicados em função do mecanismo de funcionamento do método. Trata-se de um método projetado para fluxos que mudam suas características ou padrões ao longo do tempo. Na fase de construção do modelo, a acurácia é monitorada e, caso exista alguma pequena queda no valor da mesma, o modelo é refeito utilizando como base os últimos fluxos analisados. Desse jeito, é construído um modelo bem ajustado aos padrões comportamentais dos fluxos, resultando assim num melhor desempenho na classificação - o que foi observado na Tabela 5.9.

De uma forma geral, os resultados obtidos neste estudo demonstram que o sistema proposto apresentou um funcionamento acurado para a classificação de fluxos, superando os melhores resultados exibidos nos trabalhos [44], com um pouco mais de 95% de acurácia, e [45], com acurácia de 99,04%.

5.4 Considerações de Ordem Operacional

Como o foco do presente trabalho não é o de analisar aspectos operacionais do sistema proposto, o objetivo desta seção é tecer breves comentários acerca do desempenho no processo de classificação do sistema proposto. O tempo gasto para efetuar a classificação usando os *datasets* citados variava, em média, dois segundos conforme o método de classificação utilizado. Foi verificado, também, que o aumento ou diminuição na quantidade de atributos usados para a classificação acarretava uma respectiva variação no tempo de processamento, onde esta variação de tempo de processamento não seguia um padrão exato.

Como exemplo, podemos citar o cenário 11 do *dataset* CTU-13 com 407.242 fluxos unidirecionais, para o qual todo o processamento envolvendo as fases de treinamento e teste no ambiente descrito no Capítulo 4 levou 15 segundos com o método VHT, 17 segundos com o Bagging e 19 segundos com o Adaptive Bagging. Resultando em um tempo médio 42 microssegundos para a classificação de um fluxo com

34 atributos - característica dos fluxos integrantes dos *datasets* utilizados para este trabalho. Tal tempo para classificar um fluxo pode ser associado ao tempo que o sistema leva para identificar um fluxo cuja origem é um ataque. A diferença entre os tempos de execução se explica em virtude dos mecanismos usados pelos métodos de classificação, pois quanto maior o número de modelos para efetuar a classificação, ou os recursos para retreinar o modelo, maior será o atraso adicionado ao processo de classificação. Tais valores para o tempo de execução sugerem que, do ponto de vista da operação do sistema proposto em regime de produção num ambiente real, é possível a realização de uma etapa de treinamento extremamente rápida e capaz de gerar um sistema de detecção altamente acurado e com rápido tempo de reação do sistema. Estudos a respeito da escalabilidade do sistema em termos de processamento e tempo de treinamento, considerando-se o tamanho do *dataset* e o montante de recursos computacionais, estão além do escopo do presente trabalho.

6. Conclusão e Trabalhos Futuros

O avanço dos serviços tecnológicos providos pelas redes de computadores tem assumido um papel de bastante importância na vida contemporânea. Construindo uma relação bem próxima entre pessoas, computadores, serviços e rede de computadores, decretando o sucesso ou fracasso de alguma atividade pessoal, acadêmica, econômica ou social. Em função disso, as redes de computadores se tornaram alvo constantes de ataques. Por conseguinte, garantir a integridade do funcionamento de uma rede de computadores, atualmente, é um atividade complexa dado o alto grau de sofisticação dos ataques de negação de serviço. Entretanto, apesar da sofisticação empregada, ainda é possível detectá-los através da identificação de rastros comportamentais que seus fluxos maliciosos apresentam durante o ataque.

Dado o contexto supracitado, o presente trabalho apresentou um sistema de detecção de fluxos, oriundos de ataques distribuídos de negação de serviço (DDoS), baseado em aprendizado de máquina. A utilização de aprendizado de máquina usando classificação supervisionada permitiu que os padrões comportamentais dos fluxos lícitos e de ataques fossem capturados para a construção de um modelo de classificação de fluxos. Além disso, o sistema utilizou uma abordagem de visão global dos estados dos fluxos dentro de um intervalo de tempo, dando mais subsídios ao processo de aprendizado de máquina. Para tal, o sistema proposto foi inspirado na arquitetura Lambda [7], de maneira a oferecer um maior poder de processamento e de armazenamento, assim como aspectos de elasticidade para fins de crescimento sob demanda. O sistema proposto adota, concomitante às técnicas mais sofisticadas envolvendo aprendizado de máquina via classificação supervisionada, técnicas tradicionais de estatísticas de fluxo do tráfego agregado, normalmente usadas em detecções de ataques DDoS baseadas em limiar, como por exemplo a entropia de endereços IPs e portas e o volume de pacotes dos fluxos.

O apoio tecnológico para se chegar ao protótipo do sistema foi obtido através de

uma composição de soluções baseadas em plataformas de código aberto destinadas ao processamento de grandes volumes de fluxos de dados, tais como o Apache SAMOA, o Apache STORM e o Apache Hadoop. Além disso, foi desenvolvido um componente denominado de *módulo holístico* para incorporar informações temporais dos fluxos presentes no tráfego agregado, dentro de uma janela de tempo móvel que se estende até o momento da classificação. A integração desses elementos deu origem a uma nova implementação de processamento paralelo e escalável da arquitetura lambda.

A avaliação do sistema se deu pela utilização de *datasets* realísticos cujos os fluxos foram devidamente marcados como sendo *normal* ou *ataque*. O processo de composição de parte dos fluxos de treinamento originou-se da captura de tráfego real da rede do Observatório Nacional e da emulação de ataques a partir de informações estatísticas de casos reais de ataques registrados em publicações e *sites* especializados. Para teste do sistema proposto, foram também usados fluxos de rede obtidos para três diferentes cenários de ataques DDoS contidos no *dataset* CTU-13 [60].

Mediante os resultados apresentados no Capítulo 5, é possível verificar que o sistema proposto atingiu o valor máximo de acurácia (100%), e consequentemente valores nulos de falsos positivos e falsos negativos, para a maioria dos cenários testados. Além disso, em determinados cenários foi possível constatar uma melhoria na acurácia da classificação de ataques DDoS e de fluxos normais a partir da incorporação das métricas obtidas com o módulo holístico. Portanto, é possível concluir que o sistema proposto funciona de forma eficiente na identificação de fluxos de ataques DDoS a partir de um processo de treinamento baseado em classificação supervisionada que se apoie em um conjunto de dados obtido de cenários diferentes daqueles em que se deseja aplicar, desde que os tipos de ataques que se deseja detectar sejam os mesmos.

