UNIVERSIDADE FEDERAL DO PARANÁ

BRUNO MARTINS RAHAL

BOTFETCHER: UMA ARQUITETURA PARA PREDIÇÃO DE ATAQUES DDOS E DETECÇÃO DE BOTS

CURITIBA PR

2021

BRUNO MARTINS RAHAL

BOTFETCHER: UMA ARQUITETURA PARA PREDIÇÃO DE ATAQUES DDOS E DETECÇÃO DE BOTS

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.

Orientador: Profa. Dra. Michele Nogueira Lima.

CURITIBA PR

2021

CATALOGAÇÃO NA FONTE – SIBI/UFPR

R147b Rahal, Bruno Martins

Botfetcher: uma arquitetura para predição de ataques DDOS e detecção de *bots* [recurso eletrônico]/ Bruno Martins Rahal - Curitiba, 2021.

Dissertação apresentada ao curso de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.
Orientadora: Profa. Dra. Michele Nogueira Lima

1. Rede de computadores. 2. Botnet. 3. Proteção de dados. I. Lima, Michele Nogueira. II. Título. III. Universidade Federal do Paraná.

CDD 303.4833

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 **BRUNO MARTINS RAHAL** intitulada: **BOTFETCHER: UMA ARQUITETURA PARA PREDIÇÃO DE ATAQUES DDOS E DETECÇÃO DE BOTS**, sob orientação do Prof. Dr. MICHELE NOGUEIRA LIMA, que após terem inquirido o aluno 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, 15 de Março de 2021.

Assinatura Eletrônica
16/03/2021 13:10:22.0
MICHELE NOGUEIRA LIMA
Presidente da Banca Examinadora (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
16/03/2021 10:46:32.0
ALDRI LUIZ DOS SANTOS
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
17/03/2021 11:02:56.0
RICARDO TOMBESI MACEDO
Avaliador Externo (UNIVERSIDADE FEDERAL DE SANTA MARIA)

A meus filhos, como a água que nutre.

RESUMO

O aumento do número de usuários, de dispositivos e da velocidade de acesso à Internet tem modificado como as pessoas e empresas se relacionam entre si. Esse crescimento, que é exponencial, vem trazendo muitos benefícios à sociedade, tais como facilidades na comunicação, acesso à informação e entretenimento. Entretanto, mecanismos de segurança acabam sendo relegados a segundo plano, deixando os dispositivos vulneráveis a ataques. Uma forma de ataque a esses dispositivos é através de *botnets*, uma ameaça para a segurança dos usuários e dispositivos conectados à Internet. *Botnets* representam uma ameaça a redes de computadores, pois podem interromper seus serviços através da coordenação entre uma quantidade massiva de dispositivos infectados (*bots*). Estas ameaças buscam atacar três pilares dos dados e sistemas: confidencialidade (um dado privado é tornado público), integridade (um dado se torna ilegível) e disponibilidade (o dado ou sistema não está disponível para acessos legítimos). Para evitar os danos causados por *botnets*, é necessário identificar os dispositivos que a compõem. Diversas técnicas foram propostas para identificar *botnets*: analisando todo o tráfego para identificar características conhecidas como ataque; técnicas de identificação de características relevantes; com o crescimento do tráfego, buscou-se inferir o comportamento dos nós da rede sem a análise do conteúdo do tráfego. Porém, em resposta, os atacantes se tornaram mais agressivos, melhorando suas formas de comunicação com os dispositivos comprometidos, a arquitetura da *botnet* e técnicas de evasão dos *bots*. A detecção dessas *botnets* é desafiadora devido à quantidade massiva de dados e às limitações de processamento e memória dos dispositivos presentes na rede. Este trabalho apresenta BotFetcher, uma arquitetura para predição de ataques de negação de serviço distribuído (DDoS) e detecção dos *bots*, capaz de lidar com um volume crescente e massivo de dados. Apresenta-se a fundamentação teórica que subsidia a proposta, indicando como as *botnets* operam, como obter informações para identificação destas e as técnicas para a realização da detecção. BotFetcher contribui para a detecção em escala de botnets, realizando a predição de ataques *DDoS* em redes locais. Identificado um possível ataque, o tráfego é agregado com outras redes, computando a coordenação dos nós, que caracterizam as *botnets*. Para tanto, BotFetcher se utiliza de indicadores estatísticos para a predição dos ataques. Para a detecção, BotFetcher realiza a extração das características do tráfego, selecionando as que propiciam a melhor representação dos dados. Após isso, BotFetcher realiza um agrupamento pelo comportamento dos nós presentes naquele tráfego. O comportamento é indicado pelas características apresentadas. Por fim, a detecção é realizada em cada grupo a partir de indícios de causalidade entre os dispositivos. A avaliação teve como entrada as bases de dados CTU-13 e CAIDA. BotFetcher detectou o(s) *bot(s)* presentes nas bases de dados, com uma precisão maior de 99%. Também, comparou-se a presente proposta com técnicas de aprendizagem de máquina, mostrando que os resultados obtidos são pertinentes e competem com a literatura existente. A solução proposta é capaz de operar em um grande volume de dados, reduzindo o volume de dados analisado a cada etapa e propiciando uma análise eficaz e capaz de detectar *bots* presentes no tráfego.

