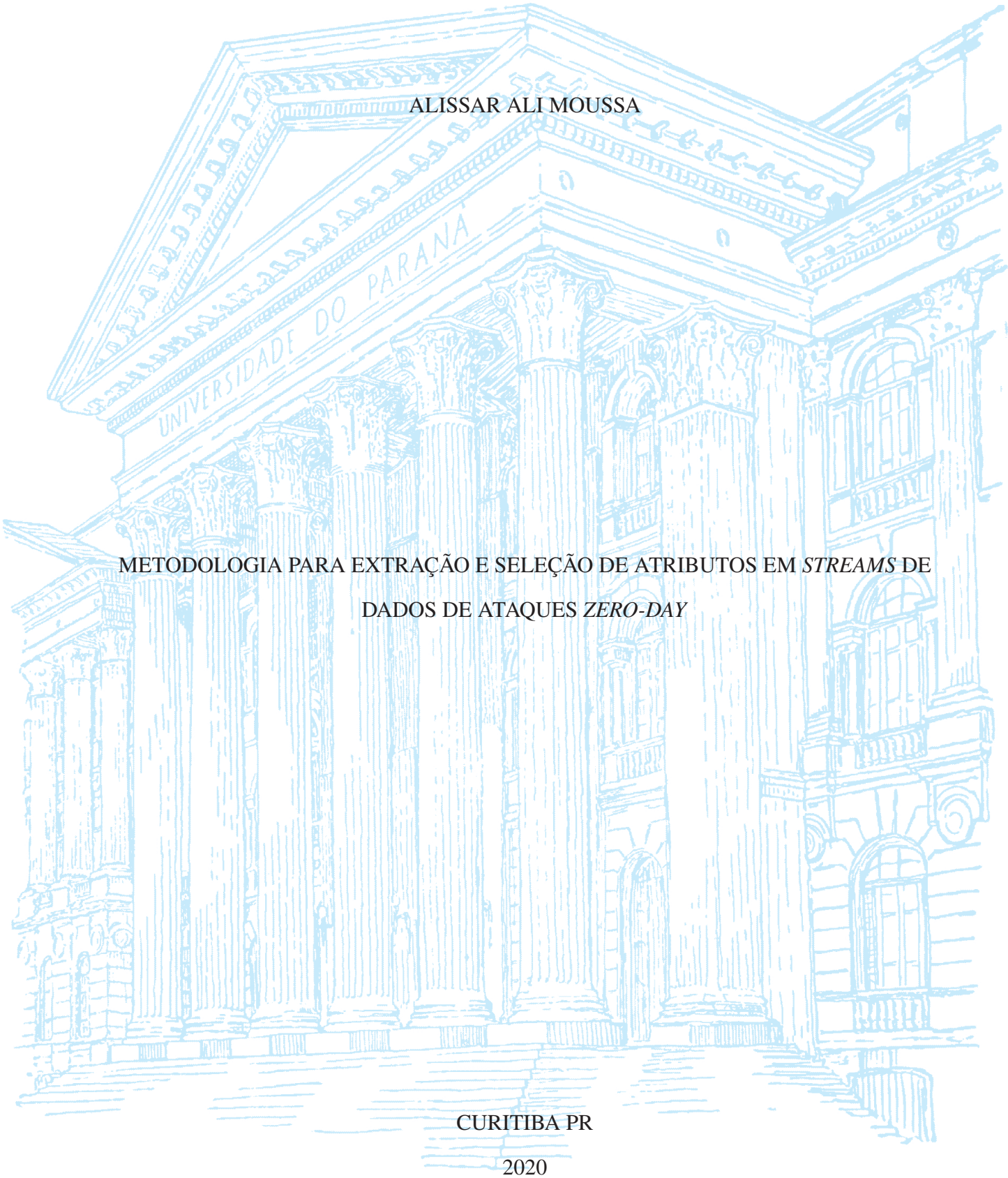
UNIVERSIDADE FEDERAL DO PARANÁ

ALISSAR ALI MOUSSA



METODOLOGIA PARA EXTRAÇÃO E SELEÇÃO DE ATRIBUTOS EM *STREAMS* DE DADOS DE ATAQUES *ZERO-DAY*

CURITIBA PR

2020

ALISSAR ALI MOUSSA

METODOLOGIA PARA EXTRAÇÃO E SELEÇÃO DE ATRIBUTOS EM *STREAMS* DE DADOS DE ATAQUES *ZERO-DAY*

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Prof. Dr. André Luiz Pires Guedes.

Coorientador: Profª. Drª. Michele Nogueira Lima.

CURITIBA PR

2020

CATALOGAÇÃO NA FONTE – SIBI/UFPR

M933m Moussa, Alissar Ali

Metodologia para extração e seleção de atributos em *streams* de dados de ataques zero-day [recurso eletrônico]/ Alissar Ali Moussa - Curitiba, 2020.

Dissertação (Mestrado) apresentada ao Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná. Área de concentração: Ciência da Computação.
Orientador: Prof. Dr. André Luiz Pires Guedes.
Coorientador: Profª. Drª. Michele Nogueira Lima.

1. Informática. 2. Zero-day. I. Guedes, André Luiz Pires. II. Lima, Michele Nogueira. III. Título. IV. Universidade Federal do Paraná.

CDD 004

Bibliotecária: Vilma Machado CRB9/1563

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da dissertação de Mestrado de **ALISSAR ALI MOUSSA** intitulada: **Extração e Seleção de Atributos em Streams de Dados de Ataques Zero-Day**, que após terem inquirido a aluna e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 24 de Setembro de 2020.

Assinatura Eletrônica
28/09/2020 14:03:40.0
ANDRÉ LUIZ PIRES GUEDES
Presidente da Banca Examinadora (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
28/09/2020 14:38:34.0
LUIZ FERNANDO BITTENCOURT
Avaliador Externo (UNIVERSIDADE ESTADUAL DE CAMPINAS)

Assinatura Eletrônica
28/09/2020 17:15:01.0
DAVID MENOTTI GOMES
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Às minhas queridas avós Tereza Maria da Conceição (in memoriam) e Fatme Idriss (in memoriam).

AGRADECIMENTOS

À minha avó Tereza, que partiu três dias antes da minha defesa. Voltar a este documento para escrever esses agradecimentos é muito difícil, mas nada comparado ao caminho que a vida a levou e que permitiu a minha vinda ao mundo. Ela nunca aprendeu a ler nem escrever. Então, eu aguardava o dia que pudesse contar a ela que finalmente me tornei mestra e dizer, dentre tantas outras coisas, algumas das palavras que estou escrevendo aqui. O destino nos tirou esse momento ou - ou apenas o redesenhou, mas confio que ela esteve presente comigo e foi testemunha. E que agora entende tudo o que está escrito e que se pode escrever (e o que não está e não se pode também). E à minha avó Fatme, a quem não pude conhecer pessoalmente mas nem por isso era menos amada, que também partiu no ano de 2020. Coisas da vida. *Maktub*.

À minha mãe e meu pai, Maria e Ali, pelo imenso apoio e suporte durante essa jornada. Minha mãe sempre fez de tudo por mim e se algo não está ao seu alcance, ela reza. Na verdade, é o que ela mais faz. Tem dado certo. Não sou religiosa, mas ela sempre enfiou na minha bolsa santinhos e terços que às vezes eu só encontrava depois de muito tempo. Todas as vezes que eu me sentia sozinha e longe de casa, eu via essas coisinhas e me sentia perto. Meu pai sempre me estimulou intelectualmente - me ensinou a ler e a escrever, compartilhou comigo desde cedo sua visão do mundo e nunca presumiu que eu não fosse capaz de entender ou discutir os assuntos que ele queria. Nunca duvidou da minha capacidade. Ele disse no meu primeiro dia de aula do ensino fundamental que minha vida começava naquele momento. Ele ainda continua dizendo isso. Gosto de pensar na vida como um conjunto inúmeros (re)começos - a culpa é dele.

Ao meu irmão Hussein, a pessoa que eu mais amo na vida, meu fiel escudeiro, que está sempre ao meu lado, e eu ao dele. Em algumas situações, inclusive, foi o único a estar. Desde junho de 2020 ele vem trabalhando como enfermeiro cuidando diretamente de pacientes com Covid-19 e a ele deixo meus cumprimentos e admiração por isso. Aliás, a todos os bravos profissionais da saúde que vêm batalhando contra essa terrível doença e com o descaso desse (des)governo. Ao meu namorado Kelvin por tanto me apoiar estar sendo lar comigo e me ajudar a aguentar firmemente essa época horrível que estamos vivendo. Te amo. Aos animaizinhos Milla, Papi, Preto, Sopa e Vanilla, meus amores, minhas vidinhas.

Às minhas amigas Alane, Ana, Marcela e Thainá - o famigerado grupo do Bebê Daltônico - que o mestrado me presenteou. Vocês me fizeram sentir acolhida, amada e pertencente a algo. Tornaram Curitiba um lugar que vou guardar na memória como extremamente amigável. Espero que eu tenha proporcionado ao menos um pouquinho do que vocês proporcionaram a mim. Eu amo vocês demais, não se esqueçam dessa amiga ausente de vocês. Aos meus amigos de longa data Danielle, Deborah, Kauê, Letícia, Ligia, Mariana e Nícolas que estão comigo desde o Fundamental e que com certeza são parte dessa minha conquista. Não há como separar vocês dessa dissertação, do meu mestrado, do meu diploma. Em mim mora ao menos um pedacinho de cada um de vocês e não sei nem como agradecer por estarem comigo há quase 20 anos.

Aos meus orientadores André e Michele pelos valiosos ensinamentos, pelas conversas, apoio, e, obviamente, pela orientação. Ao grupo TEORIA, antigo ARG, por receber essa membra extraoficial. Vocês são incríveis, professores e alunos, e me foram uma demonstração maravilhosa de cumplicidade e amizade. Ao Clube de Dança por ter me feito descobrir, após quase 25 anos, uma das coisas que eu mais gosto de fazer na vida: dançar. Vocês foram meu refúgio nos tempos difíceis e sou imensamente grata por ter conhecido vocês. À Universidade Federal do Paraná. À Capes pela concessão da bolsa que me permitiu vir a Curitiba e cursar o mestrado.

RESUMO

Os ataques *zero-day* são aqueles cujo comportamento e/ou objetivo não são conhecidos publicamente. Um ataque *zero-day* pode ser tanto um ataque completamente desconhecido, quanto um ataque conhecido, executado de maneira muito diferente do que se tem registro. Muitos ataques *zero-day* estão associados à geração de um enorme volume de dados, dificultando sua análise. Devido à falta de registros desses ataques, não é possível utilizar ferramentas fundadas em conhecimento prévio, como os Sistemas de Detecção de Intrusão (IDS) baseados em assinaturas, para detectá-los. Assim, vê-se necessário utilizar técnicas capazes de detectar comportamentos considerados diferentes ou anormais para uma rede ou sistema, a fim de os diferenciar do comportamento normal. Um dos primeiros passos para distinguir comportamentos normais dos anômalos é definir quais atributos devem ser observados, o que não é uma tarefa trivial tratando-se de ataques desconhecidos. Para realizar esse passo, uma boa alternativa é a extração e seleção de atributos. A extração e seleção de atributos geram atributos a partir dos dados de tráfego e selecionam os que são mais descritivos em relação ao estado atual da rede. Dessa forma, a identificação de partes anômalas no tráfego é facilitada e também a dimensionalidade dos dados é reduzida. Como muitos ataques estão associados a espaços de dados de alta dimensionalidade, é interessante que haja redução de dimensões. A seleção de atributos também contribui para esse objetivo, pois esta avalia os atributos e descarta aqueles que são considerados irrelevantes. A literatura carece de métodos de seleção de atributos em *stream* de dados que sejam aplicados à detecção de ataques *zero-day*. A partir do estudo aprofundado do problema, surgiu a proposta de uma metodologia que defina diretivas para facilitar e guiar o desenvolvimento de técnicas que o resolvam. Este trabalho, então, apresenta uma metodologia para extração e seleção de atributos em *streaming* de dados de tráfego de redes, com o objetivo de auxiliar na identificação de ataques *zero-day*.

Palavras-chave: Ataques *zero-day*, Seleção de atributos, Detecção de ataques

ABSTRACT

A zero-day attack is an attack in which behavior or goals are not publicly known. It may be a completely unknown attack or a known attack that is being executed differently than usual. Many zero-day attacks are associated with generating a large volume of data, which makes analysis difficult. Also, the usage of detection tools based on previous knowledge, such as signature-based Intrusion Detection Systems, is discouraged. Thus, it is preferred to use techniques and methods that detect different than usual or anomalous behaviors to distinguish them from normal behavior. One of the first steps to reach that goal is to define what features to observe, which is not a trivial task regarding zero-day attacks. A suitable alternative to accomplish this goal is feature extraction and selection. It generates features from traffic data and selects the most compliant with the current state of the network. This process facilitates the identification of anomalous traffic data and decreases data dimensionality. As many attacks are associated with high dimensionality data spaces, the detection should include a dimensionality reduction step. Feature selection also contributes to this goal because it evaluates attributes and discards the irrelevant ones. The literature lacks feature selection methods for data streams that apply to the detection of zero-day attacks. Therefore, we identified a need for a methodology to define guidelines for developing techniques that address the problem. This work presents a feature selection and extracting methodology for network traffic stream data to aid zero-day attack detection.

Keywords: Zero-day attacks, Feature selection, Attack detection

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

DoS	*Denial of Service* (Negação de Serviço)
DDoS	*Distributed Denial of Service* (Negação de Serviço Distribuído)
IDS	*Intrusion Detection System* (Sistema de Detecção de Intrusão)
IoT	*Internet of Things* (Internet das Coisas)
IP	*Internet Protocol* (Protocolo de Internet)
IPS	*Intrusion Prevention System* (Sistema de Prevenção de Intrusão)
SVM	*Support Vector Machine* (Máquina de Vetores de Suporte)

SUMÁRIO

1 INTRODUÇÃO

Com o avanço da Internet e das aplicações em rede, os atacantes estão cada vez mais sofisticados em relação à maneira de executar ataques conhecidos ou até mesmo criar novas técnicas. A aplicação de novas técnicas geram ataques desconhecidos, também chamados de ataques *zero-day*. Um ataque *zero-day* é aquele que não se tem conhecimento prévio e, em geral, os seus passos, as vulnerabilidades exploradas ou os objetivos apenas são revelados após a sua execução. Consequentemente, a identificação de ataques *zero-day* não é uma tarefa trivial, visto que grande parte das ferramentas de detecção, como os Sistemas de Detecção de Intrusão (IDS – *Intrusion Detection System*) e os Sistemas de Prevenção de Intrusão (IPS – *Intrusion Prevention System*), depende de conhecimento prévio sobre os ataques para os identificar propriamente.

Esse conhecimento prévio pode ser reunido em bases de dados que descrevem o tipo do ataque, sua identificação (também chamada de assinatura), seu funcionamento, entre outras informações. Os IDSs baseados em assinatura, por exemplo, apenas detectam intrusões e ataques que estão registrados nessas bases. Como os ataques *zero-day* não possuem registro anterior, artefatos como os IDSs falham em detectá-los prematuramente. Além disso, devido ao aprimoramento das técnicas utilizadas para gerar novos ataques, as ferramentas tradicionais de detecção somente conseguem notá-los e reportá-los quando atingem estágios avançados e o alvo muitas vezes já está comprometido, sendo tarde demais para uma ação eficiente contra esses ataques, conforme demonstrado através do modelo em (Santos et al., 2017).

Desse modo, entende-se que os ataques *zero-day* são uma grave ameaça às organizações porque são difíceis de serem detectados e prevenidos. Assim como outros tipos de ataque, um ataque *zero-day* geralmente ocorre após a exploração de vulnerabilidades presentes no seu alvo – que pode ser uma aplicação, sistema ou dispositivo. Uma vulnerabilidade é uma falha ou brecha em um sistema ou rede que pode ser manipulada com fins maliciosos, como ganhar acesso ao dispositivo e executar ataques. Pode ser chamada de *zero-day* quando tal vulnerabilidade está sendo ativamente explorada por atacantes e ainda não é conhecida pelo fabricante do *software* ou dispositivo vulnerável ou pela comunidade e, portanto, sem uma correção (um *patch*) que impeça sua exploração.

Para identificar um ataque *zero-day*, uma das técnicas que podem ser utilizadas é a detecção de anomalias. Como muitos dos ataques geram dados volumosos e pelo fato de que o comportamento de um possível ataque *zero-day* não é conhecido completamente, é interessante sujeitar os dados a um pré-processamento. O objetivo é reduzir a dimensionalidade do espaço de dados, que é crescente e potencialmente infinito, e detectar novas características associadas aos ataques. Essa tarefa, entretanto, não é simples, especialmente reconhecendo a natureza e as características dos dados que serão analisados: os dados são gerados de maneira contínua e crescente (em *streaming*), não há rótulos que identifiquem uma classe ou tipo ao qual pertencem – se são dados relativos a ataques ou a tráfego normal – e o conjunto de atributos pode não ser de tamanho fixo.

1.1 MOTIVAÇÃO

Como já mencionado, os ataques *zero-day* estão associados à identificação e exploração de vulnerabilidades *zero-day*. Esse tipo de vulnerabilidade não costuma ser tão comum, como

aponta o relatório do *Secunia Research*[1], sobre vulnerabilidades reportadas por seus clientes no ano de 2019. De 13319 vulnerabilidades, apenas 20 eram *zero-day* (Secunia Research, 2020) e entre essas, 12 ocorrências foram identificadas em sistemas operacionais. De 2014 a 2018, foram reconhecidas em média 20 vulnerabilidades *zero-day* por ano.