Como trabalhos futuros, deseja-se implementar a função de detecção de novos conjuntos de fluxos de dados no módulo holístico e implantar uma forma de "retreinamento do sistema", de maneira a contemplar ataques novos o mais rapidamente possível. A questão do tempo necessário para treinamento em função da quantidade de fluxos transmitidos ou do tamanho da rede de computadores também é uma atividade futura a ser abordada. Além disso, planeja-se aumentar a capacidade de aprendizado do sistema para a identificação de uma maior quantidade de padrões de fluxos provenientes de diversos ataques. E, também, a utilização de *datasets* diversificados para uma análise mais refinada explorando o impacto de diferentes janelas de tempo tanto para o cálculo das diferenças das entropias das características de rede, bem como, para o agrupamento dos fluxos IPs. Por fim, existe a intenção de

testar o sistema de detecção num ambiente de produção.

Referências Bibliográficas

[1] X. Yi, F. Liu, J. Liu, and H. Jin, "Building a network highway for big data: architecture and challenges," *Network, IEEE*, vol. 28, no. 4, pp. 5–13, 2014.

[2] F. Soldo and A. Metwally, "Traffic anomaly detection based on the ip size distribution," in *INFOCOM, 2012 Proceedings IEEE*, pp. 2005–2013, IEEE, 2012.

[3] P. Joshi, A. Bhandari, K. Jamunkar, K. Warghade, and P. Lokhande, "Network traffic analysis measurement and classification using hadoop," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 5, no. 3, 2016.

[4] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage, "Inferring internet denial-of-service activity," *ACM Trans. Comput. Syst.*, vol. 24, pp. 115–139, May 2006.

[5] J. Czyz, M. Kallitsis, M. Gharaibeh, C. Papadopoulos, M. Bailey, and M. Karir, "Taming the 800 pound gorilla: The rise and decline of ntp ddos attacks," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, IMC '14, (New York, NY, USA), pp. 435–448, ACM, 2014.

[6] M. Ficco and M. Rak, "Stealthy denial of service strategy in cloud computing," *Cloud Computing, IEEE Transactions on International Research Journal of Engineering and Technology (IRJET)*, vol. 3, no. 1, pp. 80–94, 2015.

[7] N. Marz and J. Warren, *Big Data: Principles and best practices of scalable realtime data systems*. Manning Publications Co., 2015.

[8] B. Li, J. Springer, G. Bebis, and M. Hadi Gunes, "A survey of network flow applications," *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 567–581, 2013.

[9] A. Bifet, G. Holmes, B. Pfahringer, and R. Gavalda, "Improving adaptive bagging methods for evolving data streams," *Asian Conference on Machine Learning*, pp. 23–37, 2009.

[10] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, pp. 15:1–15:58, July 2009.

[11] B. Wang, Y. Zheng, W. Lou, and Y. T. Hou, "Ddos attack protection in the era of cloud computing and software-defined networking," *Computer Networks*, vol. 81, pp. 308–319, 2015.

[12] A. Welzel, C. Rossow, and H. Bos, "On measuring the impact of ddos botnets," in *Proceedings of the Seventh European Workshop on System Security*, p. 3, ACM, 2014.

[13] C. Manikopoulos and S. Papavassiliou, "Network intrusion and fault detection: a statistical anomaly approach," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 76–82, 2002.

[14] F. Sebastiani, "Machine learning in automated text categorization," *ACM computing surveys (CSUR)*, vol. 34, no. 1, pp. 1–47, 2002.

[15] S. Thrun and L. Pratt, *Learning to learn*. Springer Science & Business Media, 2012.

[16] V. J. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial intelligence review*, vol. 22, no. 2, pp. 85–126, 2004.

[17] R. Brooks, S. Sitharama Iyengar, and R. Brooks, "Soft computing techniques," *Distributed sensor networks*, p. 321, 2004.

[18] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[19] L. Khan, M. Awad, and B. Thuraisingham, "A new intrusion detection system using support vector machines and hierarchical clustering," *The VLDB Journal—The International Journal on Very Large Data Bases*, vol. 16, no. 4, pp. 507–521, 2007.

[20] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[21] A. Ziviani, A. T. A. Gomes, M. L. Monsores, and P. S. Rodrigues, "Network anomaly detection using nonextensive entropy," *Communications Letters, IEEE*, vol. 11, no. 12, pp. 1034–1036, 2007.

[22] P. R. B. Guimarães, "Métodos quantitativos estatísticos," *Curitiba: IESDE Brasil SA*, 2008.

[23] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI magazine*, vol. 17, no. 3, p. 37, 1996.

[24] T. M. Mitchell, J. G. Carbonell, and R. S. Michalski, eds., *Machine Learning: A Guide to Current Research*. Norwell, MA, USA: Kluwer Academic Publishers, 1986.

[25] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 226–235, ACM, 2003.

[26] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: a review," *ACM Sigmod Record*, vol. 34, no. 2, pp. 18–26, 2005.

[27] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, *Machine learning, neural and statistical classification*. Citeseer, 1994.

[28] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[29] A. Bifet and E. Frank, "Sentiment knowledge discovery in twitter streaming data," in *International Conference on Discovery Science*, pp. 1–15, Springer, 2010.

[30] M. A. Friedl and C. E. Brodley, "Decision tree classification of land cover from remotely sensed data," *Remote sensing of environment*, vol. 61, no. 3, pp. 399–409, 1997.

[31] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[32] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[33] J. L. Gastwirth, "The estimation of the lorenz curve and gini index," *The Review of Economics and Statistics*, pp. 306–316, 1972.

[34] G. M. Menezes and G. Zaverucha, "Htilde-rt: Um algoritmo para aprenderarvores de regressao relacionais em grandes conjuntos de dados," *VIII Encontro Nacional de Inteligência Artificial*, pp. 430–442, 2011.

[35] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 71–80, ACM, 2000.

[36] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American statistical association*, vol. 58, no. 301, pp. 13–30, 1963.

[37] B. R. Prasad and S. Agarwal, "Critical parameter analysis of vertical hoeffding tree for optimized performance using samoa," *International Journal of Machine Learning and Cybernetics*, pp. 1–14, 2016.

[38] A. S. Foundation, "Vertical hoeffding tree." https://samoa.incubator.apache.org/documentation/Vertical-Hoeffding-Tree-Classifier.html. Acessado em: 15/10/2016.

[39] C. Systems, "Introduction to cisco ios netflow - a technical overview." `http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html`, 2012. Accessado em 15/10/2016.