Palavras-chave: *Botnet*. Detecção de *Botnets*. Segurança. Redes de Computadores.

ABSTRACT

The growth in the number of users, devices and bandwidth at the Internet has changed how people and companies interact with each other. This exponential growth is bringing a lot of benefits to mankind, facilitating communication, access to information and entertainment. However, security measures are often relegated and overlooked, letting the devices vulnerable to attacks. One way to attack these devices is through botnets, a threat to the security of Internet-connected users and devices. Botnets represent a threat to computer networks as they can disrupt its services through the coordination of a huge amount of infected devices (bots). These threats three data and systems fundations: confidentiality (data that is private becomes public), integrity (data becomes unreadable) and availability (data or system is not available for legit requests). Thus, to avoid damage that can be caused by botnets, identifying the devices that belongs to them is required. Several techniques were proposed to identify these botnets: analyzing all the traffic to identify features known as attacks; relevant feature identification techniques; with the traffic growth, efforts were made to characterize the behavior of the network nodes without analyzing the traffic content. But, as a response, attackers became more aggressive, improving their communication techniques with compromised devices, the botnet architecture and bots' evasion techniques. The detection of these botnets is challenging due to the enormous amount of data and the processing and memory limitations of the devices on the network. In this paper, we present BotFetcher, an architecture for DDoS prediction and bot detection, able to deal with an ever increasing and massive data volume. It is presented as the theoretical basis that allows the proposal of BotFetcher, showing how botnets work, how to obtain data to identify these botnets and the techniques that can detect bots. BotFetcher contributes to a scalable botnet detection, performing the prediction of DDoS attacks in local networks. Once a possible attack is identified, traffic is merged with other networks, computing its coordination between nodes, which characterize botnets. Therefore, BotFetcher calculates statistical indicators for the attack prediction. For the detection, BotFetcher extracts features from the traffic and, among them, selects the ones that provide a better data representation. After that, it clusters the nodes on the traffic by its behavior. The behavior is indicated by the features presented. Finally, the detection is done in each cluster, based on causality signaling between devices. The architecture was evaluated using as input the CTU-13 and CAIDA datasets. BotFetcher detected the bot(s) on these datasets with a precision over 99%. Also, the proposal was compared with machine learning techniques, showing that the results are relevant and compete with the existing literature. It has been shown that BotFetcher is able to operate with a massive dataset, reducing the volume of analyzed data at each phase, providing an effective analysis, providing an effective and capable way to detect bots in the traffic.

Keywords: Botnet. Botnet Detetion. Security. Computer Networks.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

ACK	Mensagem de *ACKnowledgement* no protocolo TCP
CTU-13	Base de Dados de Ataques por Botnets da Universidade da República Tcheca (*CTU University, Czech Republic*)
DBSCAN	*Density-Based Spatial Clustering of Applications with Noise*
DDoS	*Distributed Denial of Service* - Negação de Serviço Distribuída
DINF	Departamento de Informática
DNS	*Domain Name System*
DoS	*Denial of Service* - Negação de Serviço
DSPg	*Discrete Signal Processing on Graphs*
HTTP	*Hypertext Transfer Protocol*
ICMP	*Internet Control Message Protocol*
IoT	*Internet of Things* - Internet das Coisas
IP	*Internet Protocol*
IRC	*Internet Relay Chat*
ISP	*Internet Service Provider*
MAC address	*Media Access Control address*
Mbps	*Mega bits per second*
MITM	*Man-In-The-Middle*
MTU	*Maximum transmission unit*
PPGINF	Programa de Pós-Graduação em Informática
P2P	*Peer-to-peer*
SOCKS	Protocolo *Socket Secure*
SOM	*Self Organizing Map*
SSH	*Secure Shell*
SVM	*Support Vector Machine*
Tbps	*Tera bits per second*
TCP	*Transmission Control Protocol*
TTL	*Time To Live*
UDP	*User Datagram Protocol*
UFPR	Universidade Federal do Paraná
VPN	*Virtual Private Network*