Ainda, de acordo com dados do *Google Project Zero*[2], um projeto mantido por analistas de segurança da Google que descobrem e estudam vulnerabilidades *zero-day* em *softwares* e *hardwares*, no primeiro semestre de 2020 foram identificadas 11 vulnerabilidades *zero-day*. A maioria delas encontradas no sistema operacional Windows e nos navegadores Mozilla Firefox, Chrome e Internet Explorer. Embora não seja um número alto de vulnerabilidades, cada uma delas pode ter gerado inúmeros ataques *zero-day* até que fossem descobertas e corrigidas. Outro fator importante é que essas aplicações e sistemas contêm um número expressivo de usuários que se convertem em potenciais alvos de ataques *zero-day*.

Um ataque *zero-day* pode ser executado por um longo período, dependendo de sua discrição e da atividade maliciosa que realiza. Um ataque torna indisponíveis serviços ou sistemas – como os ataques distribuídos de negação de serviço (DDoS - *Distributed Denial of Service*), que têm maiores chances de serem notados mais cedo que ataques cujos efeitos não são percebidos imediatamente. Então, enquanto ele não é identificado tem potencial para continuamente causar danos significativos. A preocupação é tão séria que há ações como o *Zero Day Initiative*[3], criada pela Trend Micro[4], que oferecem incentivo em dinheiro a pesquisadores e *hackers* éticos para que encontrem vulnerabilidades *zero-day* em *softwares* proprietários de empresas privadas, a fim de evitar que estas sejam exploradas primeiramente por criminosos, originando ataques. Da mesma forma, há um crescente interesse na comunidade científica em estudar e propor maneiras de detectar, prevenir e mitigar o impacto de ataques *zero-day*.

Os estudos presentes na literatura que se referem à detecção de ataques *zero-day* ou seleção de atributos em *stream* de dados pouco se referem a um algoritmo, método ou técnica que relacione as duas áreas. Geralmente, os trabalhos sobre detecção de ataques *zero-day* não se aprofundam na etapa de seleção de atributos. Inclusive, algumas soluções costumam fixar os atributos que são utilizados na análise. Os trabalhos sobre seleção de atributos que focam em *stream* de dados, por sua vez, descrevem poucas aplicações em dados de rede, especialmente para detecção de ataques *zero-day*.

Existe ainda a preocupação em reconhecer e acompanhar as mudanças que ocorrem no comportamento observado nas redes de computadores – é comum que comportamentos uma vez considerados anômalos possam ser considerados normais em um outro momento e vice-e-versa. Os atacantes estão cientes dessas mudanças e adaptam suas técnicas, criando novos ataques. Essas alterações no comportamento da rede são chamadas de mudança de conceito.

A princípio, propomos um algoritmo de seleção de atributos em *streaming* chamado α-*investing*$^+$ para ser utilizado como suporte à detecção de ataques *zero-day*. Contudo, esse algoritmo ainda não se mostrou como uma solução adequada devido a seu critério de seleção e a maneira que tratava os dados. A partir disso, surgiu a motivação para mudar o foco do trabalho e passar a estudar o problema a partir dos pontos de falha encontrados no α-*investing*$^+$ e das lacunas encontradas na literatura referentes ao tema.

[1]Equipe de pesquisa em segurança da informação da Flexera, companhia que vende diversas soluções de Tecnologia da Informação para empresas. Dentre essas soluções, há produtos de gerenciamento de vulnerabilidades em *software*. Mais em: https://www.flexera.com/.

[2]*Blog* com as atualizações da equipe: https://googleprojectzero.blogspot.com/. As vulnerabilidades *zero-day* são apontadas em uma planilha em até 90 dias após terem sido descobertas ou quando existe um *patch* que as corrijam. A planilha pode ser encontrada aqui: https://googleprojectzero.blogspot.com/p/0day.html.

[3]https://www.zerodayinitiative.com/about/.

[4]Empresa de cibersegurança líder de mercado. Mais em: https://www.trendmicro.com/.

1.2 DEFINIÇÃO DO PROBLEMA

Um ataque de rede é um conjunto de atividades maliciosas executadas para prejudicar, bisbilhotar, perturbar ou negar serviços e informações que estão inseridos nas redes de computadores. Um ataque conhecido tem registros de seu funcionamento, serviços e protocolos afetados, objetivos e muitas vezes como preveni-lo. Os ataques *zero-day*, por sua vez, não têm esses registros ou os têm parcialmente. Isso significa que mesmo um ataque conhecido pode ser considerado *zero-day* caso exista alguma característica nele que difere do que se já conhece sobre ele: seja sua forma de execução, alvo ou mudanças de objetivos.

Os ataques conhecidos são mais facilmente detectados porque as técnicas de detecção mais utilizadas se baseiam em dados históricos, aprendendo os padrões desses ataques e como eles se comportam nas redes. Entretanto, para ataques pouco conhecidos e os ataques *zero-day* não é possível basear-se nas mesmas técnicas de detecção de ataques consolidados. Nesses casos, é preciso analisar os dados de rede em busca de mudanças no comportamento que é considerado normal para a rede específica monitorada. Um dos primeiros passos desta análise é definir quais são os atributos a serem observados, sendo uma tarefa difícil tratando-se da descoberta de ataques desconhecidos.

As ferramentas mais comuns de captura de tráfego geram *logs* compostos por pacotes de rede individuais, que representam uma determinada ação na rede. Contudo, as atividades normais ou maliciosas presentes na rede raramente podem ser representadas por um único pacote, pois são compostas por um conjunto de ações. Assim, a análise por pacote não é suficiente para identificar os atributos que caracterizam um determinado estado da rede. Por isso, é necessário aplicar uma transformação nos dados originais do *log* e a partir dessa transformação extrair atributos. Dessa forma, esses atributos não serão relativos a um único pacote e sim a um conjunto deles.

Porém, nem sempre todos os atributos gerados são necessários para realizar a análise do conjunto transformado. Além disso, é possível que o número de atributos seja grande e que haja a ocorrência de novos atributos ao longo do tempo, então é preciso selecionar os atributos mais descritivos do conjunto e descartar os irrelevantes. As técnicas de seleção de atributos desempenham essa atividade e para escolher a mais adequada deve-se considerar as características dos dados e do problema. O uso de métodos tradicionais, que requerem dados rotulados, não é aplicável pois o problema requer um método que dispense a necessidade de um conhecimento prévio do espaço de dados e, principalmente, de atributos. Então, uma abordagem não-supervisionada, que lide com dados e atributos em *streaming* é a que deve ser seguida.

A seleção de atributos não-supervisionada foi se mostrando cada vez mais necessária às aplicações de mundo real. É uma característica importante desses cenários que todos os dados raramente estejam rotulados, impossibilitando a utilização de técnicas supervisionadas. Analisando a partir da perspectiva de redes de computadores, há também o fato que o volume de dados trafegados na Internet aumentou consideravelmente em relação aos registros da década de 2000 e meados da década de 2010. Com o aumento do volume do tráfego, os ataques também tornam-se mais volumosos. Assim, mais uma variável é adicionada ao problema, que é a capacidade de lidar com um volume de dados extremamente grande e que tende a aumentar.

Com apenas duas variáveis, o problema torna-se desafiador. Ao adicionar a detecção de ataques desconhecidos ao seu escopo, outra variável importante surge: a necessidade de um método que, além de não-supervisionado e preparado para lidar com um grande volume de dados, trabalhe em tempo real. Finalmente, os dados analisados são provenientes de dispositivos e aplicações de redes e exige-se que o método de seleção seja adequado a eles, preservando informações importantes presentes nas conexões entre esses dados. Então um método de seleção

de atributos utilizado para dar suporte à tarefa de detecção de ataques *zero-day* em tempo real deve considerar as quatro variáveis levantadas e ser não-supervisionado, *online* e adaptado para receber um volume grande de dados de redes. A metodologia apresentada neste trabalho direciona o desenvolvimento desse método.

1.3 OBJETIVOS

Este trabalho apresenta uma metodologia de extração e seleção de atributos em *streaming* voltada para a detecção de ataques *zero-day*. O objetivo é oferecer diretrizes para o desenvolvimento de novas técnicas e métodos de extração e seleção de atributos no contexto de detecção de ataques *zero-day*, de modo que supram as lacunas identificadas durante o estudo do problema. A metodologia é descrita em três estágios.

O primeiro estágio é de transformação de dados e extração de atributos, em que amostras do *stream* de dados são coletadas para criar janelas amostrais. A partir dessas janelas, é possível extrair diversos atributos através de métricas estatísticas aplicadas nos atributos contidos em cada janela amostral. Esses novos atributos gerados são chamados de meta-atributos. Os meta-atributos são gerados de maneira contínua e podem mudar ao longo do tempo, além do conjunto total de meta-atributos extraídos ter tamanho potencialmente infinito. Assim, cada janela terá seus meta-atributos e então define-se quais desses meta-atributos são relevantes para indicar desvios de comportamento, que podem sugerir a ocorrência de ataques.

O segundo estágio é responsável por executar a seleção de atributos no conjunto de meta-atributos extraídos no estágio anterior. Essa seleção é feita através do *ranking* de meta-atributos, que os classifica de modo que os meta-atributos que ocupam as primeiras posições são os mais significativos e, consequentemente, os últimos podem ser ditos irrelevantes. O terceiro estágio, por fim, consolida mudanças de conceito. Essa metodologia está inserida no contexto de detecção de anomalia, que é uma técnica comum para realizar a detecção de ataques *zero-day*.

1.4 ORGANIZAÇÃO DO TEXTO

O texto está organizado da seguinte forma. O Capítulo 2 introduz os conceitos fundamentais para entendimento do problema. O Capítulo 3 discute o estado da arte apresentando os trabalhos da área de detecção de ataques *zero-day* e da área seleção de atributos. O Capítulo 4 motiva e detalha a metodologia proposta, assim como seus três estágios. Finalmente, o Capítulo 5 conclui o trabalho com discussões finais acerca da metodologia e direções para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo contém as definições dos conceitos fundamentais para este trabalho. A Seção 2.1 apresenta conceitos preliminares para o entendimento do mecanismo de ataques comuns e das ferramentas que realizam a sua detecção. A Seção 2.2 discute os ataques *zero-day*, discorrendo sobre como esses são diferentes de ataques comuns e apresenta os desafios na sua identificação. A Seção 2.3 apresenta os conceitos de extração e seleção de atributos, assim como argumenta que as técnicas tradicionais podem não ser as mais adequadas quando se trata de ataques *zero-day* e dados em *stream*. A Seção 2.4 resume os pontos mais importantes do capítulo.

2.1 ATAQUES DE REDE

Os ataques de rede são definidos como "um conjunto de atividades maliciosas para perturbar, negar, degradar ou destruir informações e serviços presentes nas redes de computadores."(Ghorbani et al., 2010, p. 1, tradução nossa)[1]. Um ataque de rede tem a intenção de comprometer a integridade, confidencialidade ou disponibilidade dessas redes e pode variar bastante na forma como é executado, assim como o seu objetivo final. Os ataques podem ser executados em qualquer camada de rede.

A execução de um ataque muitas vezes inicia com a identificação e exploração de uma vulnerabilidade no *host*, sistema ou rede atingida. Uma vulnerabilidade é definida como "uma propriedade de um sistema ou de seu ambiente que, em conjunção com uma ameaça interna ou externa, pode levar a uma falha de segurança, que é uma brecha das políticas de segurança de sistema" (Anderson, 2008, p. 15, tradução nossa)[2]. As vulnerabilidades geralmente são frutos de falhas na implementação de programas ou sistemas, na configuração desses e de equipamentos de rede ou de fragilidades presentes nas políticas de segurança. Quando o atacante identifica uma vulnerabilidade no seu alvo, ele cria um *exploit*, que consiste em um código explorando uma vulnerabilidade encontrada para executar ações maliciosas. Segundo (Bhuyan et al., 2017), um atacante geralmente segue uma sequência de passos bem definidos para executar um ataque. São eles:

1. Reconhecimento: o atacante tenta coletar informações sobre a rede e o sistema para depois identificar vulnerabilidades a serem exploradas;
2. Execução do ataque: o atacante explora as vulnerabilidades encontradas;
3. Ganho de acesso: o atacante tenta ganhar acesso a um sistema como um usuário normal;
4. Ganho de acesso *root*: o atacante tenta ganhar acesso ao sistema como um superusuário;
5. Manutenção do acesso: o atacante acessa e remove os resquícios de sua ação do sistema, pois são evidências que o denunciam;
6. Inserção de *backdoors*: o ataque deixa *backdoors* no sistema que irão monitorar o sistema afetado e procurar por outras vulnerabilidades.

[1] *"Network attacks are defined as a set of malicious activities to disrupt, deny, degrade or destroy information and service resident in computer networks."*

[2] *"A vulnerability is a property of a system or its environment which, in conjunction with an internal or external threat, can lead to a security failure, which is a breach of the system's security policy."*

Em relação aos tipos de ataques, segue aqui a classificação de (Ghorbani et al., 2010), que definem 10 classes de ataques: vírus, *worms*, *trojans*, negação de serviço (DoS - *Denial of Service*, *buffer overflow*, ataques físicos, ataques de senha, coleta de informação e ataques gerais de rede. Os vírus e *worms* são programas que têm a capacidade de se auto-replicar. Enquanto um vírus necessita de um arquivo infectado para se propagar e se vincular a outros arquivos para infectar um sistema, um *worm* se propaga pela rede sem precisar de um arquivo infectado. Um *trojan*, por sua vez, é um programa – ou um trecho de programa – que é construído para realizar ações normais mas que também executa ações maliciosas através de um código diferente embutido na aplicação.

A negação de serviço (DoS - *Denial of Service*) torna indisponível serviços de um determinado alvo que é, na maioria das vezes, um servidor. O atacante envia um grande número de requisições ao servidor, causando sobrecarga na rede ou na capacidade de processamento do próprio servidor, que passa a negar acesso aos usuários legítimos (Douligeris e Mitrokotsa, 2004). Uma variação do ataque de negação de serviço é a negação de serviço distribuído (DDoS - *Distributed Denial of Service*), onde o atacante é um grupo de dispositivos. O ataque de *buffer overflow* consiste em intencionalmente estourar o limite de um *buffer* ao armazenar mais dados que o permitido dado o tamanho do *buffer*. A partir desse ataque, o atacante executa códigos maliciosos que podem prejudicar, mudar ou expor os dados daquele computador.

Um ataque físico tem como objetivo danificar componentes físicos de uma rede, como roteadores, *switches* e servidores, ou dos computadores. Um ataque de senha consiste em uma séries de ações que, se bem-sucedidas, conseguem adivinhar ou recuperar uma senha. Um ataque de coleta de informações, como o nome sugere, realiza uma varredura e coleta as vulnerabilidades presentes em um computador ou rede. Um ataque geral de rede visa comprometer uma rede ou seus usuários explorando os protocolos de rede presentes desde a camada de enlace até a camada de aplicação.

2.1.1 Detecção de ataques

O administrador de rede tem à sua disposição diversas ferramentas para detectar e mitigar diferentes tipos de ataque, como *firewalls*, sistemas anti-DDoS, anti-vírus, sistemas de detecção (IDS), sistemas de prevenção de intrusão (IPS), ferramentas de monitoramento de eventos e incidentes de segurança, entre outros. Nesta subseção, o foco está nos IDSs, as ferramentas utilizadas para identificar possíveis ataques e invasões. Esses artifícios são preparados para detectar intrusões que podem levar a ataques, analisar comportamentos suspeitos e alertar que um ataque está para acontecer, em curso ou finalizado. Uma intrusão é uma tentativa de quebrar ou mal utilizar um sistema (Ghorbani et al., 2010). Quando é detectada uma intrusão, os IDSs costumam gerar dados chamados alertas, que contêm informações como o IP de origem e destino, portas, número de pacotes e o tipo de intrusão detectada, caso o IDS tenha essa informação.

De acordo com a forma de detecção, os IDSs podem ser divididos em duas categorias: baseados em assinaturas e baseados em anomalia. Os IDSs baseados em assinaturas detectam apenas o que já conhecem. Eles analisam o fluxo de dados na rede e geram alertas a partir de um trecho do tráfego cujo comportamento é compatível com alguma intrusão ou ataque já conhecido. Esses comportamentos são identificados por conjuntos de regras chamados de assinaturas. As assinaturas são definidas após a observação de comportamentos fora do comum que depois foram confirmados como ameaça ou de comportamentos comuns que, em uma certa ordem, precedem um potencial ataque ou intrusão.

Os IDSs baseados em assinaturas necessitam de atualização constante com a definição de novos ataques vindos de seus fabricantes, que geram novas assinaturas. Mas como o número de novas vulnerabilidades e ataques cresce rapidamente, esses IDSs muitas vezes não conseguem

acompanhar as mudanças por dependerem de uma análise prévia para criar uma assinatura (Bhuyan et al., 2017). Além disso, por terem comportamentos pré-definidos, os IDSs baseados em assinaturas ainda podem não ser tão efetivos contra atacantes que conheçam quais são os programas e comportamentos considerados suspeitos por essas ferramentas. Dessa forma, o atacante irá tentar usar programas ou códigos maliciosos que ainda não são reconhecidos pelo IDS em questão, evitar comportamentos considerados ruins - descobrindo-os seja por tentativa e erro ou engenharia reversa - e utilizar técnicas que deixem o menor rastro possível no sistema (Monte, 2015).