[40] P. Barford and D. Plonka, "Characteristics of network traffic flow anomalies," in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, IMW '01, (New York, NY, USA), pp. 69–73, ACM, 2001.

[41] B. Claise, "Cisco systems netflow services export version 9," 2004.

[42] K. Singh, S. C. Guntuku, A. Thakur, and C. Hota, "Big data analytics framework for peer-to-peer botnet detection using random forests," *Information Sciences*, vol. 278, pp. 488–497, 2014.

[43] Y. Du, J. Liu, F. Liu, and L. Chen, "A real-time anomalies detection system based on streaming technology," in *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2014 Sixth International Conference on*, vol. 2, pp. 275–279, IEEE, 2014.

[44] A. Lobato, M. A. Lopez, and O. C. M. B. Duarte, "Um sistema acurado de detecção de ameaças em tempo real por processamento de fluxos," in *XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos-SBRC'2016*, (Salvador, Bahia), may 2016.

[45] R. V. Bisol, A. S. da Silva, C. C. Machado, L. Z. Granville, and A. E. Schaeffer-Filho, "Coleta e análise de características de fluxo para classificação de tráfego em redes definidas por software," in *XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos-SBRC'2016*, (Salvador, Bahia), may 2016.

[46] G. De Francisci Morales, "Samoa: A platform for mining big data streams," in *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13 Companion, (Republic and Canton of Geneva, Switzerland), pp. 777–778, International World Wide Web Conferences Steering Committee, 2013.

[47] A. Bifet and G. D. F. Morales, "Big data stream learning with samoa," in *2014 IEEE International Conference on Data Mining Workshop*, pp. 1199–1202, IEEE, 2014.

[48] A. S. Foundation, "Exemplo de topologia no samoa." https://samoa.incubator.apache.org/index.html. Acessado em: 15/10/2016.

[49] A. S. Foundation, "Componentes básicos de uma topologia no samoa." https://samoa.incubator.apache.org/documentation/SAMOA-Topology.html. Acessado em: 15/10/2016.

[50] A. T. Vu, G. De Francisci Morales, J. Gama, and A. Bifet, "Distributed adaptive model rules for mining big data streams," in *Big Data (Big Data), 2014 IEEE International Conference on*, pp. 345–353, IEEE, 2014.

[51] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pp. 81–92, VLDB Endowment, 2003.

[52] A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham, *et al.*, "Storm@ twitter," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pp. 147–156, ACM, 2014.

[53] S. G. Manikandan and S. Ravi, "Big data analysis using apache hadoop," in *IT Convergence and Security (ICITCS), 2014 International Conference on*, pp. 1–4, IEEE, 2014.

[54] I. Polato, R. Ré, A. Goldman, and F. Kon, "A comprehensive view of hadoop research—a systematic literature review," *Journal of Network and Computer Applications*, vol. 46, pp. 1–25, 2014.

[55] C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on big data," *Information Sciences*, vol. 275, pp. 314–347, 2014.

[56] O. Joldzic, Z. Djuric, and P. Vuletic, "A transparent and scalable anomaly-based dos detection method," *Computer Networks*, vol. 104, pp. 27–42, 2016.

[57] P. Li, "Selecting and using virtualization solutions: our experiences with vmware and virtualbox," *Journal of Computing Sciences in Colleges*, vol. 25, no. 3, pp. 11–17, 2010.

[58] L. Deri and N. SpA, "nprobe: an open source netflow probe for gigabit networks," in *TERENA Networking Conference*, 2003.

[59] A. V. Aho, B. W. Kernighan, and P. J. Weinberger, "Awk-a pattern scanning and processing language," *Softw., Pract. Exper.*, vol. 9, no. 4, pp. 267–279, 1979.

[60] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *computers & security*, vol. 45, pp. 100–123, 2014.

[61] R. Bussiere, "Remote port mirroring system and method thereof," Mar. 21 2000. US Patent 6,041,042.

A.Descrição dos atributos do *dataset* completo

- inicio: data e horário de início do fluxo
- dur: duração do fluxo
- prot: protocolo utilizado no fluxo
- ip_org: IP de origem do fluxo
- p_org: porta de origem do fluxo
- ip_dst: IP de destino do fluxo
- p_dst: porta de destino do fluxo
- bytes: qtd de bytes transferidos pelo fluxo
- pkts: qtd de pacotes transferidos pelo fluxo
- dif1_ent_ip_org: diferença entre o valor da entropia do IP de origem do atual intervalo de tempo e do valor de 4 intervalos de tempo anteriores
- dif2_ent_ip_org: diferença entre o valor da entropia do IP de origem do atual intervalo de tempo e do valor de 3 intervalos de tempo anteriores
- dif3_ent_ip_org: diferença entre o valor da entropia do IP de origem do atual intervalo de tempo e do valor de 2 intervalos de tempo anteriores
- dif4_ent_ip_org: diferença entre o valor da entropia do IP de origem do atual intervalo de tempo e do valor de 1 intervalo de tempo anterior
- ent_ip_org: entropia do IP de origem do atual intervalo de tempo
- dif1_ent_p_org: diferença entre o valor da entropia da porta de origem do atual intervalo de tempo e do valor de 4 intervalos de tempo anteriores

- dif2_ent_p_org: diferença entre o valor da entropia da porta de origem do atual intervalo de tempo e do valor de 3 intervalos de tempo anteriores
- dif3_ent_p_org: diferença entre o valor da entropia da porta de origem do atual intervalo de tempo e do valor de 2 intervalos de tempo anteriores
- dif4_ent_p_org: diferença entre o valor da entropia da porta de origem do atual intervalo de tempo e do valor de 1 intervalos de tempo anterior
- ent_p_org: entropia da porta de origem do atual intervalo de tempo
- dif1_ent_ip_dst: diferença entre o valor da entropia do IP de destino do atual intervalo de tempo e do valor de 4 intervalos de tempo anteriores
- dif2_ent_ip_dst: diferença entre o valor da entropia do IP de destino do atual intervalo de tempo e do valor de 3 intervalos de tempo anteriores
- dif_ent_ip_dst: diferença entre o valor da entropia do IP de destino do atual intervalo de tempo e do valor de 2 intervalos de tempo anteriores
- dif4_ent_ip_dst: diferença entre o valor da entropia do IP de destino do atual intervalo de tempo e do valor de intervalo de tempo anterior
- ent_ip_dst: entropia do IP de destino do atual intervalo de tempo
- dif1_ent_p_dst: diferença entre o valor da entropia da porta de destino do atual intervalo de tempo e do valor de 4 intervalos de tempo anteriores
- dif2_ent_p_dst: diferença entre o valor da entropia da porta de destino do atual intervalo de tempo e do valor de 3 intervalos de tempo anteriores
- dif3_ent_p_dst: diferença entre o valor da entropia da porta de destino do atual intervalo de tempo e do valor de 2 intervalos de tempo anteriores
- dif4_ent_p_dst: diferença entre o valor da entropia da porta de destino do atual intervalo de tempo e do valor de 1 intervalo de tempo anterior
- ent_p_dst: entropia da porta de destino do atual intervalo de tempo
- m_bytes: média de bytes do fluxo
- m_pkts: médias de pacotes do fluxo
- var_tam_pacotes: variação do tamanho dos pacotes do fluxo
- var_dur: variação da duração do fluxo

- classe: classe a qual o fluxo pertence: ataque ou normal.