SUMÁRIO

1 INTRODUÇÃO

O uso da Internet tem crescido e vem sendo utilizada por cada vez mais pessoas. Um levantamento feito pela empresa Cisco (Cisco, 2020) mostra que, atualmente, quase 60% da população mundial está conectada a esta rede (aproximadamente 4,5 bilhões de pessoas). Além disso, é esperado um aumento de 6% ao ano, até 2023, quando 5,3 bilhões de pessoas estarão conectadas à Internet. Em meio a isso, ocorre também a popularização da Internet das Coisas (*Internet of Things* - IoT), em que objetos físicos cotidianos, com sistemas embarcados, são capazes de coletar informações sobre o meio, processá-las e transmiti-las. O meio de comunicação da IoT será, também, através da infraestrutura de Internet.

Inerente à evolução tecnológica, as velocidades de conexão também estão melhorando, propiciando maiores velocidades de acesso para os diversos dispositivos. Em 2020, a velocidade média de conexão (entre as diferentes tecnologias) foi próxima aos 20 Mbps. Em 2023, a expectativa é que os dispositivos sejam capazes de se comunicar com uma velocidade média de 44 Mbps (Cisco, 2020). Isto decorrerá da implantação de novas redes de infraestrutura, além de novas tecnologias, como o 5G (a quinta geração de padrão de tecnologia para acesso às redes de celulares). Assim, tem-se um cenário com o aumento do número de dispositivos e um aumento da velocidade de acesso destes. Prevê-se um aumento de 27% ao ano do volume de tráfego até 2022, efetivamente triplicando o volume em 5 anos (Cisco, 2018).

O acesso à Internet chega a cada vez mais pessoas e a tendência é de aumento. Dada a facilidade de acesso que é propiciada, as relações sociais e empresariais estão se moldando a este novo tipo de conexão. Diversos serviços e comportamentos estão migrando para a Internet, como bancos, compra e venda de produtos (*e-commerce*), relações inter-pessoais, notícias, saúde, lazer, estudos, entre outros (Atkin et al., 1998; AbuKhousa et al., 2012). Embora o acesso a Internet esteja se popularizando e se tornando mais fácil, evoluindo com o tempo, os dispositivos que a acessam não são atualizados com a mesma frequência. Majoritariamente, o acesso a Internet é feito por celulares *smartphones* (cerca de 41% do tráfego, em 2023, será por este tipo de dispositivo) (Cisco, 2020). E muitos destes celulares deixam de receber suporte do seu fabricante não muito tempo após serem lançados, deixando-os suscetíveis a vulnerabilidades de segurança que venham a ser descobertas após o fim do suporte (Which, 2020). Os dispositivos IoT, que corresponderão em 2023 a 34% do volume de tráfego, também são propensos a vulnerabilidades pois possuem baixa capacidade de armazenamento e processamento, limitando a possibilidade de correção de erros em seus códigos e sua atualização - quando não deixadas sem suporte pelos fabricantes, como ocorre com os *smartphones*.

O aumento da diversificação dos serviços oferecidos na Internet – assim como acontece fora dela – atrai agentes maliciosos e criminosos que tentam se utilizar de falhas de segurança e/ou força bruta para tirar proveito destes sistemas computacionais em benefício próprio em detrimento de outrem. Com a digitalização de (quase) tudo, a informação passa a ser uma *commodity* extremamente valiosa e é visada por agentes maliciosos. Estes agentes maliciosos buscam afetar um sistema computacional de três maneiras: tornando os dados inelegíveis para o seu possuidor (integridade dos dados), tornando os sistemas inacessíveis para acessos legítimos (disponibilidade de serviços) e com o roubo de dados, deixando os dados confidenciais disponíveis para acessos ilegítimos (confidencialidade) (Samonas e Coss, 2014).