Os IDSs baseados em anomalia buscam por desvios no comportamento normal do sistema ou rede que está sendo monitorado. Uma anomalia é uma ação isolada ou conjunto de ações que difere significativamente das ações normais do evento que está sendo observado. A detecção de anomalias busca identificar essas ações, para que atitudes sejam tomadas. Segundo (Ahmed e Mahmood, 2015), é importante considerar a natureza da anomalia para poder detectá-la propriamente. Dessa forma, as anomalias são classificadas em três categorias distintas:

- Anomalia pontual: uma instância em particular apresenta-se diferente ao padrão do conjunto de dados ao qual ela está inserida. Um exemplo dado por (Ahmed e Mahmood, 2015) é o gasto de combustível de um carro. O gasto comum de uma pessoa pode ser de 5 litros por dia, e se esse consumo é dado como 50 litros em algum dia aleatório, esse é o caso de uma anomalia pontual;

- Anomalia contextual: o comportamento de uma instância se altera em um contexto particular mas não em outros contextos. Um exemplo disso é o número de acessos a sites de compras *online* em períodos festivos ou de grandes promoções. Esse número tende a aumentar nessas épocas e isso pode ser considerado comportamento normal. Entretanto, um número alto de acessos em uma época diferente das descritas pode ser considerado como uma anomalia;

- Anomalia coletiva: várias instâncias parecidas se comportam de maneira anômala em relação ao restante do conjunto de dados. Ocorre quando apenas uma instância não é necessariamente uma anomalia, mas a presença desta em conjunto com outras pode ser uma anomalia.

A princípio, os IDSs baseados em anomalias tendem a considerar que todas as atividades intrusivas são anomalias e para isso cria-se um perfil de atividade normal para verificar se o sistema ou rede está desviando desse perfil. Assim, os IDSs baseados em anomalia são capazes de detectar ataques conhecidos e desconhecidos. Uma das desvantagens desse método é que são gerados números significativos de falsos positivos, pois atividades anômalas que não necessariamente são intrusões ou ataques podem ser marcadas como anomalias detectadas. Além disso, é preciso considerar que essa abordagem gera um alto consumo de recursos computacionais e que pode não ser rápida o suficiente para detectar ataques massivos em tempo real.

2.2 ATAQUES *ZERO-DAY*

Os ataques *zero-day* são ataques cibernéticos cujo comportamento, modo de operação ou objetivo não são conhecidos publicamente. Essa categoria abrange desde ataques nunca vistos antes até ataques mais clássicos, como o DDoS, executados de maneira muito diferente do usual. A detecção desses ataques através das ferramentas e práticas bem estabelecidas como os IDS, IPS, antivírus, *firewalls* e *patches* de atualização de sistemas e *softwares* costumam ser ineficazes

na detecção de ataques *zero-day*. Isso se dá pelo fato de que, a detecção de ataques *zero-day* é dificultada pela falta de informações pré-existentes sobre os mesmos, enquanto a detecção de ataques conhecidos baseia-se nos dados existentes e no conhecimento de seus modos de operação.

Esses ataques, assim como alguns outros tipos, estão associados à exploração de vulnerabilidades. No caso dos ataques *zero-day*, o ataque frequentemente ocorre a partir da exploração de uma vulnerabilidade que ainda não foi divulgada publicamente, ao contrário dos ataques comuns nos quais os atacantes se utilizam de uma vulnerabilidade conhecida. Se uma vulnerabilidade ainda não é conhecida, provavelmente não existe uma correção para ela e as ferramentas de detecção de intrusão também não possuem uma assinatura que descreva o comportamento associado à exploração dessa atividade que pode levar a um ataque. O ataque que se origina a partir dessa vulnerabilidade é um ataque *zero-day*, seja o ataque em si completamente novo ou se for um ataque clássico e bem conhecido como um DDoS.

Um exemplo bastante pertinente de ataque clássico que foi considerado *zero-day* pela forma que foi desempenhado é a *botnet* Mirai, identificada em 2016. O Mirai foi responsável por ataques DDoS massivos a grandes sites como Netflix, Twitter, Reddit e Github. Esse ataque funcionava basicamente infectando câmeras e roteadores e tentava descobrir credenciais administrativas de outros dispositivos da Internet das Coisas (IoT – *Internet of Things*) usando um dicionário de senhas (Kolias et al., 2017). A maior surpresa relacionada a essa ocorrência é o uso massivo de dispositivos de IoT para desempenhar o ataque, como as câmeras. Nesse caso, a vulnerabilidade encontrada era relativa à configuração dos dispositivos IoT, que usavam senhas comuns e pouco seguras.

De acordo com (Song et al., 2008), os ataques *zero-day* podem apresentar algumas características que auxiliam na sua detecção. Geralmente, esses ataques são sempre a mesma porta, porque o atacante descobriu uma vulnerabilidade específica em um sistema ou aplicação e a explora naquela porta. Ademais, os atacantes têm que desenvolver e avaliar o código várias vezes para explorar a vulnerabilidade e por isso o comportamento dos ataques *zero-day* podem ser inconsistentes e irregulares no início, pois o código que os executam precisa ser melhorado até que obtenha sucesso.

Um estudo empírico sobre ataques *zero-day* foi conduzido por (Bilge e Dumitras, 2012), com o intuito de analisar a relação entre vulnerabilidades de *softwares* e sistemas que já são conhecidas com ataques desconhecidos. Foram realizados experimentos com ataques do mundo real, observados no período de 2008 a 2011, originados a partir da exploração de 18 vulnerabilidades distintas. A partir desses experimentos, os autores listam uma série de descobertas e implicações sobre a natureza dos ataques *zero-day*. São elas:

- Os ataques *zero-day* não são tão raros como esperado, porque entre as 18 vulnerabilidades detectadas, 11 eram desconhecidas. A implicação é que os ataques *zero-day* são, de fato, uma ameaça séria que deve ser levada em conta pelas organizações devido ao impacto que podem causar;

- Os ataques *zero-day*, na época, duravam em média 10 meses, porém com alguns casos atingindo a marca de 30 meses. A duração longa da execução desses ataques indicam que esses não são detectados a tempo e podem gerar impacto de médio a longo prazo, pois as políticas e as tecnologias de segurança utilizadas pelas organizações naturalmente não estão preparadas para prever e detectar esses tipos de ataques;

- A maior parte dos ataques *zero-day* afetam poucos *hosts*, a menos que sejam ataques distribuídos e coordenados. Isso demonstra que as vulnerabilidades que levam a esses ataques são exploradas em alvos específicos;

- Após a exploração de uma vulnerabilidade desconhecida, o número de ataques relacionados a ela aumenta de 2 a 100000 vezes. Entende-se então que os atacantes e cibercriminosos estão atentos à divulgação de novas vulnerabilidades e se aproveitam do fato que nem sempre uma correção (ou uma nova assinatura) é disponibilizada imediatamente após a descoberta.

As ferramentas mais comuns de detecção de ataques, descritas na Seção 2.1, não são eficientes com os ataques desconhecidos pela falta de dados prévios. Os IDSs baseados em assinaturas, por exemplo, necessitam que o comportamento malicioso que dispara determinado alerta seja conhecido e registrado em suas bases de dados. Assim, um IDSs dessa categoria não seria capaz de detectar ataques *zero-day* – ou então somente detectaria alguns passos, mas não os ataques completos.

Os IDSs baseados em anomalia geram alertas para os comportamentos que sejam diferentes do comportamento normal estabelecido para os sistemas ou redes em que estão instalados. Ou seja, eles detectam o que é diferente do que eles conhecem. Entretanto, definir o que é um comportamento normal não é uma tarefa trivial. Dessa forma, um dos principais problemas relacionados a esse tipo de ataque é a identificação de indicadores para alertar a ocorrência de um ataque, ou sua possível ocorrência pois tratando-se de ataques *zero-day*, a utilização de métodos que trabalham com dados de ataques conhecidos não costuma ser eficiente.

2.3 EXTRAÇÃO E SELEÇÃO DE ATRIBUTOS

Os algoritmos de aprendizagem de máquina e mineração de dados, quando aplicados em dados de alta dimensionalidade, apresentam um problema crítico chamado de maldição da dimensionalidade (Li et al., 2017). Isso ocorre quando os dados tornam-se mais esparsos em um espaço de alta dimensionalidade, afetando o desempenho de algoritmos que foram desenvolvidos para lidar com espaços de menor dimensionalidade. Além disso, esses dados requerem um espaço maior de armazenamento na memória e aumentam bastante o custo computacional de análise. A redução de dimensionalidade é um conjunto de técnicas utilizadas para mitigar esse problema e a extração e seleção de atributos são exemplos dessas técnicas.

Um atributo, também chamado de *feature*, variável ou característica, é "uma propriedade mensurável individual de um processo que está sendo observado" (Chandrashekar e Sahin, 2014). Cada dimensão de um conjunto de dados é considerada um atributo. A extração de atributos consiste em gerar atributos a partir de dados que não têm um conjunto de atributos bem definido ou que não são facilmente compreensíveis. Eles também podem ser gerados a partir da combinação e de outras operações sob o conjunto original de atributos e/ou de dados. A seleção de atributos, por sua vez, visa selecionar um subconjunto de variáveis a partir de uma entrada, que é capaz de descrever de maneira eficiente os dados de entrada e eliminar variáveis irrelevantes e/ou redundantes (Guyon e Elisseeff, 2008). Tradicionalmente, costuma-se separar a seleção de atributos em três diferentes categorias: filtros, *wrappers* e *embedded*. Filtros independem de algoritmos de aprendizagem de máquina, ao contrário das técnicas *wrapper* e *embedded*.

Há diversas motivações para realizar a seleção de atributos além de selecionar as características mais relevantes, tais como a redução do espaço de dados, redução do espaço de atributos, melhoria de desempenho e melhor entendimento dos dados. A redução do espaço de dados é importante em casos que o armazenamento é limitado e os recursos computacionais disponíveis são reduzidos, pois a seleção de atributos pode ajudar a melhorar o desempenho do algoritmo. A redução do espaço de atributos ajuda a compactar a base de dados e a economizar recursos computacionais quando essa base está sendo processada. Essa economia é particularmente importante em algoritmos que processam os dados em tempo real, também

chamados de algoritmos *online*, pois quanto mais atributos e mais dados, mais lento será o retorno. No caso da detecção de ataques em tempo real, a velocidade com que os dados são processados pode determinar se um ataque será descoberto a tempo, antes que cause impactos significativos ao alvo, ou não.

(Guyon e Elisseeff, 2008) afirmam que na seleção de atributos, há três aspectos que devem ser considerados: a geração de subconjuntos de atributos, definição de um critério de avaliação e estimativa do critério de avaliação. O primeiro está associado à estratégia de busca, que pode ser baseada em diversas técnicas tais como buscas heurísticas ou estocásticas, buscas exaustivas ou *ranking* de atributos. O segundo define o critério de avaliação a ser utilizado, que pode ser obtido a partir da avaliação do desempenho de uma máquina de aprendizagem ou analisando relevância de um único atributo, do contexto dos atributos ou de subconjuntos de atributos. Uma vez que o critério de avaliação é escolhido, um método determina como esse critério será estimado, sendo que as práticas mais comuns são os testes estatísticos, validação cruzada ou limitantes de desempenho.

Sob a perspectiva dos dados, mais especificamente em relação aos seus rótulos, há diversas técnicas de seleção de atributos que processam dados rotulados e não-rotulados. Assim, existem algoritmos supervisionados para os dados rotulados, algoritmos não-supervisionados para dados não-rotulados e algoritmos semi-supervisionados para dados que estão parcialmente rotulados. Os algoritmos supervisionados utilizam métricas que consideram a informação sobre a classe dos dados, sendo que uma das métricas mais utilizadas é o ganho de informação (Liu e Motoda, 2007). Os algoritmos não-supervisionados empregam diversas técnicas para a seleção de atributos, como a clusterização. Por último, os algoritmos semi-supervisionados podem utilizar uma técnica de seleção apropriada para os dados rotulados disponíveis e uma técnica não-supervisionada para os dados que não estão rotulados.

2.3.1 Definições

Seja Y o conjunto original de atributos, com cardinalidade n. Seja d o número desejado de atributos no subconjunto selecionado $X \subseteq Y$. Seja a função de critério para seleção de atributos para o conjunto X denotada por $J(X)$. Sem perda de generalidade, considere que um valor mais alto de J indica um melhor subconjunto de atributos. Como o objetivo é maximizar $J(\cdot)$, uma função de critério possível é $(1 - p_e)$, tal que p_e denota a probabilidade de erro. Ao utilizar a probabilidade de erro como uma função de critério, a seleção de atributos torna-se dependente do classificador utilizado e do tamanho dos dados de treino e teste (Jain e Zongker, 1997). A definição 2.3.1 apresenta formalmente o problema de seleção de atributos.

Definição 2.3.1 (Seleção de atributos) *Sejam Y o conjunto original de atributos de tamanho n, d o número desejado de atributos selecionados e J a função de critério. O problema de seleção de atributos é encontrar um subconjunto $X \subseteq Y$ tal que $|X| = d$ e*

$$J(X) = \max_{Z \subseteq Y, |Z|=d} J(Z).$$

A premissa básica da **aprendizagem supervisionada** é a indução de uma hipótese que prediz com alta acurácia rótulos de instâncias novas baseando-se nas instâncias conhecidas. As instâncias são especificadas através da atribuição de valores ao conjunto de atributos. Idealmente, é necessária uma estrutura que classifique corretamente um subconjunto de dados do conjunto de treinamento, evitando o *overfitting* e utilizando somente o subconjunto de atributos que estão associados ao melhor desempenho. Entretanto, encontrar esse subconjunto é um problema difícil, por isso, é necessário o uso de buscas heurísticas para avaliar o espaço de hipóteses possíveis.

Na **aprendizagem não-supervisionada** tenta-se encontrar estruturas ou padrões escondidos nos dados (Yao et al., 2015). Não existe a predição de rótulos para as instâncias novas porque as instâncias já conhecidas também não os têm. Nesses casos, o objetivo é procurar por um padrão na distribuição dos dados e agrupar as instâncias baseado na similaridade que existe entre elas. Assim são formados os ***clusters***, agrupamentos que indicam a presença de diferentes categorias ou tipos entre as instâncias analisadas, e quando há novas instâncias, elas são comparadas aos *clusters* existentes.

Os dados de aplicações do mundo real contêm atributos que podem ser irrelevantes, redundantes ou ruidosos. Esses são os atributos que devem ser removidos em um processo de seleção, pois o custo computacional e de armazenamento é reduzido sem que se percam informações importantes e descritivas do conjunto de dados (Li et al., 2017). Se um atributo é capaz de discriminar duas classes (no caso da aprendizagem supervisionada) ou dois *clusters* (no caso da aprendizagem não-supervisionada), esse atributo é considerado **relevante** ou **fortemente relevante**. Caso haja mais um atributo no mesmo conjunto de atributos e dados com a mesma capacidade discriminante, este outro atributo é considerado **redundante** ou *fracamente relevante*. Se um atributo não é capaz de discriminar duas classes ou dois *clusters* de maneira alguma, ele é considerado **irrelevante**.

2.3.2 Classificação tradicional dos algoritmos de seleção de atributos

Os algoritmos de seleção de atributos são classificados de acordo com aspectos citados anteriormente no início desta seção, sendo eles a estratégia de busca, definição do critério de avaliação e estimativa do critério de avaliação. As três categorias mais comuns presentes na literatura são as de filtro, *wrapper* e *embedded*. Esses métodos foram originalmente projetados para abordagens supervisionadas. Os métodos de filtro e *wrapper* se diferenciam principalmente pelo critério de avaliação. Os filtros se utilizam de critérios que não necessitam da intervenção de algoritmos de aprendizagem e os *wrappers* necessitam. A principal diferença entre os *wrappers* e os *embedded* é que os *wrappers* avaliam todos os subconjuntos possíveis de atributos, enquanto os *embedded* otimizam essa tarefa e acabam sendo mais rápidos.

Os métodos de **filtro** assumem que os atributos contêm uma propriedade intrínseca e baseiam-se em métricas de avaliação de desempenho calculadas diretamente a partir dos dados, dispensando a necessidade de um preditor, classificador ou algoritmo de *clustering* para determinar quais atributos são os mais relevantes (Duch, 2008; Liu e Motoda, 2007). São geralmente mais computacionalmente eficientes que os outros, por utilizarem métricas mais simples como o *ranking* de variáveis, e por serem independentes de um algoritmo de aprendizagem. Além disso, os filtros podem ser aplicados em dados não-rotulados, diferentemente dos métodos *wrapper* e *embedded*. Contudo, o conjunto de atributos selecionado pode não ser o ótimo para o problema ao qual a seleção está sendo aplicada, devido à falta de um algoritmo de aprendizagem para guiá-la (Li et al., 2016a).

(Duch, 2008) define que um filtro de atributos é uma função que retorna um índice de relevância, denotada como $J(S|D)$. Dado um conjunto de dados D, a função estima o quão relevante um dado subconjunto de atributos S é para uma tarefa de classificação ou aproximação/regressão denotada como Y. Contudo, na maioria das vezes essa função pode ser reescrita como $J(S)$, já que os dados e a tarefa são estáticos e apenas os subconjuntos S variam. A partir de $J(S)$, é possível construir *rankings* para um determinado conjunto de atributos X, onde cada $X_i \in X, i = 1...N$, tem seu próprio *ranking* denotado como $J(X_{i_1}) \leq J(X_{i_2}) ... \leq J(X_{i_N})$. Os atributos que têm os *rankings* mais baixos são eliminados, tornando essa técnica mais adequada aos casos cujos atributos são independentes entre si. Isso não exclui a possibilidade de atributos redundantes serem selecionados, por serem considerados igualmente importantes aos demais.