B.Módulo Holístico: métricas avançadas

```
#!/bin/awk -f
# Script usado para calcular as metricas avancadas: entropias e suas
    diferencas, desvio-padrao da duracao do fluxo e desvio-padrao do
    tamanho dos pacotes dos fluxos.

BEGIN{
FS = ","
cont=0
qtd_exec_cont=0
}

function abs (valor){

   return (valor<0?-valor:valor);
}

{

#Atribuindo os valores das colunas aos seus respectivos arrays

          ip_origem[$4]++
        porta_origem[$5]++
        ip_destino[$6]++
        porta_destino[$7]++
        bytes_in+=$9
        pkts_in+=$8
        cont++
#Convertendo o Timestamp de início do fluxo
```

```
        split($1,ts_inicio,/-|\/|:|\.| /,seps)
        tem_inicial = mktime(ts_inicio[1]" "ts_inicio[2]" "ts_inicio[3]" "
            ts_inicio[4]" "ts_inicio[5]" "ts_inicio[6]) + (ts_inicio
            [7]/1000000)

#Pegando timestamp sem os microssegundos do primeiro fluxo
        if (FNR==1){
           timestamp_inicial = mktime(ts_inicio[1]" "ts_inicio[2]" "
              ts_inicio[3]" "ts_inicio[4]" "ts_inicio[5]" "ts_inicio[6])
           timestamp_comeco=timestamp_inicial
           timestamp_corrente = 0
        }

        # Pegando o timestamp sem os microssegundos do fluxo corrente
        if (FNR >=2){
           timestamp_corrente = mktime(ts_inicio[1]" "ts_inicio[2]" "
              ts_inicio[3]" "ts_inicio[4]" "ts_inicio[5]" "ts_inicio[6])
        }

#Convertendo o Timestamp de térrmino do fluxo
        split($2,ts_final,/-|\/|:|\.| /,seps)
        tem_final = mktime(ts_final[1]" "ts_final[2]" "ts_final[3]" "
            ts_final[4]" "ts_final[5]" "ts_final[6]) + (ts_final
            [7]/1000000)

        tempo_decorrido = (timestamp_corrente - timestamp_inicial)
        duracao_conexao[cont]=$2
        soma_duracao_conexao+=$2
        gsub("-","/",$1)
        linha[cont]=$1","$2","$3","$4","$5","$6","$7","$8","$9
        ultimocampo[cont]=$10
        tam_pkt[cont] = ($9/$8)

#Calculando a entropias (dos IPs e portas de origem e destino) a cada 30
    segundos

      if (tempo_decorrido >= 30 || FNR == max){
      qtd_exec_cont++
      timestamp_inicial = timestamp_corrente
      for (x in ip_origem) {
```

```
        entropia_ip_origem=(entropia_ip_origem + ((ip_origem[x]/(cont))*
            log((ip_origem[x]/(cont)))))
    }

          for (y in porta_origem) {

        entropia_porta_origem=(entropia_porta_origem + ((porta_origem[y]/(
            cont))*log((porta_origem[y]/(cont)))))
    }

    for (v in ip_destino) {

        entropia_ip_destino=(entropia_ip_destino + ((ip_destino[v]/(cont))
            *log((ip_destino[v]/(cont)))))
    }

    for (z in porta_destino) {

        entropia_porta_destino=(entropia_porta_destino + ((porta_destino[z
            ]/(cont))*log((porta_destino[z]/(cont)))))
    }

#Calculando a entropia
    if(cont==2 && FNR == max && timestamp_comeco != timestamp_corrente &&
         qtd_exec_cont > 1){
    entropia_normalizada_ip_origem = 0
    entropia_normalizada_porta_origem = 0
    entropia_normalizada_ip_destino = 0
    entropia_normalizada_porta_destino = 0

    }else if(cont>2){
    entropia_normalizada_ip_origem = (abs(entropia_ip_origem)/log((cont))
        )
    entropia_normalizada_porta_origem = (abs(entropia_porta_origem)/log((
        cont)))
    entropia_normalizada_ip_destino = (abs(entropia_ip_destino)/log((cont
        )))
    entropia_normalizada_porta_destino = (abs(entropia_porta_destino)/log
        ((cont)))
```

```
    }

    ord_ent_ip_orig[qtd_exec_cont] = entropia_normalizada_ip_origem
    ord_ent_port_orig[qtd_exec_cont] = entropia_normalizada_porta_origem
    ord_ent_ip_dest[qtd_exec_cont] = entropia_normalizada_ip_destino
    ord_ent_port_dest[qtd_exec_cont] = entropia_normalizada_porta_destino

#Calculando as medias dos bytes, pacotes, tamanho dos pacotes e duração da
    conexão

    media_bytes_in = (bytes_in/(cont))
    media_pkts_in = (pkts_in/(cont))
    tam_medio_pkts = (bytes_in/pkts_in)
    tam_medio_duracao_conexao = (soma_duracao_conexao/cont)

#Calculando a variância dos tamanhos dos pacotes
    for (y = 1; y <=cont; y++){
       var_tam_pacotes += ((tam_pkt[y] - tam_medio_pkts)^2)
       var_duracao_conexao += ((duracao_conexao[y]-
          tam_medio_duracao_conexao)^2)
    }

#Calculo do calculos padrao
    dv_tam_pacotes = sqrt(var_tam_pacotes)
    dv_duracao_conexao = sqrt(var_duracao_conexao)

#Escrevendo em arquivo as saidas do programa
    gsub(".csv","",FILENAME)
  for (w = 1; w <= cont; w++){
    if(qtd_exec_cont<=4){
    printf("%s,0,0,0,0,%f,0,0,0,0,%f,0,0,0,0,%f,0,0,0,0,%f,%f,%f,%f,%f,%s
        \n",linha[w],
    ord_ent_ip_orig[qtd_exec_cont],ord_ent_port_orig[qtd_exec_cont],
    ord_ent_ip_dest[qtd_exec_cont],
    ord_ent_port_dest[qtd_exec_cont],media_bytes_in,media_pkts_in,
    dv_tam_pacotes,dv_duracao_conexao,ultimocampo[w]) > FILENAME"_entdv-
        completo.csv"

    }else{
```

```
        printf("%s,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,
        %f,%f,%f,%f,%f,%f,%s\n",linha[w],ord_ent_ip_orig[qtd_exec_cont]-
            ord_ent_ip_orig[qtd_exec_cont-4],ord_ent_ip_orig[qtd_exec_cont
            ]-ord_ent_ip_orig[qtd_exec_cont-3],ord_ent_ip_orig[
            qtd_exec_cont]-ord_ent_ip_orig[qtd_exec_cont-2],ord_ent_ip_orig
            [qtd_exec_cont]-ord_ent_ip_orig[qtd_exec_cont-1],
            ord_ent_ip_orig[qtd_exec_cont],ord_ent_port_orig[qtd_exec_cont
            ]-ord_ent_port_orig[qtd_exec_cont-4],ord_ent_port_orig[
            qtd_exec_cont]-ord_ent_port_orig[qtd_exec_cont-3],
            ord_ent_port_orig[qtd_exec_cont]-ord_ent_port_orig[
            qtd_exec_cont-2],ord_ent_port_orig[qtd_exec_cont]-
            ord_ent_port_orig[qtd_exec_cont-1],ord_ent_port_orig[
            qtd_exec_cont],ord_ent_ip_dest[qtd_exec_cont]-ord_ent_ip_dest[
            qtd_exec_cont-4],ord_ent_ip_dest[qtd_exec_cont]-ord_ent_ip_dest
            [qtd_exec_cont-3],ord_ent_ip_dest[qtd_exec_cont]-
            ord_ent_ip_dest[qtd_exec_cont-2],ord_ent_ip_dest[qtd_exec_cont
            ]-ord_ent_ip_dest[qtd_exec_cont-1],ord_ent_ip_dest[
            qtd_exec_cont],ord_ent_port_dest[qtd_exec_cont]-
            ord_ent_port_dest[qtd_exec_cont-4],ord_ent_port_dest[
            qtd_exec_cont]-ord_ent_port_dest[qtd_exec_cont-3],
            ord_ent_port_dest[qtd_exec_cont]-ord_ent_port_dest[
            qtd_exec_cont-2],ord_ent_port_dest[qtd_exec_cont]-
            ord_ent_port_dest[qtd_exec_cont-1],ord_ent_port_dest[
            qtd_exec_cont],media_bytes_in,media_pkts_in,dv_tam_pacotes,
            dv_duracao_conexao,ultimocampo[w]) > FILENAME"_entdv-completo.
            csv"

        }


    }

#Apagando todos os elementos do array para não interferir na proxima
    sequencia de calculos
      delete ip_origem
      delete porta_origem
      delete ip_destino
      delete porta_destino
      delete linha
```

```
      delete ultimocampo
      delete tam_pkt
      delete duracao_conexao
      #delete tempo_decorrido

#Zerando o conteúdo das variáveis para não interferir na proxima rodada de
     calculos
      cont = 0
      entropia_normalizada_ip_origem = 0
      entropia_ip_origem = 0
      entropia_normalizada_porta_origem = 0
      entropia_porta_origem = 0
      entropia_normalizada_ip_destino = 0
      entropia_ip_destino = 0
      entropia_normalizada_porta_destino = 0
      entropia_porta_destino = 0
      media_bytes_in = 0
      media_pkts_in = 0
      bytes_in = 0
      pkts_in = 0
      tam_medio_pkts = 0
      tam_medio_duracao_conexao = 0
      soma_duracao_conexao = 0
      tempo_decorrido = 0



      }


}

          END {

          }
```