O ataque à integridade dos dados se dá quando um *bot* consegue acesso a um dispositivo vulnerável e consegue tornar os dados inelegíveis, através de criptografia, com o uso, por exemplo, de *ransonwares* ou, inclusive apenas destruindo os dados presentes no dispositivo. Já o ataque

à confidencialidade dos dados se dá quando o *bot* é capaz de extrair informações sigilosas e enviá-las a um dispositivo externo, que pode torná-las públicas ou revendê-las, por exemplo. No ataque a disponibilidade, o *bot* deve ser capaz de tornar o serviço inacessível. Para tanto, ele pode ter acesso ao dispositivo e interromper a execução do serviço desejado ou realizar uma sobrecarga de requisições ao serviço que pretende interromper, através de solicitações que pareçam legítimas. Vale notar que os ataques à integridade e à confidencialidade necessariamente exigem um acesso indevido aos dispositivos pelo *bot*, enquanto o ataque à disponibilidade pode ser realizada por dispositivos terceiros, infectados, que realizam requisições ao serviço, que se parecem legítimos, sobrecarregando-o e tornando-o inacessível. Este ataque é conhecido com Ataque de Negação de Serviço Distribuído (DDoS)

1.1 MOTIVAÇÃO

Agentes maliciosos dispõem de diversas ferramentas e artifícios para gerar e executar os mais variados tipos de ataques na Internet. Os ataques podem ocorrer de maneira passiva ou ativa. Na forma passiva, há a captura de informações e monitoramento da rede sem interferir no funcionamento adequado dos sistemas, com a vigilância da rede e o uso de *keyloggers* (aplicativos que monitoram e armazenam todas as entradas do teclado). Já no ataque ativo, o agente interfere e manipula o comportamento e a operação de sistemas para alcançar o seu objetivo (afetar a disponibilidade, a confidencialidade e/ou a integridade do sistema). Alguns exemplos de ataques ativos são ataques de negação de serviço, *phishing* e ataques na rede, como os ataques *man-in-the-middle* (MITM) e *masquerade*, quando uma entidade busca se passar por outra.

Diversos tipos de ataques se beneficiam das vulnerabilidades de dispositivos e os infectam com algum código malicioso para se utilizar dos dispositivos e seus recursos (processamento, armazenamento e/ou conexão de rede) na realização de ataques. Uma vez infectado, um dispositivo passa a realizar ações indesejáveis e não controladas por seu possuidor, se tornando um *bot*. Estes dispositivos infectados (*bots*) podem ser controlados por um ou mais mestres (*botmaster*) que os comandam e conduzem diversos tipos de ataque, tais como o ataque de negação de serviço distribuído (*Distributed Denial of Service* – DDoS). Além disso, diferentes ataques podem ser combinados, como o uso de *bots* para o envio de *spam*, em campanhas de *phishing*, enquanto, também, realiza o roubo de dados, por exemplo.

Botnets representam uma – talvez a maior – ameaça à segurança das redes de computadores e seus usuários. Elas podem, por exemplo, interromper seus serviços através da coordenação de uma quantidade massiva de dispositivos infectados (*bots*), auxiliar e executar um ataque de roubo de dados e envio de *spam*. Levantamento feito pela Accenture, mostra que o prejuízo global estimado devido a crimes cibernéticos no período 2018-2023 supera US$5 trilhões. Para uma única empresa que é vítima desses ataques, o custo de ataques por *botnets* é, em média, de aproximadamente 400 mil dólares (Bissell et al., 2019). Buscando mitigar os ataques e evolução destes *bots*, diversos estudos científicos já foram publicados buscando maneiras de frear ou eliminar esta ameaça. Em resposta, os desenvolvedores e operadores das *botnets* se tornaram mais agressivos e ofensivos, com o uso de criptografia em suas comunicações, arquiteturas descentralizadas e maneiras de não evasão do seu ataque.