Dessa forma, o *ranking* não garante que o maior subconjunto de atributos importantes seja encontrado. Os filtros podem ser construídos a partir de diversas métricas e técnicas, como a correlação, distância entre distribuições, teoria da informação e árvores de decisão.

Os métodos ***wrapper*** têm como objetivo principal selecionar o subconjunto ótimo de atributos atrelando um algoritmo de aprendizagem de máquina, definido por (Kohavi e John, 1997) como algoritmo de indução, ao processo. O algoritmo de indução é um classificador que guia a seleção de atributos, funcionando como uma função de avaliação, uma "caixa-preta", que pontua cada subconjunto de variáveis de acordo com o seu poder de predição. Um modelo baseado nesse método pode ser qualquer combinação de classificador e estratégia de busca, realizando uma busca no espaço de todos os subconjuntos de atributos possíveis, cuja solução mais trivial seria uma busca por força-bruta. Essa abordagem costuma ser computacionalmente cara e tende a ter vícios de acordo com o classificador escolhido (Aleyani et al., 2013). Entretanto, estudos empíricos demonstram que os *wrapper* são melhores, em termos de acurácia, apesar das aplicações do mundo real utilizarem os filtros pela sua rapidez, especialmente com bases de dados volumosas. Uma solução é combinar os filtros e os *wrappers* em um método híbrido.

Os métodos ***embedded*** se diferenciam dos métodos de filtro e *wrapper* por não separarem a aprendizagem e a etapa de seleção de atributos. Como exposto anteriormente, os filtros não incorporam nenhuma técnica de aprendizagem de máquina e os *wrappers* utilizam a aprendizagem para avaliar a qualidade dos subconjuntos de atributos que foram selecionados. No caso dos *embedded*, os atributos são selecionados em tempo de aprendizagem (Aleyani et al., 2013). O objetivo principal é reduzir o tempo computacional requerido pela abordagem *wrapped* para selecionar o melhor subconjunto de atributos, incorporando a seleção ao processo de treinamento (Chandrashekar e Sahin, 2014). Nessa categoria, há três subtipos principais de abordagens: seleção *forward*, que é iniciada com um atributo ou um número pequeno destes e até que o critério de parada seja atendido mais atributos são adicionados; seleção *backward*, que inicia com todos os atributos e os remove até o critério de parada; e seleção aninhada, que a cada iteração os atributos podem ser tanto adicionados quanto removidos (Lal et al., 2008). Todos esses métodos dependem do treinamento do classificador que está sendo utilizado.

2.3.3 Métodos não-supervisionados de seleção de atributos

A abordagem não-supervisionada é utilizada quando os dados disponíveis não estão rotulados, sendo impossível utilizar um método que dependa do rótulo para classificar um dado. No caso da detecção de ataques, é esperado que no início os dados não estejam rotulados. As abordagens mais comuns para seleção de atributos não-supervisionadas são os filtros, *wrappers* e híbridos (Aleyani et al., 2013; Solorio-Fernández et al., 2020).

Os métodos de filtro geram um *ranking* dos atributos de acordo com a sua capacidade discriminatória e a avaliação dos atributos pode ser univariável ou multivariável. A categoria univariável é assim chamada porque cada atributo é avaliado sozinho, enquanto na categoria multivariável os atributos são avaliados considerando os outros (Aleyani et al., 2013). Dentro da categoria univariável, há duas abordagens principais, baseada em informação ou baseada em similaridade espectral. A primeira abordagem avalia a relevância de cada tipo de acordo com técnicas fundamentadas no cálculo da entropia dos dados ou em Teoria da Informação. A segunda abordagem utiliza conceitos de análise espectral para buscar similaridades entre os dados verificados.

Os métodos *wrappers* não-supervisionados utilizam um algoritmo de *clustering*, em contrapartida dos métodos *wrappers* supervisionados que utilizam classificadores, para avaliar a qualidade do subconjunto de atributos que foi selecionado. O processo é simples: primeiro é encontrado um subconjunto de atributos e depois o algoritmo de *clustering* avalia este subconjunto.

Isso se repete até que uma determinada qualidade seja atingida, mas para isso seria necessário avaliar todos os subconjuntos possíveis, o que é bastante impraticável. Assim, são utilizadas heurísticas que podem ser categorizadas como sequenciais, onde os atributos são adicionados ou removidos sequencialmente, bio-inspiradas, que procuram adicionar aleatoriedade à busca para evitar os pontos ótimos locais, e iterativas, transformam o problema de seleção não-supervisionada de atributos em um problema de estimativa (Solorio-Fernández et al., 2020). Os métodos híbridos combinam pontos fortes dos filtros e dos *wrappers* e estes podem ser baseados em *ranking* de atributos ou não.

A identificação de atributos relevantes na abordagem não-supervisionada é uma tarefa mais difícil que na abordagem supervisionada. (Solorio-Fernández et al., 2020) realizaram uma revisão de vários métodos não-supervisionados de seleção de atributos e através desse estudo puderam identificar três critérios principais para determinar se um atributo é relevante. O primeiro é a escolha de atributos que mantenham a estrutura dos dados originais, que é bastante presente nos métodos de filtro da categoria de análise espectral univariável e multivariável. O segundo consiste em utilizar algoritmos de clusterização para buscar indicadores de *clusters* que são chamados de pseudo-rótulos para transformar a tarefa em supervisionada.

A categoria multivariável, por sua vez, contém métodos estatísticos ou baseados em informação, bio-inspirados e baseados em aprendizagem esparsa. Essa abordagem é utilizada nos filtros, *wrappers* e híbridos, também na categoria de análise espectral multivariável. O último critério é oriundo da análise de correlação de atributos, sendo que os atributos são considerados relevantes dependendo da escolha de projeto em selecionar o subconjunto com maior ou menor correlação entre as atributos. Esse critério é utilizado principalmente em filtros que se baseiam em análise estatística multivariável.

2.3.4 Seleção de atributos no contexto de detecção de ataques *zero-day*

Nas subseções anteriores, foram apresentadas diversas abordagens para realizar a seleção de atributos. Entretanto, dificilmente as técnicas tradicionais, tanto supervisionadas quanto não-supervisionadas, serão efetivas para a seleção de atributos para detecção de ataques *zero-day*, devido a características importantes inerentes ao problema. São elas:

1. **As fontes dos dados geram dados não-rotulados.** Espera-se que os dados a serem tratados sejam originados a partir de ferramentas de monitoramento, captura e/ou análise de tráfego. Os dados oriundos geralmente não são rotulados e podem apresentar formatos diferentes. As formas de captura de dados serão melhor discutidas no Capítulo 4. E como os dados não estão rotulados, naturalmente será necessário um método não-supervisionado de seleção de atributos, o que dispensa o uso de métodos supervisionados;

2. **O espaço total de atributos pode não ser conhecido previamente.** Os métodos chamados *batch*, que processam dados históricos de maneira *offline*, consideram que o conjunto completo de atributos estão disponíveis antes da análise. No caso da detecção de ataques, muitas vezes o espaço de atributos é conhecido conforme a ferramenta que gera os dados. Porém, é importante considerar os casos onde são extraídos novos atributos a partir dos existentes;

3. **O espaço total de dados não é totalmente conhecido previamente.** A seleção de atributos será aplicada na detecção de ataques, analisando dados de rede que são continuamente gerados. Um método *batch* seria capaz de somente analisar dados que já

foram coletados, o que pode atrasar ou mesmo impossibilitar a detecção de um ataque em curso;

4. **O processamento em tempo real dos dados pode ser necessário.** Para identificar um ataque prestes a acontecer ou em curso é necessário analisar os dados gerados em tempo real. Dessa forma, os métodos chamados *batch* não são adequados a essa situação. Esses seriam mais adequados às situações em que foram identificados ações ou comportamentos suspeitos na rede ou dispositivos monitorados, mas que não são referentes à um ataque em curso. Embora esses dados possam ajudar a construir o perfil normal da rede, entende-se que é necessário considerar métodos *online* de seleção de atributos que consigam lidar com dados em *streaming* para auxiliar na detecção em tempo real de ataques *zero-day*.

A partir dessas características, a abordagem mais apropriada é a seleção de atributos em *streaming*, também chamada de seleção *online* de atributos. Essa abordagem surgiu a partir da necessidade de realizar a seleção de atributos em tempo real em dados volumosos de alta dimensionalidade, como nos problemas relativos à *big data*. Segundo (Li et al., 2017), os métodos de seleção de atributos em *streaming* podem ser divididos em duas categorias: algoritmos de seleção de atributos de *streams* de atributos e algoritmos de seleção de atributos de *streams* de dados. O primeiro considera que o número de instâncias é constante e os atributos são apresentados um por vez, de maneira contínua. O segundo é exatamente o contrário: as instâncias são contínuas e o número de atributos é fixo. Um algoritmo de seleção de atributos de *streams* de atributos determina se o atributo mais recente deve ser adicionado ao conjunto de atributos selecionados, a cada iteração.

Além de ter a capacidade de analisar dados e atributos em *streaming*, uma característica desejada para um algoritmo de seleção de atributos que seja adequado à detecção de ataques *zero-day* é a habilidade de encontrar mudanças de conceito. Segundo (Gama et al., 2014), uma mudança de conceito, (ou *concept-drift*), se refere principalmente a um cenário de aprendizagem *online*, onde a relação entre os dados de entrada e as classes ou *clusters* muda ao decorrer do tempo. No caso da seleção de atributos, um algoritmo de seleção que detecta *concept drifts* consegue se adaptar às mudanças constantes que ocorrem nas distribuições de dados de redes e selecionar os atributos que são mais representativos à situação atual da rede.

2.4 RESUMO DO CAPÍTULO

Este capítulo discutiu os conceitos básicos de ataques de rede e as formas de detectá-los, em especial os sistemas de detecção de intrusão (IDS - *Intrusion Detection Systems)*. Explicou-se o funcionamento de um IDS comum e porque este pode falhar em detectar algumas formas de ataques. Essa falha é mais expressiva no caso dos IDSs baseados em assinatura, que necessitam que alguém, seja um ser humano ou uma máquina, rotule um conjunto de ações que configurem em um ataque ou intrusão. Como essa tarefa pode ser realizada somente depois da ocorrência do ataque, tratando-se de ataques desconhecidos, uma assinatura obviamente não estaria disponível no momento em que o ataque está sendo executado. Foram apresentados os ataques *zero-day*, evidenciando seus aspectos únicos, elencado suas características e as implicações, relacionando a dificuldade de detectá-los através de IDSs baseados em assinatura ou anomalias. Isso se deve à natureza imprevisível desse tipo de ataque e também à dificuldade de modelar um comportamento normal para redes que podem estar em constante mudança.

O capítulo ainda expôs os conceitos de extração e seleção de atributos, explicando o que é um atributo, o problema relacionado à alta dimensionalidade dos dados e as técnicas

tradicionais de seleção de atributos, que são supervisionados. Foram apresentadas alternativas não-supervisionadas e ainda argumentou-se que as técnicas tradicionais não são adequadas para o problema tratado neste trabalho e a seleção de atributos para *streams* de dados e/ou atributos foi apresentada como alternativa para lidar com dados e atributos que são apresentados de maneira contínua, que é o caso dos dados de redes.

3 TRABALHOS RELACIONADOS

Este capítulo discute os trabalhos relacionados à seleção de atributos e sua aplicação na detecção de ataques *zero-day*. Durante a busca por trabalhos relacionados tanto de modo mais genérico como nos Periódicos da CAPES e *Google Scholar* e em bases científicas mais especializadas como IEEE *Xplore*, ACM *Digital Library*, *ScienceDirect*, foi possível perceber que o tema de seleção de atributos em *streaming* de dados de ataques *zero-day* é pouco endereçado. Ao dividir o tema em duas partes – seleção de atributos em *streaming*/seleção *online* de atributos e detecção de ataques *zero-day*, resultados mais interessantes surgiram.

Entretanto, foi possível constatar que ambos os temas não são geralmente citados entre si. Por exemplo, em diversos trabalhos de detecção de ataques *zero-day* o problema de seleção de atributos era vagamente ou sequer citado. E em trabalhos sobre seleção *online* de atributos, os resultados não eram relacionados à detecção de ataques *zero-day* em específico – alguns utilizaram bases de dados de registros de ataque, mas sem a intenção de endereçar as suas soluções à detecção de ataques *zero-day*. Por consequência, os trabalhos das duas áreas são apresentados separadamente neste capítulo, apontando as lacunas relevantes ao problema de seleção de atributos em *streaming* como suporte à detecção de ataques *zero-day*. Assim que identificadas essas lacunas, é possível então construir uma metodologia para guiar o desenvolvimento de métodos de seleção de atributos que possam preencher esses espaços.

Esse capítulo está dividido em quatro seções. A Seção 3.1 expõe os trabalhos voltados a métodos de detecção de anomalias e ataques *zero-day* que requerem um passo de seleção de atributos ou que tratam especificamente de seleção de atributos nesse contexto. Na Seção 3.2, estão presentes os trabalhos de cunho mais teórico em seleção de atributos, independente da área de aplicação mas focando em métodos não-supervisionados. A Seção 3.3 discute e compara os trabalhos das Seções 3.1 e 3.2. Finalmente, a Seção 3.4 resume o capítulo.

3.1 TRABALHOS NA ÁREA DE DETECÇÃO DE ATAQUES *ZERO-DAY*

Os trabalhos apresentados nesta seção são referentes aos métodos de seleção de atributos e detecção de anomalias e ataques *zero-day*. (Aygun e Yavuz, 2017) propõem uma técnica de detecção de anomalia e (Zhang et al., 2017) apresentam um conjunto de métricas para avaliar a resiliência das redes em relação aos ataques *zero-day*. Finalmente, (Song et al., 2008), (Casas et al., 2012) e (Flanagan et al., 2019) tratam especificamente de detecção de ataques *zero-day*.

Sobre as avaliações de cada trabalho, (Song et al., 2008) utilizaram uma base de dados de alertas gerados pelo sistema de detecção de intrusão SNS7160[1]. Esse IDS estava inserido no perímetro da Universidade de Kyoto, monitorando seis segmentos de rede. Cada alerta continha 15 atributos, entre eles endereços IP e portas de origem e destino, horário da detecção e assinatura. (Casas et al., 2012) avaliaram sua ferramenta no repositório MAWI que é parte do projeto WIDE (Cho et al., 2000), sendo o WIDE uma rede que conecta diversas instituições no Japão. O repositório contém capturas de pacote em espaços de 15 minutos, coletados diariamente ao longo de 10 anos. Eles também avaliaram na base KDD99[2], que é uma base de dados sintética contendo simulações de ataques. (Flanagan et al., 2019) usaram a base de

[1] *Symantec Network Security 71000 Series*. Informações em: `https://www.broadcom.com/avcenter/security/Content/Product/Product_SNS.html`

[2] The KDD Cup 1999 Dataset, disponível em: `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`

dados NSL-KDD (Tavallaee et al., 2009) para avaliar o método e apresentar os resultados. A NSL-KDD é uma base sintética de *logs* contendo 41 atributos, sendo três deles categóricos e os outros 38 numéricos. Os atributos categóricos são *protocol-type*, *service* e *flag*. Alguns atributos numéricos que podem ser encontrados na base são *Num_shells*, *Src_bytes*, *Dst_bytes* e *Num_file_creations*.

(Song et al., 2008) apresentam um esquema de extração de atributos genéricos para detectar ataques *zero-day* utilizando alertas gerados por IDS. Os autores afirmam que ataques *zero-day* costumam gerar alertas em IDS porque muitos atacantes utilizam *shellcodes* que utilizam métodos já conhecidos para explorar vulnerabilidades em sistemas ou em aplicações. Dessa forma, mesmo que o ataque não seja conhecido, é possível detectá-lo a partir dos alertas que são originados por explorações cujas assinaturas estão presentes na base de dados de IDSs. Neste trabalho, são definidos sete atributos e seus valores são extraídos a partir do cálculo de métricas sob o *log* de alertas. Assim, cria-se um conjunto de dados contendo apenas as instâncias calculadas com esses sete atributos e somente este conjunto é analisado, dispensando o uso do *log* completo. Então, uma SVM (*Support Vector Machine* - Máquina de vetor de Suporte) de uma classe é utilizada para extrair ataques *zero-day* do conjunto de dados gerado.

(Casas et al., 2012) apresentam um NIDS baseado em anomalia não-supervisionado, que pretende identificar ataques conhecidos e desconhecidos sem informações prévias. O método de detecção utilizado é baseado em uma técnica chamada *Sub-Space Clustering*. Esse sistema pode ser descrito em três passos, sendo o primeiro responsável por detectar uma janela de tempo anômala para que seja analisada através de *clustering*. Nesse passo, os pacotes de rede são transformados em fluxos de tráfego e a detecção de anormalidade é realizada através de um algoritmo de mudança de conceito que basicamente verifica se o volume de tráfego mudou. O segundo passo consiste em coletar todos os fluxos marcados como anômalos para submetê-los a um processo de clusterização afim de encontrar *outliers* e construir um *ranking* de *outliers*. O último passo é marcar os *outliers* com maior pontuação no *ranking* como anomalias. A respeito da seleção de atributos, não foi utilizada nenhuma técnica não-supervisionada, que os autores julgam ser a ideal mas crítica e desafiadora. Como eles se limitaram a analisar ataques bem conhecidos, definiram nove atributos fixos para a análise. Ao avaliar o desempenho da ferramenta contra um IDS baseado em anomalia comum na base KDD99, foi constatado que ambos se equiparam. Os autores também comentam sobre o tempo computacional da ferramenta e concluem que o número baixo de atributos utilizados contribuiu na obtenção de um baixo custo.