C. *Script* para Obter o ID dos Fluxos de Teste

```
#!/bin/awk -f

BEGIN{
FS = ","

}


{

     if ($34 == "normal"){
          linha_normal[FNR]=FNR

     }else if ($34 == "ataque"){
          linha_ataque[FNR]=FNR
     }

}
          END {
        printf ("Normais\n") >> FILENAME"-if-calc-met.txt"
        for (i in linha_normal){
          printf ("%s,",linha_normal[i]) >> FILENAME"-if-calc-met.txt"
        }

        printf ("\nAtaques\n") >> FILENAME"-if-calc-met.txt"
        for (j in linha_ataque){
          printf ("%s,",linha_ataque[j]) >> FILENAME"-if-calc-met.txt"
        }
```

```
        printf("\nFinalizado!\n")

          }
```

D.*Script* para Calcular As Métricas

```
 #!/bin/awk -f

# Usado para calcular a acuracia, os falsos positivos, os falsos negativos
    , a taxa de falsos positivos e a taxa de falsos negativos atraves dos
    arquivos de "IFs" e arquivo das valores preditos

BEGIN{
FS = ","
qtd_normais = 0
qtd_ataques = 0
fp=0
fn=0
}

 {
              if(FILENAME == "/home/eduardo/datasets/resultados-3grupos/
                  resultados-misturados-unidirecionais/resultados-
                  misturados-vht-uni/cen4-classific-com-entropia.log" ||
                  FILENAME == "/home/eduardo/datasets/resultados-3grupos/
                  resultados-misturados-unidirecionais/resultados-
                  misturados-vht-uni/cen4-classific-com-estat.log" ||
                  FILENAME == "/home/eduardo/datasets/resultados-3grupos/
                  resultados-misturados-unidirecionais/resultados-
                  misturados-vht-uni/cen4-classific-completo.log" ||
                  FILENAME == "/home/eduardo/datasets/resultados-3grupos/
                  resultados-misturados-unidirecionais/resultados-
                  misturados-vht-uni/cen4-classific-puro.log"){
        if(FNR==NR && NR==2){
          for (i=1;i<=NF;i++){
```

```
            linhas_normais[$i+785322]=$i+785322
            qtd_normais++
         }

      }

      if(FNR==NR && NR==4){

         for (j=1;j<=NF;j++){
            linhas_ataques[$j+785322]=$j+785322
            qtd_ataques++
         }

      }
      if (NR>FNR && NR in linhas_normais){

                  if($1 == "Instancia- Valor real: 1" && $2 == " Valor
                      predito: 0"){
                     fp++
                  }


      }
      if (NR>FNR && NR in linhas_ataque){

                  if($1 == "Instancia- Valor real: 0" && $2 == " Valor
                      predito: 1"){
                     fn++
                  }


      }
}

            if(FILENAME == "/home/eduardo/datasets/resultados-3grupos/
               resultados-misturados-unidirecionais/resultados-
               misturados-vht-uni/cen10-classific-com-entropia.log" ||
               FILENAME == "/home/eduardo/datasets/resultados-3grupos/
               resultados-misturados-unidirecionais/resultados-
               misturados-vht-uni/cen10-classific-com-estat.log" ||
```

```
          FILENAME == "/home/eduardo/datasets/resultados-3grupos/
          resultados-misturados-unidirecionais/resultados-
          misturados-vht-uni/cen10-classific-completo.log" ||
          FILENAME == "/home/eduardo/datasets/resultados-3grupos/
          resultados-misturados-unidirecionais/resultados-
          misturados-vht-uni/cen10-classific-puro.log"){
if(FNR==NR && NR==2){
   for (i=1;i<=NF;i++){
      linhas_normais[$i+916829]=$i+916829
      qtd_normais++
   }

}

if(FNR==NR && NR==4){

   for (j=1;j<=NF;j++){
      linhas_ataques[$j+916829]=$j+916829
      qtd_ataques++
   }

}
if (NR>FNR && NR in linhas_normais){

          if($1 == "Instancia- Valor real: 1" && $2 == " Valor
              predito: 0"){
             fp++
          }


}
if (NR>FNR && NR in linhas_ataque){

          if($1 == "Instancia- Valor real: 0" && $2 == " Valor
              predito: 1"){
             fn++
          }


}
```

```
}

            if(FILENAME == "/home/eduardo/datasets/resultados-3grupos/
                resultados-misturados-unidirecionais/resultados-
                misturados-vht-uni/cen11-classific-com-entropia.log" ||
                FILENAME == "/home/eduardo/datasets/resultados-3grupos/
                resultados-misturados-unidirecionais/resultados-
                misturados-vht-uni/cen11-classific-com-estat.log" ||
                FILENAME == "/home/eduardo/datasets/resultados-3grupos/
                resultados-misturados-unidirecionais/resultados-