As *botnets* se utilizam da infraestrutura de rede pré-existente e formam sobre ela uma rede de comunicação entre os *bots*. Esta rede de comunicação sobreposta à rede tradicional é utilizada pelo atacante para controlar os *bots*, de forma a consolidar seu ataque. A detecção de *botnets* é um grande desafio, pois o conjunto de dados tratado tende a ser massivo. Um exemplo relativamente recente é o caso da *botnet* Mirai, que em 2016 infectou milhares de dispositivos da Internet das Coisas e desempenhou ataques de negação de serviço distribuído cujos tráfegos

atingiram a marca de 1.1 Tbps (Kolias et al., 2017). Desse modo, diante da quantidade massiva de dados, é importante buscar identificar quais características do tráfego são relevantes para distinguir dispositivos maliciosos de benignos. Assim, diante da complexidade e a evolução contínua destes ataques, há uma crescente necessidade de prover soluções de segurança que sejam capazes de detectar a presença de *bots* e identificá-los de maneira eficaz, buscando mitigar e/ou prevenir os efeitos de ataques de *botnets*.

1.2 PROBLEMA

Com a evolução das *botnets*, identificar características relevantes do tráfego se faz necessário para uma detecção eficaz. A análise do tráfego, realizando apenas a leitura do conteúdo do que está sendo trafegado (para identificação de um ataque), pode não ser eficaz ou até impossível de ser realizada (por exemplo, devido ao uso de criptografia na comunicação dos *bots*). Entretanto, a interação entre o grupo de *bots* e seus alvos sendo atacados é notada na rede, através do tráfego gerado, podendo ser extraídas informações ou características deste tráfego (Karim et al., 2014). Uma *botnet* pode se utilizar de uma arquitetura centralizada de controle e comando, onde um nó da rede dispara ordens e os outros nós executam as ordens recebidas (*bots*). Se faz necessária a identificação de quem está comandando os *bots* para a consequente derrubada da *botnet*. Pode-se buscar verificar, no tráfego, uma maior quantidade de pacotes se dirigindo ao servidor de comando e controle, comandado pelo *botmaster*, por exemplo. Entretanto, outra *botnet* pode se utilizar de uma arquitetura descentralizada, sobreposta a uma rede *Peer-to-Peer* (P2P), impondo a necessidade de verificar características diferentes no tráfego (que antes seriam irrelevantes no primeiro cenário, centralizado). Além disso, o atacante pode mascarar a comunicação, utilizando o protocolo *HTTP* (*Hypertext Transfer Protocol* – comum para requisições em navegadores da Internet) para encaminhar mensagens de conteúdo real SSH (utilizado para comando de servidores), já que o protocolo HTTP é raramente bloqueado por *firewalls* (Iglesias e Zseby, 2015), já que ele é utilizado por navegadores (*browsers)* da Internet.

De qualquer forma, mesmo havendo diferenças no modo como as *botnets* são construídas, elas ainda se utilizam da infraestrutura de rede comum e a Internet (Karasaridis et al., 2007). Dessa forma, deve seguir as regras pré-definidas nestas redes, tais como a utilização de endereços (MAC, IP), protocolos existentes (como DNS, ICMP, TCP e UDP) e até aplicações conhecidas. Ao se posicionar como um agente da rede, o mecanismo de análise e detecção pode extrair informações do tráfego destes agentes e buscar inferir relações que propiciem a detecção de agentes maliciosos na rede. Assim, o grande desafio é identificar estes *bots*, visando minimizar a sua ação. Além disso, há um grande fator dificultador devido ao volume de tráfego da rede, grande quantidade de nós da rede[1] e da limitação de recursos, como poder de processamento, memória e largura de banda de vários dispositivos que compõem estas redes.

Além disso, ataques de negação de serviço distribuídos sobrecarregam sistemas, tornando-os indisponíveis para acessos legítimos. Atualmente, realizar disparos desse tipo de ataque é relativamente fácil, até sendo possível contratar redes de *botnets* geradoras de DDoS por algumas dezenas de dólares. Diante da facilidade que é possível atacar redes e os danos causados pela indisponibilidade dos sistemas, rapidamente identificar os agentes maliciosos propicia um auxílio na mitigação dos danos, quando feito de maneira ágil e precisa.

[1]Nó de rede: um dispositivo conectado a uma rede de computadores capaz de se comunicar com outros nós da rede

1.3 OBJETIVOS E CONTRIBUIÇÕES

Diante do problema gerado por ataques de *botnets*, tem-se por objetivo detectar estas *botnets* de maneira precisa, diante de um volume massivo de dados, antecipando ataques de negação de serviço distribuído. Note que, na presente proposta, não se busca interromper este ataque, mas sim identificar os agentes maliciosos e reportá-los a outros sistemas ou administradores de rede para que possam, então, atuar na rede, interrompendo ou atenuando os efeitos dos ataques. Assim, a arquitetura tem por objetivo identificar antecipadamente ataques DDoS, de maneira distribuída, enquanto, de maneira centralizada, busca-se identificar os agentes maliciosos causadores dos ataques de negação de serviço distribuídos.