(Aygun e Yavuz, 2017) propõem um método estocástico de detecção de anomalia baseado em modelos de *autoencoders* que visa identificar ataques *zero-day*. Os *autoenconders* são redes neurais compostas por uma camada de entrada, uma ou mais camadas escondidas e uma camada de saída. Um *autoencoder* tem duas fases: a codificação e a decodificação. Na codificação, o *autoencoder* tenta codificar a entrada com menos unidades que a camada de entrada, utilizando as camadas ocultas que têm tamanho menor que as camadas de entrada e saída. Na decodificação, o *autoencoder* reconstrói a entrada através da informação codificada armazenada nas camadas escondidas. A decodificação tem um erro de reconstrução associado. Os autores argumentam que os modelos de detecção de anomalia treinados de maneira supervisionada não são efetivos para esse tipo de ataque, pois estes ainda não são conhecidos e consequentemente não estão presentes nos modelos treinados ou em bases de conhecimento, como é o caso de sistemas de detecção baseados em assinaturas. Por isso, o método apresentado segue uma abordagem semi-supervisionada, a qual os erros de reconstrução retornados pelos *autoenconders* são empregados para criar limiares que indicam a existência de anomalia ou comportamento normal. Os resultados apresentados são satisfatórios, indicando que o método é eficaz para detectar anomalias na redes, em especial os ataques *zero-day*. Porém, nos experimentos foi

necessário converter os três atributos categóricos da base de dados em 84 valores binários, para que possam servir de entrada ao *autoenconder*. Esse processo, então, fez crescer o número de atributos do conjunto de dados, que era 41 e aumentou para 125.

(Zhang et al., 2017) abordam o conceito de diversidade de rede de computadores como um conjunto de métricas de segurança para melhorar a resiliência de redes críticas em relação a potenciais ataques *zero-day*. A primeira métrica conta todos os recursos distintos em uma rede, considerando o seu grau de distribuição e um grau variável de similaridade entre os recursos. Essa métrica é baseada nos modelos matemáticos de biodiversidade, estudados na ecologia e ela é interessante nos casos em que todos os recursos são igualmente importantes. Contudo, os autores apontam que essa métrica falha em considerar as possíveis relações causais entre os recursos. A segunda métrica é construída a partir do esforço mínimo requerido para que um ataque possa comprometer recursos importantes, dessa vez considerando as relações causais. Essa métrica falha em prover uma visão geral de como a diversidade por afetar a segurança, por mudar o foco para pontos específicos. A terceira métrica é probabilística e aponta a média de esforço necessário para atacar e comprometer serviços essenciais. As métricas são avaliadas em quatro casos de usos diferentes, que envolvem estudos de *malwares*, propagação de *worms*, ataque a um alvo específico e outra abordagem para medir a diversidade dentro de uma rede chamada *Moving Target Defense*. O objetivo com esses casos de uso é, basicamente, analisar como a diversidade de uma rede se relaciona com o quão fácil é comprometê-la e como ela se mostra resiliente a ataques *zero-day*.

(Flanagan et al., 2019) abordam o problema de detecção de novos ataques, utilizam uma rede neural proposta pelos próprios autores, chamada *Dynamic Degenerative Neural Network* (2D2N) para reconhecer mudanças no comportamento da rede. O trabalho preocupa-se em monitorar e mapear comportamentos de usuários em uma determinada rede, porque o alvo da pesquisa é detectar ataques originados de dentro da organização, os chamados *insider threats*. É proposta uma arquitetura para gerar e analisar imagens geradas a partir dos dados *online* de redes. Há três camadas na arquitetura: a primeira é responsável por clusterizar os dados em formato NetFlow, a segunda gera imagens a partir dos dados pré-clusterizados da camada anterior e a camada final contém a rede 2D2N que detecta mudanças nas imagens da camada 2. Cria-se uma nova classe de saída para a rede neural toda vez que uma anomalia é detectada, além da adição de novas camadas convolucionais para mapear essa anomalia e as mudanças causadas por ela na distribuição dos dados. Os testes foram feitas de maneira *offline*, mas os autores sugerem que com o hardware especializado é possível executar a solução de maneira *online*. Além disso, a etapa de geração de imagens é extremamente lenta - assim dita pelos autores, e isso atrasa o processo de análise em tempo real e pode ser que o ataque já esteja em curso quando um comportamento suspeito for detectado.

3.2 TRABALHOS NA ÁREA DE SELEÇÃO DE ATRIBUTOS

Nessa seção é apresentado um trabalho que trata de um método *batch* não-supervisionado (Li et al., 2016b), focado especificamente no processamento de dados de rede, outros dois trabalhos contendo métodos de seleção de atributos *online* (Huang et al., 2015; Eskandari e Javidi, 2016).

(Huang et al., 2015) apresentam um método não-supervisionado de seleção de atributos para *streams* de dados para situações em que seja preciso verificar os dados apenas uma vez e utilizar armazenagem limitada. Os autores alegam que um bom algoritmo de seleção de atributos deve ser capaz de lidar com um volume grande de dados, considerar que o *stream* de dados é potencialmente infinito e que a importância dos atributos pode mudar ao longo do tempo, devido

ao *concept drift* - atributos uma vez considerados importantes podem tornar-se irrelevantes. Então, os autores propõem uma abordagem de seleção que se adapta ao *concept drift* e que retorna um peso, chamado de pontuação de importância do atributo, a cada iteração. O método incorpora noções de *matrix sketching* para manter uma aproximação *low-rank* de todos os dados observados em cada iteração. A aproximação é explorada para gerar os pesos de cada atributo, que são utilizados como critério de seleção. Os experimentos foram executados sob bases de dados de imagens e de textos, mostrando que o método é eficaz para detectar o *concept drift* ao longo de tempo e que não sofre de sobrecarga de memória quando submetido a bases de dados de tamanho na ordem de 10^6, além de desempenho satisfatório na seleção dos atributos.

(Eskandari e Javidi, 2016) tratam de seleção de atributos em *streaming* através de um método *online* de seleção baseado em conjuntos aproximados. Os conjuntos aproximados expressam incerteza e imprecisão na região limítrofe de um conjunto. O método explora essa região em busca de informações úteis que possam ser utilizadas para diferenciar os atributos, pois normalmente apenas a região positiva do conjunto é considerada em outras técnicas de seleção baseadas em conjuntos aproximados. A partir das informações contidas nos conjuntos, são calculadas métricas de dependência entre os atributos e também é realizada uma análise de significância a cada vez que um novo atributo é apresentado ao algoritmo que implementa o método. Os resultados experimentais foram obtidos a partir de bases categóricas e numéricas e o número de atributos selecionados é consideravelmente menor se comparado a outras técnicas de seleção de atributos em *streaming* pré-existentes.

(Li et al., 2016b) apresentam um método de seleção de atributos não-supervisionado para dados de tráfego de redes. A necessidade de um método que se aplica a um tipo de dado específico dá-se ao fato que os dados de tráfego são inerentemente conectados entre si e que a obtenção de rótulos demanda tempo e muito trabalho. O método extrai conexões inerentes através de uma representação latente, para descobrir atributos escondidos codificados na estrutura da rede. Então, para reduzir o número de conexões ruidosas, o modelo de aprendizagem da representação é atrelado à seleção de atributos e assim ambas as partes podem cooperar. Os testes foram executados em bases com dados de aplicações em redes, como redes sociais e e de sites de avaliação de produtos, obtendo baixa acurácia em todos os casos, porém esses resultados eram ainda melhores que nos testes onde foram aplicados outros algoritmos que não consideram a natureza dos dados de rede. Portanto, esses resultados sugerem que ainda há espaço para melhorias no método.

3.3 DISCUSSÃO

Ao analisar os trabalhos anteriormente apresentados, percebe-se que todos mostram pontos relevantes para a construção de um método adequado para seleção de atributos em *streaming* utilizados na detecção de ataques *zero-day*. Destacam-se os trabalhos mais recentes da área de seleção de atributos, que nos últimos anos vêm tentando atender à demanda de métodos e técnicas que sejam capazes de lidar com dados em *streaming*, pois são diversas as aplicações que geram esse tipo de dado. Os trabalhos dessa categoria, entretanto, necessitam ser adequados ao contexto do problema de detecção de ataques *zero-day*.

Os trabalhos da área de detecção de ataques *zero-day* ainda apresentam soluções baseadas em conceitos mais antigos, que requerem conhecimento do espaço de atributos e/ou de dados, ou então não apresentam técnica alguma de seleção de atributos. O uso de atributos fixos (Song et al., 2008; Casas et al., 2012) e a prática de extração de atributos sem uma etapa de seleção seguinte (Song et al., 2008; Flanagan et al., 2019; Aygun e Yavuz, 2017) demonstram que essa problemática não está sendo endereçada com frequência. O trabalho que se diferencia

nessa categoria é o de (Zhang et al., 2017), que apresenta métricas de segurança baseadas no conceito de diversidade da rede. Embora não componham um método de detecção em si, as métricas geradas podem ser utilizadas como atributos a serem considerados na análise de dados.

(Song et al., 2008) utilizaram alertas gerados por IDS em seu trabalho para a detecção de ataques *zero-day*, alegando que esse tipo de ataque pode disparar alertas em IDS ao identificar comportamentos conhecidos e traduzidos em assinaturas. Apesar desse fato não ser falso, é necessário considerar que os ataques recentes podem gerar um volume grande de dados e que cada vez mais as técnicas vão se diversificando, para que os atacantes não sejam facilmente descobertos. Além disso, o método proposto utiliza sete atributos fixos, sem considerar que ao longo do tempo esses atributos possam estar defasados e que talvez outros atributos sejam mais expressivos em relação ao estado normal da rede e ao tráfego anômalo provavelmente oriundo de um ataque. Por isso, não existe uma fase de seleção de atributos.

O método de detecção de ataques *zero-day* descrito por (Casas et al., 2012) se mostra relevante ao tema por não necessitar de dados rotulados para a análise mas que nos testes não utilizou a seleção não-supervisionada de atributos. Foram utilizados nove atributos físicos e os resultados são satisfatórios, especialmente pelo baixo número de atributos utilizados. Dessa forma, sugere-se que o método seja eficiente para detectar ataques *zero-day* com custo baixo desde que existam poucos atributos e isso dependerá do método de seleção de atributos utilizados e do tipo de dado analisado.

No trabalho de (Flanagan et al., 2019), embora a rede proposta gere anomalias que provém *insights* interessantes, é questionável a necessidade de submeter os dados a um processo tão demorado quanto a geração de imagens a partir de uma fonte extremamente volumosa. Os próprios autores argumentam que afirmam que a técnica ainda não foi testada de maneira *online* e que acreditam que com o *hardware* certo é possível fazê-lo, o que dificulta seu uso na prática. Além disso, a seleção dos atributos a serem considerados especialmente na fase de clusterização são um tanto genéricos - não há de fato uma análise para verificar se esses atributos são relevantes.

Nos trabalhos da área de seleção de atributos, (Huang et al., 2015) e (Eskandari e Javidi, 2016) enfatizam a necessidade de métodos *onlines* que possam analisar os *streams* de atributos em tempo real e que não dependam exclusivamente de dados históricos, armazenados em memória, como é comum aos métodos *batch*. No entanto, há a preocupação em saber como esses métodos em *streaming* se comportam em um cenário de entrada massiva de dados em um curto espaço de tempo. Isso se deve ao fato que, em ambos os trabalhos, os resultados experimentais indicam um baixo desempenho. Os casos mais críticos, como os de detecção de ataques, exigem ação rápida para impedir ou mitigar os impactos causados pelo ataque ou por qualquer atividade que esteja sendo monitorada.

(Huang et al., 2015) apresentam conceitos importantes e intrínsecos ao problema de seleção de atributos em *streaming*, tendo em vista que as mudanças de conceito, os chamados *concept-drifts*, nos ataques é um fenômeno real. Um exemplo clássico disso é que ao final da década de 1990 e meados dos anos 2000, um servidor *web* recebendo inúmeras requisições ao mesmo tempo era um claro indicativo de ataques DoS. Atualmente, essas inúmeras requisições simultâneas podem ser consideradas comportamento normal, pois o acesso à Internet aumentou bastante desde então. Em virtude das redes sociais e de serviços online, o número de acessos também tende a aumentar, principalmente em épocas de vendas altas como *Black Friday*, Dia das Mães, entre outras datas comemorativas. Então, é importante considerar que os aspectos de um ataque podem mudar ao longo do tempo e que um método de detecção deve ser capaz de detectar essas mudanças.

Finalmente, a Tabela 3.1 mostra um comparativo entre os trabalhos discutidos neste capítulo. A escolha dos critérios considera os aspectos que melhor se adaptam à natureza do problema. Portanto, os critérios para comparação são quatro:

1. O método é capaz de lidar com dados e/ou atributos em *streaming*, sem que seja necessário o conhecimento prévio total ou parcial dos espaços de dados e/ou atributos? As análises podem ser feitas em tempo real e com espaço de armazenamento limitado?

2. O método consegue reconhecer mudanças no conceito ao longo do tempo, avaliando se atributos considerados relevantes ainda o são?

3. O método é adequado aos dados de redes e pode aplicar transformações nesses dados a fim de extrair e calcular atributos, sem perder as informações intrínsecas das conexões?

	(1) *Online*	(2) *Concept-drift*	(3) Redes
(Song et al., 2008)	**parcial**	não	**sim**
(Casas et al., 2012)	**parcial**	não	**sim**
(Osanaiye et al., 2016)	não	não	**sim**
(Aygun e Yavuz, 2017)	não	**parcial**	**sim**
(Zhang et al., 2017)	não	não	**sim**
(Flanagan et al., 2019)	**sim**	**parcial**	**sim**
(Khan et al., 2018)	não	não	**sim**
(Huang et al., 2015)	**sim**	**sim**	**parcial**
(Eskandari e Javidi, 2016)	**sim**	**sim**	não
(Li et al., 2016b)	**sim**	**parcial**	**sim**

Tabela 3.1: Tabela comparativa entre métodos apresentados trabalhos relacionados

3.4 RESUMO DO CAPÍTULO

Neste capítulo foram apresentados os trabalhos relacionados, divididos em duas categorias, e discutidos os aspectos individuais e em comum destes. Também foram discutidos os métodos desenvolvidos e levantados os pontos mais relevantes que podem ser aplicados no problema seleção de atributos em *streaming* para detecção de ataques *zero-day*. A partir desse levantamento, foi possível determinar as características que devem estar presentes em um método voltado para este problema. Embora os métodos da categoria de detecção de ataques *zero-day* não se apliquem a *streams*, há técnicas de extração de informação que podem ser aproveitadas para a extração de atributos dos dados de tráfego de redes, como se propõe neste trabalho.

Os trabalhos da categoria de seleção de atributos mostram as técnicas presentes no estado da arte para processar dados em tempo real e executar a seleção de maneira *online*. Um dos principais desafios dessa tarefa é desenvolver um método que não precise reanalisar dados já analisados e que não cause sobrecarga na memória. Isso se deve principalmente ao fato que as aplicações que geram *streams* de dados geralmente o fazem de maneira volumosa, sendo inviável, em termos de espaço, processar tudo em memória. Além disso, é importante que os métodos se adaptem à mudança de conceitos ao longo do tempo, visto que comportamentos considerados normais no passado podem tornar-se parte de um conjunto de ações que se caracterizam como comportamento anormal em um outro momento.

Por fim, foi realizado um comparativo entre os trabalhos presentes no capítulo. Os critérios de comparação são referentes à habilidade do algoritmo, método ou técnica apresentada no trabalho em lidar com *streams* de dados e de atributos, detectar mudanças de conceito ao longo do tempo e processar dados não-rotulados de tráfegos de rede. A partir do levantamento desses critérios, obtém-se os problemas que devem ser cobertos pela metodologia.

4 METODOLOGIA PARA EXTRAÇÃO E SELEÇÃO DE ATRIBUTOS

Este capítulo apresenta uma metodologia de três passos para a extração e seleção de atributos em *stream* de dados. A Seção 4.1 discute os aspectos que levaram à decisão de remover o foco no desenvolvimento de algoritmo de seleção de atributos e colocá-lo na descrição de uma metodologia. A Seção 4.2 apresenta a visão geral da metodologia e seus estágios, uma breve discussão sobre detecção de anomalias e seu papel na identificação de ataques *zero-day* e também nas formas de obtenção dos dados. As Seções 4.3, 4.4 e 4.5 abordam em detalhes os Estágios I, II e III da metodologia, respectivamente.

4.1 CONTEXTUALIZAÇÃO

Através da pesquisa realizada na literatura sobre o uso de métodos *online* não-supervisionados de seleção de atributos em *streams* de dados na detecção de ataques *zero-day*, foram identificadas técnicas relevantes para a execução dessa tarefa e também lacunas a serem preenchidas. Essas lacunas não necessariamente significam que há falhas graves nas técnicas estudadas, mas sim sugerem que existe espaço para melhorias de funcionalidades existentes.