                misturados-vht-uni/cen11-classific-completo.log" ||
                FILENAME == "/home/eduardo/datasets/resultados-3grupos/
                resultados-misturados-unidirecionais/resultados-
                misturados-vht-uni/cen11-classific-puro.log"){
      if(FNR==NR && NR==2){
        for (i=1;i<=NF;i++){
          linhas_normais[$i+75074]=$i+75074
          qtd_normais++
        }

      }

      if(FNR==NR && NR==4){

        for (j=1;j<=NF;j++){
          linhas_ataques[$j+75074]=$j+75074
          qtd_ataques++
        }

      }
      if (NR>FNR && NR in linhas_normais){

              if($1 == "Instancia- Valor real: 1" && $2 == " Valor
                  predito: 0"){
                fp++
              }


      }
      if (NR>FNR && NR in linhas_ataque){
```

```
                    if($1 == "Instancia- Valor real: 0" && $2 == " Valor
                        predito: 1"){
                       fn++
                    }


        }
   }
}

          END {
        printf (" %i, %i, %i, %i\n", qtd_normais, qtd_ataques,fp,fn)
        printf("%s: acurÃ¡cia = %.6f, tfp = %.6f, tfn = %.6f, fp = %i, fn
            = %i, qtd de inst. teste %i \n",FILENAME,((((qtd_normais+
            qtd_ataques)-(fp+fn))/(qtd_normais+qtd_ataques))*100),(fp/
            qtd_normais)*100,(fn/qtd_ataques)*100,fp,fn,qtd_normais+
            qtd_ataques) >> "/home/eduardo/datasets/resultados-3grupos/
            resultados-misturados-bidirecional-adaptivebagging.txt"
        printf("Executou!\n")

          }
```

E. *Script* Para Gerar os Quatro Cenários Usados

```
 #!/bin/awk -f
# Remove os atributos do dataset completo para gerar os quatro cenarios:
    atributos primarios+entropias, atributos primarios+estatisticos,
    atributos primarios+entropias+estatisticos e primarios.
# todos os campos
## incio,dur,prot,ip_org,p_org,ip_dst,p_dst,bytes,pkts,dif1_ent_ip_org,
    dif2_ent_ip_org,dif3_ent_ip_org,dif4_ent_ip_org,ent_ip_org,
    dif1_ent_p_org,dif2_ent_p_org,dif3_ent_p_org,dif4_ent_p_org,ent_p_org,
    dif1_ent_ip_dst,dif2_ent_ip_dst,dif_ent_ip_dst,dif4_ent_ip_dst,
    ent_ip_dst,dif1_ent_p_dst,dif2_ent_p_dst,dif3_ent_p_dst,dif4_ent_p_dst
    ,ent_p_dst,m_bytes,m_pkts,var_tam_pacotes,var_dur,classe
BEGIN{
FS = ","
}


{

if (FNR >= 1){

        if (NF == 34){
           com_estat[FNR]=$1","$2","$3","$4","$5","$6","$7","$8","$9"
               ,,,,,,,,,,,,,,,,,,,,"$30","$31","$32","$33","$34
           com_entropia[FNR]=$1","$2","$3","$4","$5","$6","$7","$8","$9","
               $10","$11","$12","$13","$14","$15","$16","$17","$18","$19","
               $20","$21","$22","$23","$24","$25","$26","$27","$28","$29"
               ,,,,,"$34
           puro[FNR]=$1","$2","$3","$4","$5","$6","$7","$8","$9"
               ,,,,,,,,,,,,,,,,,,,,,,,,,"$34
```

```
            }

        if (NF == 33){
           com_estat[FNR]=$1","$2","$3","$4","$5","$6","$7","$8","$9"
              ,,,,,,,,,,,,,,,,,,,,,","$30","$31","$32","$33","
           com_entropia[FNR]=$1","$2","$3","$4","$5","$6","$7","$8","$9","
              $10","$11","$12","$13","$14","$15","$16","$17","$18","$19","
              $20","$21","$22","$23","$24","$25","$26","$27","$28","$29"
              ,,,,,,"
           puro[FNR]=$1","$2","$3","$4","$5","$6","$7","$8","$9"
              ,,,,,,,,,,,,,,,,,,,,,,,,,,,"

        }
}
}
           END {
        gsub("completo.csv","",FILENAME)
printf ("incio,dur,prot,ip_org,p_org,ip_dst,p_dst,bytes,pkts,m_bytes,
    m_pkts,var_tam_pacotes,var_dur,classe\n") >> FILENAME"com-estat.csv"
        for (i in com_estat){
           printf ("%s\n",com_estat[i]) >> FILENAME"com-estat.csv"
        }
printf ("incio,dur,prot,ip_org,p_org,ip_dst,p_dst,bytes,pkts,
    dif1_ent_ip_org,dif2_ent_ip_org,dif3_ent_ip_org,dif4_ent_ip_org,
    ent_ip_org,dif1_ent_p_org,dif2_ent_p_org,dif3_ent_p_org,dif4_ent_p_org
    ,ent_p_org,dif1_ent_ip_dst,dif2_ent_ip_dst,dif_ent_ip_dst,
    dif4_ent_ip_dst,ent_ip_dst,dif1_ent_p_dst,dif2_ent_p_dst,
    dif3_ent_p_dst,dif4_ent_p_dst,ent_p_dst,classe\n") >> FILENAME"com-
    entropia.csv"
        for (j in com_entropia){
           printf ("%s\n",com_entropia[j]) >> FILENAME"com-entropia.csv"
        }
printf ("incio,dur,prot,ip_org,p_org,ip_dst,p_dst,bytes,pkts,classe\n") >>
     FILENAME"puro.csv"
        for (z in puro){
           printf ("%s\n",puro[z]) >> FILENAME"puro.csv"
        }

           }
```