Como contribuição deste trabalho, pode-se destacar a extração de características híbridas para a análise do tráfego, provenientes tanto dos cabeçalhos do tráfego quanto de um grafo formado pelos pacotes trocados. Estas características ampliam o leque de informações que podem auxiliar na distinção do comportamento de nós maliciosos e benignos. Também, como diferencial, pode-se ressaltar a arquitetura distribuída para identificação precoce de ataques DDoS, sendo que a identificação destes ataques é feita em redes locais, que podem rodar em dispositivos com poucos recursos computacionais, como roteadores domésticos. O tráfego das redes locais, assim, é agrupado em um nó centralizador (de um provedor de Internet, por exemplo) para identificação dos *bots*. Por fim, a análise para detectar os agentes maliciosos se dá através de um método determinístico, capaz de analisar a causalidade do comportamento de um nó, devido a ação de outro nó, tal como um *bot* atacando um dispositivo computacional.

Além do descrito acima, uma seleção de características é realizada, contribuindo com a redução do volume de dados a serem analisados sem prejudicar a análise. Ela é feita de modo que características que sejam dependentes umas das outras não sejam utilizadas na análise, validada por um teste estatístico. Esta seleção permite que informações que sejam relevantes para a detecção sejam destacadas das demais. Diversos trabalhos se utilizam de características do tráfego, sem uma análise da relevância dessas características. Também, com o intuito de acelerar a análise e, consequentemente a detecção dos *bots*, uma clusterização é realizada, agrupando nós com comportamento similar, através de uma técnica de aprendizagem de máquina não-supervisionada. Essa clusterização permite que uma análise paralela de cada *cluster* seja realizada, além de auxiliar na detecção dos *bots*.

Portanto, a proposta permite trabalhar em uma escala massiva de dados, utilizando uma arquitetura distribuída para: identificação precoce de possíveis ataques, realizando a detecção dos *bots* apenas nas redes em que um possível ataque seja identificado; selecionar as características mais relevantes; agrupar os nós em *clusters*, diminuindo o volume de dados analisados entre si na análise causal. Estas abordagens permitem que a análise seja realizada de maneira confiável, precisa e capaz de operar em um grande volume de dados, como os de ataques DDoS, possibilitando uma detecção dos *bots* e dos agentes maliciosos.

1.4 ESTRUTURA DA DISSERTAÇÃO

Este trabalho está organizado no formato apresentado a seguir. O Capítulo 2 apresenta os fundamentos que sustentam a proposta e os subsídios para entendimento do exposto nos capítulos seguintes. O Capítulo 3 discute os trabalhos relacionados e as pesquisas que fundamentam a proposta do BotFetcher. O Capítulo 4 descreve o mecanismo de predição, assim como método de detecção de *botnets*, dividindo-o em três etapas. O Capítulo 5 descreve a metodologia de avaliação e os resultados. Por fim, o Capítulo 6 conclui o trabalho.

2 FUNDAMENTOS

Neste capítulo, são apresentados os fundamentos que subsidiam a proposta (Capítulo 4) e colaboram com o entendimento dos Trabalhos Relacionados (Capítulo 3). Na Seção 2.1, são apresentadas as *botnets*, como operam para aumentar sua rede de *bots*, como é sua comunicação e para que são utilizadas. Na Seção 2.2 são apresentadas as características do tráfego, expondo como ele é estruturado, protocolos utilizados e como podemos extrair e inferir informações a partir do tráfego. Também são mostradas maneiras de selecionar as características para serem utilizadas em aprendizado de máquina. Na Seção 2.3 apresenta-se os indicadores estatísticos utilizados para inferir um possível ataque DDoS de maneira precoce. Também, apresenta-se os fundamentos e técnicas de aprendizado de máquina não supervisionado e o que pretende-se obter como resultado de sua aplicação. Por fim, se detalha como podemos inferir correlações causais em dados não estruturados, que possibilitam a detecção de *botnets*.