O principal ponto levantado no estudo foi que os métodos de detecção de ataque *zero-day* podem apresentar desempenho reduzido quando aplicados em *streams* de dados, mesmo que realizem a etapa de seleção de atributos. Ainda, existe a preocupação em observar as mudanças que ocorrem naturalmente nas redes de computadores, tornando normais os comportamentos antes considerados anômalos, sendo que o inverso também pode ocorrer. Os atacantes estão cientes dessas mudanças e se adaptam criando novos ataques, então as técnicas de detecção devem acompanhar esse movimento. Finalmente, no ponto de vista da seleção de atributos, essa etapa deve ser planejada dentro do contexto de detecção de ataques *zero-day*, considerando as particularidades dos dados analisados – na maioria das vezes, pacotes de rede gerados por ferramentas de captura e monitoramento de dados.

Com base nessas informações a princípio propomos um algoritmo de seleção de atributos em *streaming* chamado de α-*investing*$^+$, apresentado em (Moussa et al., 2019). O α-*investing*$^+$ é baseado no α-*investing* (Zhou et al., 2005), um dos primeiros algoritmos dessa categoria a serem descritos na literatura. O α-*investing*$^+$ é um filtro, cujo critério de seleção é o resultado da execução do Teste Chi-Quadrado para Independência de Variáveis. Quando um novo atributo é detectado no *stream*, ele é adicionado a um conjunto de atributos conhecidos. Então, realiza-se o teste para determinar se o novo atributo é estatisticamente dependente de algum dos atributos presentes no conjunto de atributos selecionados. Se o resultado for verdadeiro, o atributo é descartado, caso contrário, o atributo é adicionado ao conjunto dos atributos selecionados. Todos os elementos do conjunto de atributos selecionados são estatisticamente independentes entre si. A Figura 4.1 ilustra esse processo.

O principal problema com o α-*investing*$^+$ é em relação a maneira que o conjunto de entrada é tratado. Ali, o conjunto é composto por pacotes gerados por ferramentas de capturas de tráfego, sendo que cada pacote é avaliado individualmente. Esse tipo de avaliação desrespeita a própria natureza dos dados de rede, pois um único pacote não provê informações necessárias para caracterizar a rede como um todo. Por isso, a inferência de mudanças de estado (por exemplo, de um estado normal – sem ataques – a um estado anômalo – com ataques) a partir de observações unitárias não é confiável. Ainda que o conjunto de entrada seja composto por coleções de pacotes,

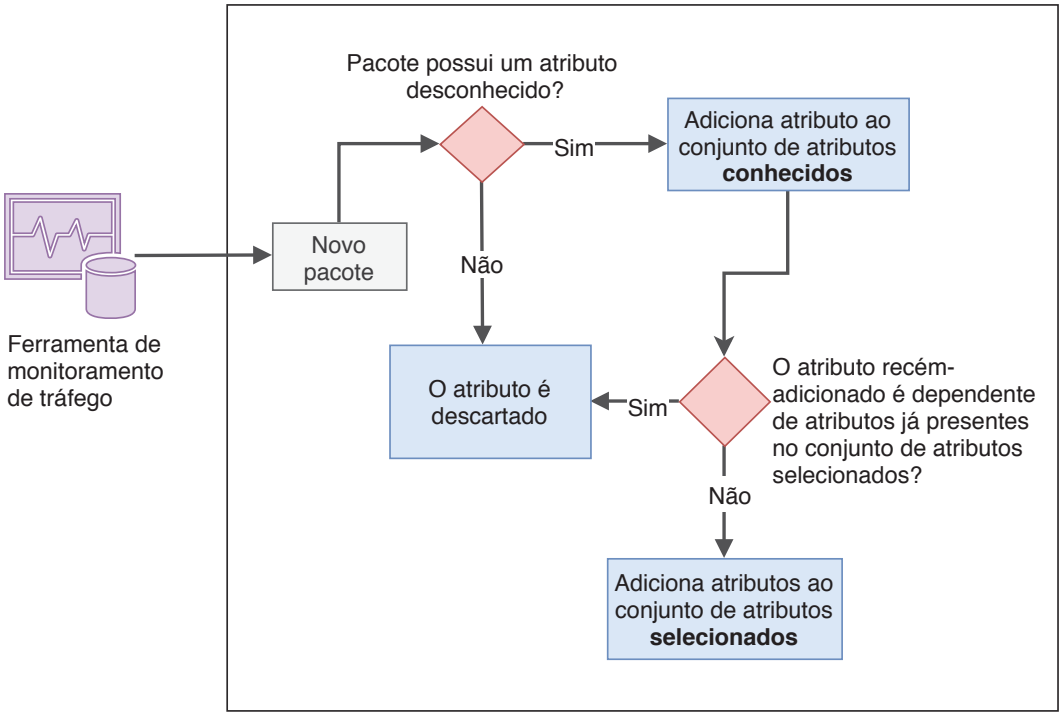


Figura 4.1: Critério de seleção do algoritmo α-*investing*$^+$

como os *NetFlow* que são gerados a partir do fluxo dos dados capturados na rede, há outro problema a ser considerado que é o critério de seleção.

O critério de seleção do α-*investing*$^+$ não acompanha as mudanças da rede, tampouco reavalia os atributos selecionados para decidir se estes continuam sendo os mais descritivos em relação à realidade atual da rede. Outro ponto é que não há um tratamento específico para o *stream* de dados, que apesar de ser uma entrada válida – ou seja, o algoritmo não rejeita a adição de elementos novos no espaço de dados – não há qualquer condição ou ferramenta que avalie novamente o conjunto de atributos selecionados na presença de novos dados. Como o espaço de dados é crescente e volátil, o conjunto de atributos selecionados fica defasado e com vícios ao longo do tempo. Mais detalhes sobre o α-*investing*$^+$ podem ser lidos no Apêndice A.

A abordagem na criação do α-*investing*$^+$ estava focada na proposta de uma ferramenta – no caso, um algoritmo – que fosse capaz de identificar atributos em *streams* de dados para indicar a possível existência de um ataque desconhecido em curso. Embora o algoritmo esteja distante desse objetivo, seu processo de desenvolvimento e, posteriormente, o reconhecimento de suas falhas foram proveitosos para gerar *insights* sobre o problema e pensar em outras formas de resolvê-lo. A partir dessas reflexões e dos pontos já expostos, optou-se então por remover o foco da solução e atribuí-lo ao problema. Por consequência, surge a metodologia apresentada neste trabalho, cujo objetivo é propor caminhos para as possíveis soluções dos pontos levantados durante o desenvolvimento do α-*investing*$^+$ que direcionam para a resolução do problema que este se propunha a tratar.

4.2 VISÃO GERAL E DEFINIÇÕES INICIAIS

A metodologia é apresentada em três estágios que descrevem uma cadeia de processos para extração e seleção de atributos em *streams* de dados de rede, dado o contexto de detecção de ataques *zero-day*. O objetivo é prover diretivas para guiar e facilitar o desenvolvimento de métodos ou técnicas de seleção de atributos que enderecem esse problema específico. A premissa

é que a observação de atributos que se desviam do comportamento normal da rede observada pode indicar um ataque *zero-day* em curso. Ademais, assume-se que a detecção de ataques *zero-day* é realizada por um método de detecção de anomalias, cuja definição e detalhamento não está no escopo deste trabalho. Por isso, é importante ressaltar que a metodologia deve ser seguida como parte do processo de detecção de ataques *zero-day*, cujo foco está na fase de pré-processamento dos dados.

Por simplicidade, assume-se que algumas das descrições apresentadas a partir daqui são referentes a um método hipotético que foi desenvolvido de acordo com as diretivas da metodologia proposta. Portanto, sempre que houver a menção a um método, a referência é à metodologia. As diretivas estão organizadas em estágios de acordo com seu propósito: extrair atributos, selecionar atributos e verificar mudança de conceito. A Figura 4.2 apresenta uma visualização da metodologia, ilustrando o fluxo do *stream* de dados pelos estágios e a interação com a base de perfis normais da rede. Essa base é gerada por outra etapa do processo de detecção de anomalia que não está no escopo deste trabalho. Os estágios são descritos a seguir.

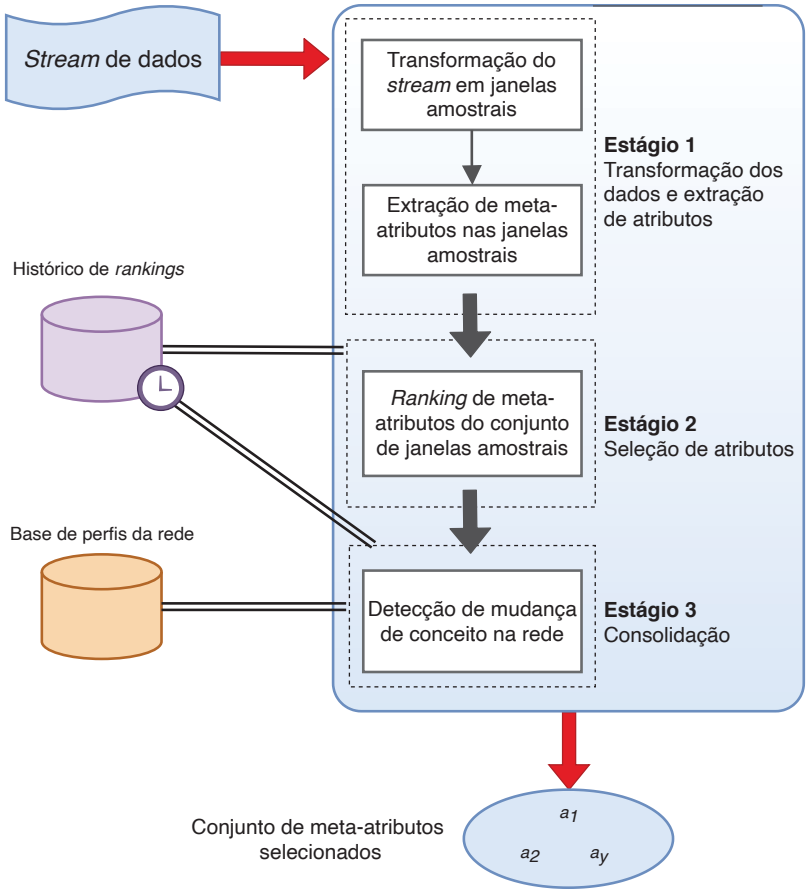


Figura 4.2: Visualização da metodologia

1. **Estágio I: Transformação dos dados e extração de atributos.** Nesse primeiro estágio, são selecionadas amostras do *stream* de dados para extrair um conjunto de meta-atributos, calculados a partir dos atributos originais do *stream*. Esses meta-atributos garantem que a análise não será feita sob um volume alto de dados e sim sob um conjunto reduzido.

2. **Estágio II: Seleção de atributos.** Os meta-atributos extraídos do Estágio I servem de dados de entrada a um algoritmo de seleção de atributos. O objetivo é encontrar

	Estágio I	*Estágio II*	*Estágio III*
Entrada(s)	*Stream* de dados de rede	Conjunto de meta-atributos	*Ranking* de meta-atributos e conjunto de meta-atributos conhecidos;
Saída(s)	Conjunto de meta-atributos	*Ranking* de meta-atributos	Conjunto de meta-atributos selecionados
Processo(s)	Amostragem de dados e extração de atributos	Seleção de atributos, alimentação do histórico de *rankings* e do conjunto de atributos conhecidos	Detecção de mudança de conceito
Pré-requisito	Não há	Não há	Perfil normal da rede e histórico de *rankings*

Tabela 4.1: Descrição das entradas, saídas, processos e pré-condições em cada estágio.

os meta-atributos mais representativos naquele conjunto, gerando um *ranking* que os classifica de acordo com sua relevância. Cada *ranking* é armazenado em um histórico de *rankings*, para comparação no Estágio III. Nesse estágio é construído também o conjunto de meta-atributos já conhecidos na rede, importante para identificar novos meta-atributos.

3. **Estágio III: Consolidação.** Nesse estágio, verifica-se a existência de mudanças significativas entre o *ranking* recém-gerado no Estágio II e os *rankings* mais recentes armazenados no histórico. Esse processo detecta mudanças de conceito. As mudanças significativas indicam a possível existência de uma anomalia na rede e os meta-atributos relacionados devem ser considerados na tarefa de detecção de anomalia.

Cada estágio possui obrigatoriamente uma entrada, uma saída, um processo e um pré-requisito associado. A **entrada** é o conjunto de dados que é apresentado ao estágio e a **saída** é o conjunto de dados obtido através da aplicação de um processo sob o conjunto de entrada. O **processo** é a tarefa cuja função é transformar o conjunto de entrada no conjunto de saída esperado naquele estágio. Finalmente, um **pré-requisito** é um dado que não está no conjunto de entrada, mas necessário para o funcionamento do processo daquele estágio. A Tabela 4.1 exibe as entradas e saídas esperadas em cada estágio, assim como os processos e as pré-condições associadas. Os detalhes de cada item são especificados na subseção referente ao estágio.

Um dos pré-requisitos definidos é o **perfil normal** da rede. (Bhuyan et al., 2014) afirmam que o conceito de *normal*, no contexto de detecção de anomalias, apresenta-se através de um modelo que expressa as relações entre as variáveis envolvidas na dinâmica do sistema. Esse modelo, então, é a norma daquele sistema e também pode ser chamado de perfil normal. Uma variação significativa da norma é considerada uma anomalia. O perfil normal da rede é extraído através de um algoritmo de detecção de anomalias. A Subseção 4.2.2 apresenta brevemente alguns algoritmos encontrados na literatura que podem desempenhar essa função.

Um perfil normal de uma rede é gerado a partir de análises do seu histórico de tráfego limpo, isto é, dados que expressem o funcionamento normal da rede sem a interferência de ataques ou intrusões. Esse é um processo potencialmente demorado e complexo de ser feito. Uma rede do mundo real, por exemplo, teria que ser analisada por dias para a construção de seu perfil normal e qualquer ataque ou intrusão presentes nesse meio tempo podem afetar essa tarefa. O caso do perfil normal utilizado pela metodologia não é diferente. Dessa forma, é necessário

considerar esse tempo de treinamento e construção do perfil normal quando a metodologia for utilizada.

4.2.1 Captura de dados da rede

Os dados de tráfego de uma rede podem ser originados de diversas maneiras e formatos, dependendo do dispositivo analisado e da maneira que são obtidos. As ferramentas de captura de tráfego de redes são uma das formas de extrair esses dados, sendo estas responsáveis por monitorar, registrar e às vezes analisar o comportamento da rede. Uma das formas mais comuns para realizar essa tarefa é através da ferramenta `tcpdump`[1], que é um analisador de pacotes, em conjunto com o `libpcap`, que é uma biblioteca escrita em C/C++ para captura de tráfego. O `tcpdump` exibe o conteúdo dos pacotes e *frames* capturados em uma determinada interface de rede, de acordo com os parâmetros estabelecidos na execução do comando.

Embora as capturas possam ser realizadas em qualquer ponto da rede – computadores, servidores, roteadores, dispositivos móveis e de IoT, entre outros – recomenda-se fazê-las nos locais que podem ser interessantes aos atacantes. Para detectar ataques *zero-day*, entretanto, não é uma tarefa trivial determinar esses pontos, pois não se conhece o atacante, seu alvo nem seu objetivo. Nesse caso, o recomendado é posicionar-se em pontos críticos na rede, em que há entrada de tráfego externo e que hospedam os dados, serviços ou dispositivos mais sensíveis da rede. Essa decisão é dependente das particularidades de cada cenário.

A Figura 4.3 apresenta uma configuração em um cenário bastante comum, com servidores *web*, banco de dados e um *firewall* filtrando todo o tráfego que entra e sai da Internet. Os IDSs de rede geralmente são posicionados dessa forma. A detecção de ataques *zero-day* ocorre após o filtro do *firewall* e ali está inserido o método para extrair e selecionar atributos dos dados que estão entrando na rede.

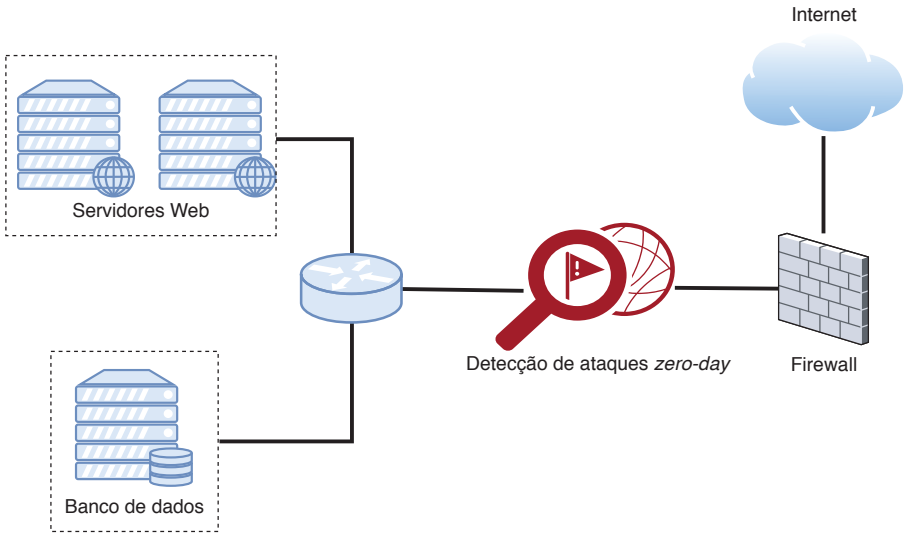


Figura 4.3: Detecção com dados filtrados pelo *firewall*

4.2.2 Algoritmos para detecção de anomalias

Na literatura, há diversos trabalhos que abordam especificamente a tarefa de detecção de anomalias. Apesar da metodologia estar inserida no contexto de detecção de anomalias, o

[1] *Website* oficial do `tcpdump` e `libpcap`: https://www.tcpdump.org/.

objetivo deste trabalho não é definir nem detalhar a aplicação de um algoritmo desse tipo. Por isso, são listados brevemente exemplos de técnicas que podem ser utilizadas. Classificadores *one-class* são utilizados em (Song et al., 2008), presente na seção 3, e no trabalho de (Aleroud e Karabatis, 2012). Os classificadores *one-class* tentam identificar se um determinado objeto pertence ou não à uma única classe presente no problema. Técnicas não-supervisionadas são aplicadas em (Casas et al., 2012) e (Ahmed e Mahmood, 2015).