F. *Script* Removedor de Marcações do CTU-13

Script para remover as marcacoes do dataset CTU-13 para o grupo 1

```
#!/bin/awk -f

BEGIN{
FS = ","
cont_normais_cen4 = 0
cont_ataques_cen4 = 0

cont_normais_cen10 = 0
cont_ataques_cen10 = 0

cont_normais_cen11 = 0
cont_ataques_cen11 = 0


}


{
            if (FILENAME == "/home/eduardo/datasets/novo_calc_ent-
                apenasCTU-13/cenario11-2classes-ordenado-trat_entdv.csv")
                {
        if(cont_normais_cen11 <= 79270 && $34 == "normal"){
           cont_normais_cen11++
           normais_label_cen11++
           #printf ("%s\n",$0)>>FILENAME"-sem-label"

        }else if(cont_normais_cen11 >= 79271 && $34 == "normal"){
```

```
        cont_normais_cen11++
        normais_slabel_cen11++
        #printf ("%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s
            ,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,\n",$1,$2,$3,
            $4,$5,$6,$7,$8,$9,$10,$11,$12,$13,$14,$15,$16,$17,$18,$19,
            $20,$21,$22,$23,$24,$25,$26,$27,$28,$29,$30,$31,$32,$33)>>
            FILENAME"-sem-label"
        if_normais_cen11[FNR]=FNR

        }else if(cont_ataques_cen11 <= 6532 && $34 == "ataque"){
           cont_ataques_cen11++
           ataques_label_cen11++
           #printf ("%s\n",$0)>>FILENAME"-sem-label";

           }else if(cont_ataques_cen11 >= 6533 && $34 == "ataque"){
              cont_ataques_cen11++
              ataques_slabel_cen11++
              #printf ("%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s
                  ,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,\n
                  ",$1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$13,$14,$15,
                  $16,$17,$18,$19,$20,$21,$22,$23,$24,$25,$26,$27,$28,$29
                  ,$30,$31,$32,$33)>>FILENAME"-sem-label"
              if_ataques_cen11[FNR]=FNR
     }
   }

        if (FILENAME == "/home/eduardo/datasets/novo_calc_ent-
            apenasCTU-13/cenario10-2classes-ordenado-trat_entdv.csv")
            {
     if(cont_normais_cen10 <= 962752 && $34 == "normal"){
        cont_normais_cen10++
        normais_label_cen10++
        #printf ("%s\n",$0)>>FILENAME"-sem-label"

     }else if(cont_normais_cen10 >= 962753 && $34 == "normal"){
        cont_normais_cen10++
        normais_slabel_cen10++
        #printf ("%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s
            ,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,\n",$1,$2,$3,
            $4,$5,$6,$7,$8,$9,$10,$11,$12,$13,$14,$15,$16,$17,$18,$19,
```

```
            $20,$21,$22,$23,$24,$25,$26,$27,$28,$29,$30,$31,$32,$33)>>
            FILENAME"-sem-label"
        if_normais_cen10[FNR]=FNR

        }else if(cont_ataques_cen10 <= 85082 && $34 == "ataque"){
           cont_ataques_cen10++
           ataques_label_cen10++
           #printf ("%s\n",$0)>>FILENAME"-sem-label"

           }else if(cont_ataques_cen10 >= 85083 && $34 == "ataque"){
              cont_ataques_cen10++
              ataques_slabel_cen10++
              #printf ("%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s
                  ,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,\n
                  ",$1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$13,$14,$15,
                  $16,$17,$18,$19,$20,$21,$22,$23,$24,$25,$26,$27,$28,$29
                  ,$30,$31,$32,$33)>>FILENAME"-sem-label"
              if_ataques_cen10[FNR]=FNR
    }
}

           if (FILENAME == "/home/eduardo/datasets/novo_calc_ent-
               apenasCTU-13/cenario4-2classes-ordenado-trat_entdv.csv"){
    if(cont_normais_cen4 <= 894797 && $34 == "normal"){
       cont_normais_cen4++
       normais_label_cen4++
       #printf ("%s\n",$0)>>FILENAME"-sem-label"

    }else if(cont_normais_cen4 >= 894798 && $34 == "normal"){
       cont_normais_cen4++
       normais_slabel_cen4++
       #printf ("%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s
           ,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,\n",$1,$2,$3,
           $4,$5,$6,$7,$8,$9,$10,$11,$12,$13,$14,$15,$16,$17,$18,$19,
           $20,$21,$22,$23,$24,$25,$26,$27,$28,$29,$30,$31,$32,$33)>>
           FILENAME"-sem-label"
       if_normais_cen4[FNR]=FNR

       }else if(cont_ataques_cen4 <= 2064 && $34 == "ataque"){
          cont_ataques_cen4++
```

```
            ataques_label_cen4++
            #printf ("%s\n",$0)>>FILENAME"-sem-label"

            }else if(cont_ataques_cen4 >= 2065 && $34 == "ataque"){
               cont_ataques_cen4++
               ataques_slabel_cen4++
               #printf ("%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s
                   ,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,\n
                   ",$1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$13,$14,$15,
                   $16,$17,$18,$19,$20,$21,$22,$23,$24,$25,$26,$27,$28,$29
                   ,$30,$31,$32,$33)>>FILENAME"-sem-label"
               if_ataques_cen4[FNR]=FNR
         }
      }
}

         END {