2.1 BOTNETS

Uma *botnet* é uma coleção de dispositivos computacionais conectados à Internet que foram comprometidos e são controlados por atacantes para propósitos maliciosos e/ou ilegais. O termo se origina dos *softwares* utilizados (chamados de robôs) pelo seu comportamento automatizado (do inglês, *robots* – ou *bots* na forma curta) (Wang e Ramsbrock, 2013). O *software* de um *bot* é um *malware* altamente evoluído para a Internet, incorporando componentes de vírus, *spyware* e outros *softwares* maliciosos. Diferentemente dos vírus anteriormente conhecidos, a motivação para operar uma *botnet* é financeiro. *Botnets* são extremamente lucrativas, podendo gerar para seus operadores centenas de milhares de dólares por dia. Os operadores, conhecidos como *botmasters*, podem alugar o processamento da sua *botnet* ou gerar lucro diretamente, pelo envio de *spam*, distribuindo *spyware* que auxiliam no roubo de dados e até extorquindo dinheiro pela ameaça de ataques de negação de serviço.

Bots e *botnets* são a evolução dos *malwares* para a Internet. Os *bots* são dispositivos conectados à Internet que foram infectados por um *software* executável malicioso. Estes dispositivos comprometidos (*bots*) usualmente se comunicam com um servidor de comando e controle, responsável pelos disparos de ordens e instruções de ações para os *bots*. O objetivo de uma *botnet* é realizar ataques ou atividades maliciosas em nome de seu operador. O operador da *botnet* é denominado *botmaster* e possui o controle do servidor de comando e controle (servidor C&C). Para evitar a detecção de sua ação, a comunicação de um *botmaster* com um servidor C&C pode passar por diversos *proxies* até chegar ao *botmaster*. Diferentes *botnets* não possuem as mesmas características e formas de atuação e, por serem códigos programáveis, são atualizadas e evoluem ao longo do tempo. Elas podem diferir na maneira como encontram e infectam os dispositivos a serem comprometidos, pela maneira como se comunicam com o servidor C&C, pelo seu propósito e sua topologia, combinando diferentes mecanismos de cada característica. A seguir são apresentados diversos aspectos que propiciam o entendimento e a caracterização das *botnets*, detalhando e comparando características das informações apresentadas.

2.1.1 Ciclo de Vida

O primeiro passo no ciclo de vida de uma *botnet* é a sua criação. A criação consiste no desenvolvimento do *software* do *bot*, recorrentemente utilizando-se de códigos previamente

existentes, com a adição de novas funcionalidades. Diversas linguagens de programação, metodologias, técnicas e formas podem ser utilizadas no desenvolvimento. Uma vez desenvolvido o *software*, o código é compilado e um programa executável (*bot binary*) será o vetor de infecção para as máquinas comprometidas. Do ponto de vista de um *bot*, uma infecção começa com a execução do executável binário no dispositivo da vítima.

O executável compilado é transferido para o computador da vítima através de um mecanismo de propagação (Subseção 2.1.2). Após a infecção, em sua execução, o *bot* busca se comunicar com o servidor de comando e controle (servidor C&C) em um processo conhecido como *rallying* (Subseção 2.1.3). Através deste processo, o *bot* estabelece um canal de comunicação entre ele e o servidor C&C, recebendo atualizações e comandos. Baseado na maneira como a comunicação é estabelecida, também podemos caracterizar a sua topologia.