(Song et al., 2008) usam uma SVM *one-class* para extrair ataques *zero-day* dos conjuntos de treinamento e testes. O objetivo é mapear os dados não-rotulados de tráfego de rede para pontos em um espaço de atributos através de um mapeamento não-linear e criar uma hiperesfera onde está contida a maioria dos pontos do espaço de atributos. Desse modo, os pontos que estão fora da hiperesfera são classificados como anomalias, considerando que em um conjunto de dados gerados por IDS há poucas instâncias referentes a um ataque *zero-day*. (Aleroud e Karabatis, 2012) utilizam um algoritmo de *One Class Nearest Neighbor* (1-NN) para identificar novos tipos de ataques. A abordagem proposta, primeiramente, cria perfis de mau uso e de anomalias para serem utilizados na detecção. Essa etapa ocorre de maneira *offline*. A etapa seguinte ocorre em tempo real e utiliza os perfis para detectar ataques conhecidos e desconhecidos. O algoritmo 1-NN é empregado para detectar desvios do comportamento normal.

(Casas et al., 2012) apresentam um algoritmo com múltiplas técnicas de *clustering* para detectar mau uso e anomalias em uma rede. O algoritmo é uma combinação das técnicas de *Sub-Space Clustering* (SSC), *Density-based Clustering* e *Evidence Accumulation Clustering* (EAC). O objetivo é obter conhecimento sobre fluxos de dados marcados como anômalos. Os fluxos são dados como anômalos através de uma análise que verifica o volume do tráfego em bytes, número de pacotes e número de fluxos em um determinado período de tempo. Após obter o conhecimento, o algoritmo faz um *ranking* baseado no grau de anormalidade de todos os fluxos anômalos, considerados *outliers*. (Ahmed e Mahmood, 2015) também aplicam uma técnica de *clustering* para a detecção de anomalias, chamada *Co-clustering*. Os autores visam encontrar padrões no tráfego de rede para analisá-los utilizando uma adaptação de *co-clustering*, que adiciona elementos de teoria da informação para melhorar os resultados e diminuir a taxa de detecção de falsos positivos.

4.3 ESTÁGIO I: TRANSFORMAÇÃO DOS DADOS E EXTRAÇÃO DE META-ATRIBUTOS

No primeiro estágio, o objetivo é capturar o *stream* de dados e gerar um conjunto de meta-atributos que serão submetidos à seleção no Estágio II. O processo presente nesse estágio é responsável por selecionar amostras do *stream* de dados, chamadas **janelas amostrais**, que são de tamanho variável, e a partir dessas extrair os atributos. Esses atributos são calculados a partir dos atributos de cada amostra e são os chamados **meta-atributos**, cujo conjunto é potencialmente infinito. Ao longo do tempo, é possível que o conjunto de meta-atributos receba novos elementos e também que elementos antigos sejam removidos, dependendo do quão relevantes eles são em um determinado momento t. A Figura 4.4 apresenta o processo completo desse primeiro estágio.

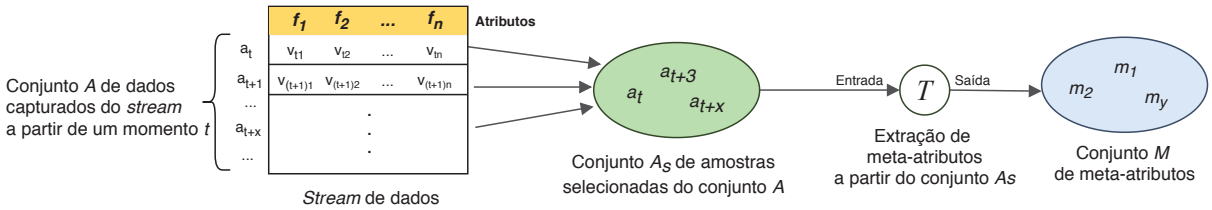


Figura 4.4: Representação em alto nível do Estágio I

Na Figura 4.4, observa-se os dados sendo apresentados de forma contínua e com a captura iniciando a partir de um momento t, tal que $t \geq 0$. Cada dado é uma tupla $\langle f_0, f_1, ..., f_n \rangle \in A$, denotada como a, sendo $n > 0$ o número de atributos na tupla e A o conjunto de tuplas. Cada atributo f_i pertence ao conjunto F, que contém todos os atributos presentes no conjunto de dados. Uma tupla a_t é uma tupla capturada em um momento t, assim como a_{t+x} é uma tupla capturada em um momento $t + x$, tal que $x > 0$. Uma janela amostral é um conjunto $A_s \subseteq A$ com tuplas consecutivas. Ou seja $A_s = \{a_s, a_{s+1}, \ldots, a_{s+t}\}$, para um dado momento s inicial e um tamanho t. Uma janela amostral, dessa forma, é uma representação do *stream* de dados no momento de sua captura.

A partir do conjunto A_s, são gerados os meta-atributos, denotados como m. Os meta-atributos pertencem ao conjunto M, de tamanho potencialmente infinito, e eles são extraídos a partir do conjunto de atributos de cada $a_s \in A_s$. A extração de meta-atributos pode ser oriunda de uma operação de transformação de dados, cálculos estatísticos ou até mesmo ser pré-definidos pelo usuário.

4.4 ESTÁGIO II - SELEÇÃO DE META-ATRIBUTOS

Nesse estágio, o objetivo é realizar a seleção dos meta-atributos obtidos através dos processos de captura e extração de dados no Estágio I. O filtro univariável é a estratégia de seleção utilizada no processo. A Figura 4.5 mostra como o conjunto M de meta-atributos, oriundo do processo de extração de atributos do Estágio I, sofre transformações ao longo dos processos do Estágio II.

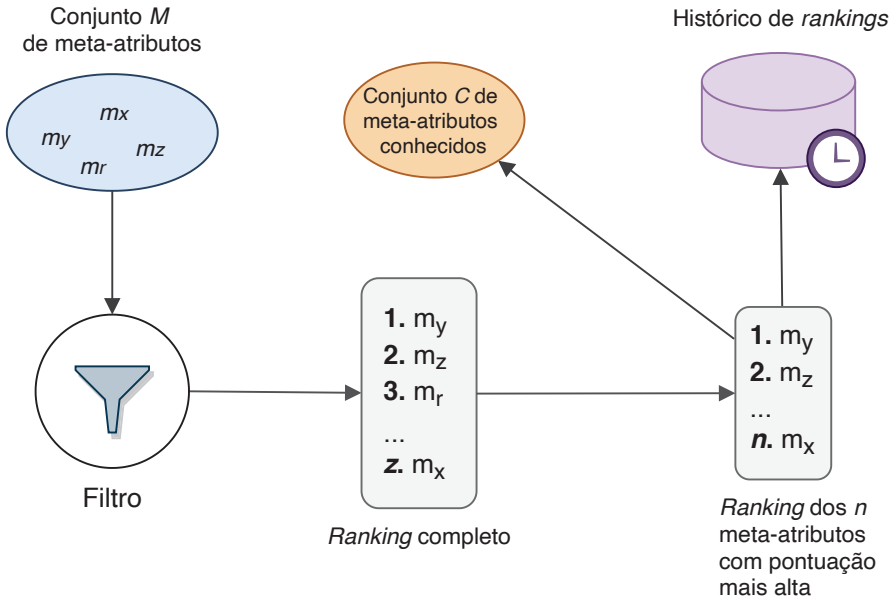


Figura 4.5: Representação em alto nível do Estágio II

O filtro irá gerar um ***ranking*** a partir do conjunto M, do mais relevante ao menos relevante avaliados naquela janela amostral. Esse *ranking* deve sofrer um corte, para que apenas os meta-atributos mais bem-colocados possam prosseguir para o Estágio III. O número de meta-atributos a serem escolhidos é arbitrário e um novo *ranking* é gerado a partir desse corte. Esse novo *ranking* é armazenado em um **histórico de *rankings***, que contém somente os *rankings* que sofreram corte.

Cada meta-atributo único que já foi observado no *stream* e que obteve uma boa colocação em um *ranking* é armazenado no conjunto de meta-atributos conhecidos, denotado como C. Ou seja, é possível que um meta-atributo observado anteriormente seja considerado novo caso ele nunca tenha obtido uma boa pontuação nos *rankings* antes. Quando um meta-atributo é adicionado ao conjunto C, necessariamente o *ranking*, pode ser necessária a definição de um novo perfil normal que o inclua. Essa decisão é tomada no Estágio III.

É importante ressaltar que conceito de relevância em um filtro univariável não considera a redundância de atributos, mas o critério de seleção típico dessa categoria traz resultados de maneira mais veloz que outras abordagens, como as dos filtros multivariáveis, os *wrappers* ou os *embedded*. A escolha é baseada no reconhecimento da existência de um *trade-off* entre utilizar uma abordagem que trará um conjunto contendo somente atributos fortemente relevantes, porém mais lenta, e utilizar uma abordagem rápida que trará um conjunto com mais atributos. Em casos críticos, como a ocorrência de um ataque volumétrico, é preferível que nesse estágio a seleção de atributos seja rápida.

4.5 ESTÁGIO III - CONSOLIDAÇÃO

Nesse estágio, são recebidos o *ranking* de meta-atributos e o conjunto atual de meta-atributos. Primeiramente, o *ranking* recém-gerado recebe um peso w, que indica o quão recente ele é. Quanto maior é o valor de w, mais recente é o *ranking*. Todos os *rankings* anteriores, gerados a partir de um momento t_0, têm um peso w_{t_i} associado, sendo que $0 \geq t \geq i$ e i é i-ésimo *ranking* gerado. Naturalmente, $w_{t_i} > w_{t_{(i-1)}}$. Após a atribuição de peso, o *ranking* é comparado a um número de *rankings* mais antigos que estão presentes no histórico. O objetivo é verificar se existe um desvio significativo em relação ao conjunto selecionado. A Figura 4.6 mostra um gráfico que ilustra a relação entre os *rankings* e seu peso através do tempo. Cada barra é um *ranking* e a linha vermelha indica um exemplo de corte para selecionar o conjunto de *rankings* mais recentes para comparação com o atual – no caso, de tamanho 4.

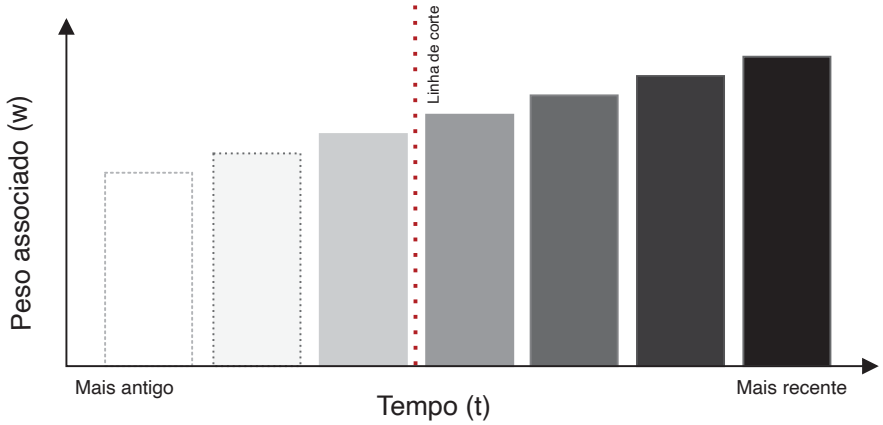


Figura 4.6: Relação entre peso e tempo dos *rankings* calculados na rede e a simulação de uma linha de corte.

Um *ranking* que desvia significativamente do conjunto escolhido de *rankings* recentes pode estar relacionado a uma anomalia. A confirmação da existência de uma anomalia não é feita por nenhum dos estágios e sim pelo algoritmo de detecção de anomalias em que o método está inserido. Caso o *ranking* de fato contenha comportamento anômalo, não é recomendado adicioná-lo no histórico de *rankings*, pois ali assume-se que estão os *rankings* que representam o comportamento normal da rede. O descarte desse *ranking* nessa etapa diz respeito somente ao

processo de detecção de mudança de conceitos – todos os *rankings* que chegam como entrada no Estágio III são convertidos em saída. Nem mesmo o peso w é incluído na saída, visto que esse, a comparação de *rankings* e a verificação de detecção de novo atributo são parte de processos internos do estágio.

Como citado acima, a detecção de novo atributo é um dos processos presentes no Estágio III. A verificação é simples e consiste em buscar um ou mais meta-atributos contidos no *ranking* que ainda não estão contidos no conjunto C de atributos conhecidos. A presença de novos atributos pode desencadear a definição de um novo perfil normal para a rede, sendo essa tarefa opcional. É necessário considerar, porém, que a comparação do *ranking* que contém novos atributos com um determinado número de *rankings* do histórico pode indicar que há discrepância entre os dois – o que não significa necessariamente uma anomalia. Na verdade, espera-se que haja diferenças, caso contrário o novo atributo nem mesmo ocuparia uma posição relevante no *ranking* caso não fosse significativo nesse conjunto. Como os *rankings* anteriores não possuem esse atributo, é preciso afrouxar o limite do que se consideraria um desvio grande do *ranking* mais recente em relação aos demais.

Tanto a detecção de novos atributos quanto a construção e atualização do histórico de *rankings* que representam o comportamento normal garante que os dados não tenham vícios que ignorem as mudanças naturais que ocorrem em uma rede. Aqui, entende-se e assume-se que uma mudança de conceito, no contexto de ataques *zero-day*, não é um processo abrupto e que requer uma longa observação. Por isso, recomenda-se cautela ao atualizar o histórico de *rankings* e também nos gatilhos para a definição de um novo perfil normal, quando é necessário. O objetivo da metodologia é, em suma, realizar uma triagem no *stream* de dados e indicar mudanças suspeitas que possam significar ataque – esse último sendo evidenciado nesse estágio.

4.6 RESUMO DO CAPÍTULO

Esse capítulo apresentou uma metodologia para extração e seleção de atributos em *stream* de dados, no contexto de detecção de ataques *zero-day*. Primeiramente, contextualizou-se o problema para justificar a construção da metodologia e seus detalhes. Foram expostos também os pontos de falha do algoritmo de seleção de atributos em *streaming* α*-investing*$^+$, que foi o primeiro esforço realizado para resolver o problema. Também discutiu-se como a construção do α*-investing*$^+$ proveu um melhor entendimento de como a seleção de atributos deve funcionar dentro do cenário e condições propostas.

Em seguida, a visão geral da metodologia e dos seus três estágios e foram definidas as entradas, saídas, processos e pré-requisitos de cada um. Ademais, foram elucidados questões adjacentes – porém importantes – para o funcionamento da metodologia, como a maneira de obter os dados, definição de perfil normal e uma breve explicação de alguns métodos de detecção de anomalia. Finalmente, foram apresentados os detalhes de cada estágio, explicando seus processos e como o *stream* de dados é transformado em um conjunto de atributos – consolidando o objetivo principal da metodologia que é selecionar atributos.

5 CONCLUSÃO

Neste trabalho, foi apresentada uma metodologia de extração e seleção de atributos de *stream* de redes, para auxiliar na detecção de ataques desconhecidos, chamados de *zero-day*. Essa tarefa é um passo importante para a detecção de ataques *zero-day*, pois indica quais atributos devem ser observados e descarta informações irrelevantes. A metodologia contém três estágios e a saída esperada é um conjunto de meta-atributos, extraídos a partir de janelas construídas com amostras do *stream* de dados e selecionadas por um algoritmo de seleção de atributos. A definição da metodologia surgiu a partir do estudo do problema e das alternativas presentes na literatura que se aproximam de sua solução. Foi percebida assim a dificuldade de encontrar trabalhos que enderecem a seleção de atributos em *stream* de dados aplicada no contexto de detecção de ataques *zero-day*.

A literatura apresenta diversos algoritmos e técnicas de seleção de atributos que atendam a parte das características do problema e métodos de detecção de ataques *zero-day* que pouco discutiam a etapa de seleção de atributos, quando existia uma. Ao identificar esses pontos, primeiramente propomos um algoritmo de detecção de atributos em *streaming* chamado α-*investing*$^+$. Porém, o algoritmo ainda apresentava uma solução incompleta, que não considerava a mudança de conceito no conjunto de dados e também falhava em prover uma visão mais geral da rede no momento da captura dos dados. Isso se deve ao fato que no α-*investing*$^+$ os dados de entrada eram pacotes de rede analisados de maneira individual.