      if (FILENAME == "/home/eduardo/datasets/novo_calc_ent-apenasCTU
          -13/cenario11-2classes-ordenado-trat_entdv.csv"){

         printf("normais\n")>>FILENAME"-if"
         for (i in if_normais_cen11){

              printf("%s,",if_normais_cen11[i])>>FILENAME"-if"

         }

         printf("\nataques\n")>>FILENAME"-if"
         for (j in if_ataques_cen11){

              printf("%s,",if_ataques_cen11[j])>>FILENAME"-if"

         }

         printf ("Normais com label: %16.2f\% (%i fluxos)\n",(
             normais_label_cen11/99087)*100,normais_label_cen11)
         printf ("Normais sem label: %16.2f\% (%i fluxos)\n",(
             normais_slabel_cen11/99087)*100,normais_slabel_cen11)
         printf ("Normais: %16.2f\% (%i fluxos)\n",((normais_label_cen11
             +normais_slabel_cen11)/(FNR))*100,(normais_label_cen11+
```

```
        normais_slabel_cen11))

    printf ("Ataques com label: %16.2f\% (%i fluxos)\n",(
        ataques_label_cen11/8164)*100,ataques_label_cen11)
    printf ("Ataques sem label: %16.2f\% (%i fluxos)\n",(
        ataques_slabel_cen11/8164)*100,ataques_slabel_cen11)
      printf ("Ataques: %16.2f\% (%i fluxos)\n",((
          ataques_label_cen11+ataques_slabel_cen11)/(FNR))*100,(
          ataques_label_cen11+ataques_slabel_cen11))
    printf ("Arquivo: %s\n", FILENAME)
    printf ("Total de fluxos: %i\n", (FNR))

}else if (FILENAME == "/home/eduardo/datasets/novo_calc_ent-
    apenasCTU-13/cenario10-2classes-ordenado-trat_entdv.csv"){
    printf("normais\n")>>FILENAME"-if"
    for (i in if_normais_cen10){

         printf("%s,",if_normais_cen10[i])>>FILENAME"-if"

    }

    printf("\nataques\n")>>FILENAME"-if"
    for (j in if_ataques_cen10){

         printf("%s,",if_ataques_cen10[j])>>FILENAME"-if"

    }
    printf ("Normais com label: %16.2f\% (%i fluxos)\n",(
        normais_label_cen10/1203439)*100,normais_label_cen10)
    printf ("Normais sem label: %16.2f\% (%i fluxos)\n",(
        normais_slabel_cen10/1203439)*100,normais_slabel_cen10)
    printf ("Normais: %16.2f\% (%i fluxos)\n",((normais_label_cen10
        +normais_slabel_cen10)/FNR)*100,(normais_label_cen10+
        normais_slabel_cen10))

    printf ("Ataques com label: %16.2f\% (%i fluxos)\n",(
        ataques_label_cen10/106352)*100,ataques_label_cen10)
    printf ("Ataques sem label: %16.2f\% (%i fluxos)\n",(
        ataques_slabel_cen10/106352)*100,ataques_slabel_cen10)
      printf ("Ataques: %16.2f\% (%i fluxos)\n",((
```

```
        ataques_label_cen10+ataques_slabel_cen10)/(FNR))*100,(
        ataques_label_cen10+ataques_slabel_cen10))
  printf ("Arquivo: %s\n", FILENAME)
  printf ("Total de fluxos: %i\n",FNR)

}else if (FILENAME == "/home/eduardo/datasets/novo_calc_ent-
   apenasCTU-13/cenario4-2classes-ordenado-trat_entdv.csv"){
  printf("normais\n")>>FILENAME"-if"
  for (i in if_normais_cen4){

       printf("%s,",if_normais_cen4[i])>>FILENAME"-if"
  }

  printf("\nataques\n")>>FILENAME"-if"
  for (j in if_ataques_cen4){
       printf("%s,",if_ataques_cen4[j])>>FILENAME"-if"


  }
  printf ("Normais com label: %16.2f\% (%i fluxos)\n",(
     normais_label_cen4/1118496)*100,normais_label_cen4)
  printf ("Normais sem label: %16.2f\% (%i fluxos)\n",(
     normais_slabel_cen4/1118496)*100,normais_slabel_cen4)
  printf ("Normais: %16.2f\% (%i fluxos)\n",((normais_label_cen4+
     normais_slabel_cen4)/(FNR))*100,(normais_label_cen4+
     normais_slabel_cen4))

  printf ("Ataques com label: %16.2f\% (%i fluxos)\n",(
     ataques_label_cen4/2580)*100,ataques_label_cen4)
  printf ("Ataques sem label: %16.2f\% (%i fluxos)\n",(
     ataques_slabel_cen4/2580)*100,ataques_slabel_cen4)
    printf ("Ataques: %16.2f\% (%i fluxos)\n",((
        ataques_label_cen4+ataques_slabel_cen4)/(FNR))*100,(
        ataques_label_cen4+ataques_slabel_cen4))
  printf ("Arquivo: %s\n", FILENAME)
  printf ("Total de fluxos: %i\n", (FNR))
}


#printf("Finalizado!\n")
```

```
}
```

G.*Script* de Formatação de Fluxos Capturados

```
#!/bin/awk -f
# Script usado para cololocar os datasets ON normal e hping3 ataques no
    mesmo formato do dataset ctu-13 para estarem propicios a serem
    convertidos para arquivos .arff

BEGIN{
FS = ","
}


{

        if (FNR >= 2){
           inicio = $1
           gsub(/[-:]/," ",$1)
           t_inicio= mktime($1)
           printf ("%f\n",t_inicio)
           gsub(/[-:]/," ",$2)
           t_fim = mktime($2)
           printf ("%f\n",t_inicio)
           duracao=t_fim-t_inicio
           printf ("%f\n",duracao)
           gsub("-","/",inicio)
           linha[FNR]=$1","duracao","$3","$4","$5","$6","$7","$8","$9",
               normal"

        }
```

```
        print ("StartTime,Dur,Proto,SrcAddr,Sport,DstAddr,Dport,TotBytes,
            TotPkts,Label\n")
        for (i in linha){

          printf ("%s\n",linha[i])# >> "/home/eduardo/datasets/on/fluxos-
              para-treinamento-normal70%/"FILENAME".new"
        }


}

          END {
        printf("Executou!\n")

          }
```