Então, percebeu-se necessário o estudo mais aprofundado do problema, com foco nas lacunas encontradas na literatura e dos pontos de falha do algoritmo apresentado previamente como solução. A partir desse estudo, surgiu a metodologia, que propõe transformar o conjunto original dos dados, calcular atributos sob o conjunto transformado – chamados meta-atributos – e nele performar a seleção de atributos. A metodologia também prevê mudanças de conceito, através de uma constante atualização de um histórico de *rankings*, que são gerados pelo estágio responsável pela seleção de atributos. A abordagem indicada para realizar a tarefa de seleção foi a de filtro univariável.

5.1 TRABALHOS FUTUROS

Em relação aos trabalhos futuros, a prioridade é realizar experimentos com instâncias construídas seguindo a metodologia apresentada neste trabalho. Entende-se que os experimentos são importantes para encontrar possíveis questões conceituais a serem melhoradas, como aconteceu durante o desenvolvimento do α-*investing*$^+$. Esses experimentos, a princípio, podem ser realizados utilizando combinações de algoritmos já existentes, que atendam aos propósitos e cumpram os requisitos de cada estágio. A configuração ideal para a execução dos experimentos é composta pelos itens abaixo (mas não somente):

- Uma base de dados, experimental ou real, que contenha dados de ataque e dados de tráfego normal. Um exemplo é a base CICIDS2017 (Sharafaldin et al., 2018), que é uma base pública, disponibilizada pela Universidade de New Brunswich em 2017, que contém dados de ataques recentes. Na base estão armazenados registros de tráfego normal e também dados gerados por Sistemas de Detecção de Intrusão (IDS) e por Sistemas de Prevenção de Intrusão (IPS), incluindo dados de ataques de força-bruta, ataques *heartbleed*, *botnets*, ataques DoS e DDoS, ataques Web e ataques de infiltração;

- Um algoritmo que execute o processo de transformação dos dados descrito na Seção 4.3;
- Um método para extrair atributos. O ideal é que seja utilizada uma técnica bem definida, como um algoritmo de extração de atributos ou o cálculo de métricas estatísticas sob o conjunto de dados. Mas, para propósitos experimentais, a extração pode ser feita através de combinações aleatórias de (caso haja poucos atributos no conjunto de dados original) ou a definição estática de meta-atributos;
- Um algoritmo não-supervisionado de seleção de atributos, que siga a abordagem de filtro univariável;
- Um algoritmo de detecção de anomalias, como os apresentados na Subseção 4.2.2. A instância da metodologia deverá gerar atributos que alimentem esse algoritmo.

REFERÊNCIAS

Ahmed, M. e Mahmood, A. N. (2015). Network traffic pattern analysis using improved information theoretic co-clustering based collective anomaly detection. Em *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, volume 153, páginas 204–219.

Aleroud, A. e Karabatis, G. (2012). A contextual anomaly detection approach to discover zero-day attacks. Em *Proceedings of the 2012 ASE International Conference on Cyber Security, CyberSecurity 2012*, páginas 40–45. IEEE.

Aleyani, S., Tang, J. e Liu, H. (2013). Feature Selection for Clustering: A Review. Em *Data Clustering: Algorithms and Applications*, capítulo: 2. Chapman and Hall, 1 edition.

Anderson, R. J. (2008). *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley Publishing, Indianapolis, 2 edition.

Aygun, R. C. e Yavuz, A. G. (2017). Network Anomaly Detection with Stochastically Improved Autoencoder Based Models. Em *Proceedings - 4th IEEE International Conference on Cyber Security and Cloud Computing, CSCloud 2017 and 3rd IEEE International Conference of Scalable and Smart Cloud, SSC 2017*, páginas 193–198. IEEE.

Bhuyan, M. H., Bhattacharyya, D. K. e Kalita, J. K. (2014). Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys and Tutorials*, 16(1):303–336.

Bhuyan, M. H., Bhattacharyya, D. K. e Kalita, J. K. (2017). *Network Traffic Anomaly Detection and Prevention*. Computer Communications and Network. Springer, Cham.

Bilge, L. e Dumitras, T. (2012). Before We Knew It: An Empirical Study of Zero-Day Attacks in The Real World. Em *Proceedings of the 2012 ACM conference on Computer and communications security - CCS '12*, página 833, New York, New York, USA. ACM Press.

Casas, P., Mazel, J. e Owezarski, P. (2012). Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge. Em *Computer Communications*, volume 35, páginas 772–783. Elsevier.

Chandrashekar, G. e Sahin, F. (2014). A survey on feature selection methods. *Computers and Electrical Engineering*, 40(1):16–28.

Cho, K., Mitsuya, K. e Kato, A. (2000). Traffic data repository at the WIDE project. Em *ATEC '00 Proceedings of the annual conference on USENIX Annual Technical Conference*, páginas 51–52.

Douligeris, C. e Mitrokotsa, A. (2004). DDoS attacks and defense mechanisms: Classification and state-of-the-art. *Computer Networks*, 44(5):643–666.

Duch, W. (2008). Filter Methods. Em *Feature Extraction*, páginas 89–117. Springer Berlin Heidelberg, Berlin, Heidelberg.

Eskandari, S. e Javidi, M. M. (2016). Online streaming feature selection using rough sets. *International Journal of Approximate Reasoning*, 69:35–57.

Flanagan, K., Fallon, E., Jacob, P., Awad, A. e Connolly, P. (2019). 2D2N: A Dynamic Degenerative Neural Network for Classification of Images of Live Network Data. Em *2019 16th IEEE Annual Consumer Communications and Networking Conference, CCNC 2019*, páginas 1–7. IEEE.

Freedman, D., Pisani, R. e Purves, R. (1998). *Statistics*. W.W. Norton.

Gama, J. a., Žliobaitė, I., Bifet, A., Pechenizkiy, M. e Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37.

Ghorbani, A. A., Lu, W. e Tavallaee, M. (2010). *Network Intrusion Detection and Prevention*. Springer US.

Guyon, I. e Elisseeff, A. (2008). An Introduction to Feature Extraction. Em *Feature Extraction*, páginas 1–25. Springer Berlin Heidelberg.

Huang, H., Yoo, S. e Kasiviswanathan, S. P. (2015). Unsupervised Feature Selection on Data Streams. Em *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management - CIKM '15*, páginas 1031–1040. ACM Press.

Jain, A. e Zongker, D. (1997). Feature selection: evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):153–158.

Khan, S., Gani, A., Wahab, A. W. A. e Singh, P. K. (2018). Feature Selection of Denial-of-Service Attacks Using Entropy and Granular Computing. *Arabian Journal for Science and Engineering*, 43(2):499–508.

Kohavi, R. e John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324.

Kolias, C., Kambourakis, G., Stavrou, A. e Voas, J. (2017). DDoS in the IoT: Mirai and other botnets. *Computer*, 50(7):80–84.

Lal, T. N., Chapelle, O., Weston, J. e Elisseeff, A. (2008). Embedded Methods. Em *Feature Extraction*, capítulo: 5, páginas 137–165. Springer Berlin Heidelberg.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J. e Liu, H. (2016a). Feature Selection: A Data Perspective. *ACM Computing Surveys*, 50(6):1–73.

Li, J., Hu, X., Wu, L. e Liu, H. (2016b). Robust Unsupervised Feature Selection on Networked Data. Em *Proceedings of the 2016 SIAM International Conference on Data Mining*, páginas 387–395.

Li, Q., Tan, Z., Jamdagni, A., Nanda, P., He, X. e Han, W. (2017). An intrusion detection system based on polynomial feature correlation analysis. Em *Proceedings - 16th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 11th IEEE International Conference on Big Data Science and Engineering and 14th IEEE International Conference on Embedded Software and Systems*, páginas 978–983. IEEE.

Liu, H. e Motoda, H. (2007). *Computational Methods of Feature Selection*. Chapman & Hall/CRC.

Monte, M. (2015). *Network Attacks & Exploitation*. Wiley, 1 edition.

Moussa, A. A., Nogueira, M. e Guedes, A. L. (2019). Seleção Online de Features em Streaming Baseada em Alpha-investing para Ataques DDoS. Em *II Workshop de Gerência e Operação de Redes e Serviços- WGRS*, página 14, Porto Alegre. SBC.

Osanaiye, O., Cai, H., Choo, K. K. R., Dehghantanha, A., Xu, Z. e Dlodlo, M. (2016). Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing. *Eurasip Journal on Wireless Communications and Networking*, 2016(1):130.

Rizzo, M. (2007). *Statistical Computing with R*. Chapman & Hall/CRC The R Series. Taylor & Francis.

Robert, C. P. e Casella, G. (2004). *Monte Carlo Statistical Methods*. Springer-Verlag New York, 2 edition.

Santos, A. A., Nogueira, M. e Moura, J. M. F. (2017). A stochastic adaptive model to explore mobile botnet dynamics. *IEEE Communications Letters*, 21(4):753–756.

Secunia Research (2020). Vulnerability Review 2020. Relatório técnico, Flexera. Disponível em: `https://info.flexera.com/SVM-REPORT-Vulnerability-Review-2020`. Acessado em 11 de agosto de 2020.

Sharafaldin, I., Habibi Lashkari, A. e Ghorbani, A. A. (2018). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. Em *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018)*, páginas 108–116.

Solorio-Fernández, S., Carrasco-Ochoa, J. A. e Martínez-Trinidad, J. F. (2020). A review of unsupervised feature selection methods. *Artificial Intelligence Review*, 53(2):907–948.

Song, J., Takakura, H. e Kwon, Y. (2008). A generalized feature extraction scheme to detect 0-day attacks via IDS alerts. Em *Proceedings - 2008 International Symposium on Applications and the Internet, SAINT 2008*, páginas 55–61. IEEE.

Tavallaee, M., Bagheri, E. e Lu, W. (2009). A detailed analysis of the kdd cup 99 data set. Em *Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*.

Yao, J., Mao, Q., Goodison, S., Mai, V. e Sun, Y. (2015). Feature selection for unsupervised learning through local learning. *Pattern Recognition Letters*, 53:100–107.

Zhang, M., Wang, L., Jajodia, S. e Singhal, A. (2017). Evaluating the network diversity of networks against zero-day attacks. Em *Network Security Metrics*, volume 11, páginas 117–140. IEEE.

Zhou, J., Foster, D., Stine, R. e Ungar, L. (2005). Streaming feature selection using alpha-investing. Em *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining - KDD '05*. ACM Press.

APÊNDICE A – ALGORITMO *ALPHA-INVESTING*$^+$ PARA SELEÇÃO DE ATRIBUTOS EM *STREAMING*

O α-*investing*$^+$ é uma adaptação do algoritmo α-*investing* original, proposto por (Zhou et al., 2005). O α-*investing* não requer um tamanho fixo para o conjunto de atributos, admitindo a adição de novos atributos gerados dinamicamente ao longo do tempo ou a partir de outros já existentes. Além disso, este algoritmo funciona independentemente de como os atributos são gerados. O algoritmo α-*investing*$^+$ foi criado para se adequar à natureza dos dados tratados diante de cenários dos mais diversos ataques. Isso se deve ao fato que muitos dos dados gerados por ferramentas de captura e sistemas de detecção de intrusão são não-paramétricos e categóricos. A decisão de adaptar o algoritmo original para uma versão capaz de lidar com dados crus, sem que tenham passado por nenhuma filtragem, tratamento ou processo que crie uma representação a partir desses dados, dá-se principalmente pelo fato de que a análise precisa acontecer em tempo real, evitando assim atrasos de processamento.

A.1 DEFINIÇÃO DO α-*INVESTING*$^+$

O α-*investing*, base para o α-*investing*$^+$, é um algoritmo de seleção de atributos em *streaming*, que visa diminuir a taxa de falsas descobertas (FDR - *False Discovery Rate*) ao ajustar dinamicamente um limiar α_i. Este limiar determina se um atributo recém-descoberto pode ser adicionado ao conjunto de atributos selecionados (Zhou et al., 2005).

Algoritmo 1 α-investing$^+$

```
1: W ← [0.5]    // Conjunto de pesos
2: w_0 ← W_0    // Peso inicial
3: F ← {}    // Conjunto de atributos detectados
4: S ← {}    // Conjunto de atributos selecionados
5: α_Δ ← 0.5
6: i ← 1
7: while novos atributos forem detectados do
8:     adicionarAtributo(f_i, F)
9:     α_i ← w_i/2 * i
10:    for all s ∈ S tal que s ≠ f_i do
11:        tabela ← gerarTabelaContingencia(f_i, m)
12:        resultado ← chiQuadrado(tabela)
13:        if resultado[valorP] ≤ α_i then
14:            w_{i+1} ← w_i − α_i
15:        else
16:            adicionarAtributo(f_i, S)
17:            w_{i+1} ← w_i + α_Δ − α_i
18:        end if
19:    end for
20:    i ← i + 1
21: end while
```

Seja $D = \{d_1, d_2, ..., d_n\}$ um conjunto de dados gerados até um momento t, tal que $n > 0$ e cada $d_i \in D$, sendo $0 < i \leq n$ e d_i uma j-tupla $\langle f_1, ..., f_j \rangle$ (também chamado de **registro**

neste capítulo). Temos que para $j > 0$, f_k é um campo de texto que descreve uma característica de um d_i e $0 < k \leq j$. Cada f_j é denotado como **atributo**. Sejam $F = \{f_l$: um atributo detectado $| \ l > 0\}$ e $S = \{s_m$: um atributo selecionado $| \ m > 0\}$. Ambos os conjuntos iniciam-se vazios e não admitem elementos repetidos. É importante notar que cada atributo único presente nos registros do conjunto D são pertencentes ao conjunto F e $S \subseteq F$. O algoritmo, ao ser iniciado, recebe um dado registro d_i (linha 7 em Algoritmo 1), que também pode ser representado como d_{t_s}, tal que t_s é o momento inicial da análise e $t_s > t$. Para que um atributo seja adicionado ao conjunto F (linha 8 em Algoritmo 1), são executados os seguintes passos:

1. Lê-se um novo registro;
2. Verificam-se quais atributos têm seus valores preenchidos naquele registro;
3. Descartam-se os atributos com valores não-preenchidos;
4. Para cada atributo que não foi descartado, é verificado se ele já pertence a F;
   (a) Se o atributo está em F, a execução é interrompida e o passo 4 é executado para o próximo atributo;
   (b) Se o atributo não está em F, ele é adicionado ao conjunto.

Um atributo f_i é adicionado a S se ele não tiver relação de dependência com algum s_m em S (linha 16 em Algoritmo 1). O limiar α_i, atualizado na linha 9 do Algoritmo 1, determina a probabilidade de se incluir um atributo irrelevante ao modelo ao i-ésimo passo do algoritmo. O peso w_i representa o número de possíveis falsos positivos a serem incluídos ao modelo nos passos seguintes. As linhas 14 e 17 do Algoritmo 1 mostram a atualização de w_i quando f_i é dependente de algum elemento em S ou independente de algum elemento em S, respectivamente. Um falso positivo, no contexto deste trabalho, é um atributo $x \in S$ tal que existe pelo menos um $y \in S$ que possui uma relação de dependência com x e $x \neq y$. α_i é ajustado a partir do peso w_i. O valor-p corresponde à probabilidade dos atributos f_1 e f_2 serem variáveis independentes e pode ser obtido a partir de um teste estatístico.

Os dados de ataques de redes são, frequentemente, não-paramétricos e categóricos. Assim, é necessário que o teste estatístico utilizado seja adequado a esse tipo de dados. Por isso, neste trabalho optou-se por utilizar o Teste do Chi-Quadrado para Independência de Variáveis, que basicamente é um teste de hipóteses. Neste trabalho, o valor-p é obtido a partir de uma simulação de Monte Carlo utilizada para o Teste Chi-Quadrado para Independência de Variáveis (linha 12 em Algoritmo 1).

O teste Chi-Quadrado pode ser calculado a partir de uma tabela de contingência que determina a frequência dos valores observados entre as categorias de duas variáveis v_1 e v_2 (Freedman et al., 1998) . Cada célula nessa tabela aponta a frequência que cada um dos valores de v_1 ocorre com cada um dos valores de v_2 em uma mesma observação. Em relação às hipóteses que serão testadas, a hipótese nula, denotada como H_0, e a hipótese alternativa, denotada como H_1, são definidas da seguinte forma:

$$H_0\text{: } v_1 \text{ e } v_2 \text{ são independentes entre si}$$
$$H_1\text{: } v_1 \text{ e } v_2 \text{ são dependentes entre si}$$

Tratando-se de dados de rede, uma tabela de contingência pode facilmente tornar-se esparsa, devido à grande variedade de informações contidas em cada variável categórica. Quando o teste Chi-Quadrado é aplicado em tabelas que contém frequências observadas menores do que

1, os resultados retornados não são confiáveis, incluindo o valor-p. Nesse caso, uma opção viável é estimar o valor-p através de uma simulação de Monte Carlo (Robert e Casella, 2004). A partir de uma simulação de Monte Carlo é possível obter um valor empírico do erro de Tipo I, que ocorre quando H_0 é rejeitada quando na realidade ela é verdadeira. O teste é replicado diversas vezes assumindo a H_0 como verdadeira e o erro Tipo I empírico é a amostra proporcional de estatísticas significantes entre os testes replicados (Rizzo, 2007).

O critério de parada do α-*investing*$^+$ é arbitrário, pois enquanto o algoritmo estiver sendo executado, serão esperadas detecções de novos atributos. Ao fim da execução, o conjunto S é denominado vetor de características e assume-se que estas características são as que melhor representam o conjunto de dados analisados do momento t_s até a parada.