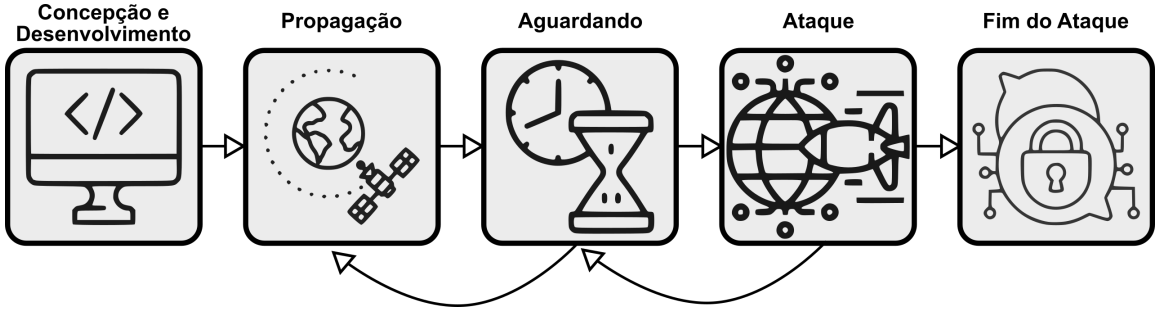


Figura 2.1: Ciclo de Vida de uma *Botnet*

Os *bots* recém recrutados aguardam por instruções do servidor C&C para que realizem o real propósito (Subseção 2.1.4) da *botnet*. Enquanto aguardam o início da execução do ataque, podem, também, buscar novos dispositivos vulneráveis para infectá-los. Espera-se que as *botnets* não sejam percebidas durante o seu ciclo de vida. Portanto, o uso de mecanismos de evasão (Subseção 2.1.5) são normalmente implementados de maneira a evitar a detecção da infecção e a identificação dos agentes responsáveis pelo ataque.

2.1.2 Propagação

Os *bots* têm como um dos seus principais objetivos aumentar a rede de nós comprometidos, infectando outros dispositivos, crescendo o número de *bots* e a *botnet*. Muitos dos executáveis distribuídos têm mecanismos internos que facilitam a disseminação e contaminação de outros dispositivos. Dependendo da quantidade de interação humana necessária para realizar uma nova infecção, podemos classificar o método de propagação como ativo ou passivo.

No modo passivo, é necessária alguma interação do usuário da máquina que será infectada. O atacante armazena o binário do *bot* em um servidor remoto e o usuário realiza o acesso, através de um navegador, por exemplo. Quando o usuário acessa a página, o executável é instalado se utilizando de falhas do navegador, *plug-ins* ou de erros cometidos pelo usuário, que desconhece o que está instalando. Outro modo passivo de ataque é a utilização de Cavalo-de-Tróia (*Trojan horse*): o atacante fornece o *software* malicioso junto a um *software* legítimo e, quando instalado pelo usuário, ambas as funcionalidades residem no dispositivo: a legítima (como um jogo ou protetor de tela) e o *bot*. Também, pode-se realizar o ataque passivo através de mídias infectadas, como *pen-drives* ou CDs. O envio de arquivos por mídia eletrônicas como correio eletrônico (*e-mail*) e mensagens instantâneas também é um vetor de ataque significativo.

O uso de ataques de engenharia social também é um vetor de ataque perigoso e prevalente. Em outros métodos, busca-se o aumento do número de *bots* em massa, sem considerar quem

é seu alvo, apenas aumentando o número de dispositivos infectados. Na engenharia social o alvo é conhecido e o ataque é direcionado, sendo os danos potenciais ainda mais desastrosos para a vítima, pois, o atacante conhece e entende o valor que pode ser extraído do ataque à vítima. Caso a *botnet* esteja articulada para realizar um roubo de dados, o valor da informação já é de conhecimento do atacante, no ataque em conjunto com a engenharia social. Assim, busca-se convencer a vítima a realizar o *download* do binário do *bot* e executá-lo na máquina a ser comprometida, utilizando artimanhas sociais, como, por exemplo, se identificando como o suporte de informática da empresa e convencendo a vítima a compartilhar credenciais ou executar comandos que propiciem o acesso ilegítimo a rede ou ao dispositivo computacional.

Na forma ativa de propagação, o atacante continuamente busca por dispositivos vulneráveis que estejam conectados à rede. Essa busca pode ser feita por *bots* que tentam atacar vulnerabilidades conhecidas em dispositivos e tentam explorá-los. Uma vez conquistado o acesso, o *software* do *bot* é instalado e busca-se a persistência no sistema, com o intento de ganhar privilégio administrativo do dispositivo. Diversas ferramentas permitem que uma varredura completa na rede seja feita, identificando dispositivos presentes nela, com portas abertas a acessos e que possivelmente possuam vulnerabilidades. Sendo executado o binário após obtido o acesso, o *bot* buscará contactar o servidor de C&C, como é feito também na forma passiva de propagação.

2.1.3 *Rallying* (Descoberta) e Comunicação

Através do *rallying*, os *bots* descobrem o servidor de comando e controle para que a comunicação seja estabelecida. A partir desta descoberta, o *bot* se torna disponível para a *botnet*. Por se utilizar de protocolos de comunicação conhecidos e acessíveis pela Internet, os servidores de comando e controle são encontrados pelos *bots* através do seu endereço IP. Assim, o primeiro objetivo do *bot* é saber com qual endereço IP se comunicar. Para isto, durante o desenvolvimento do *software bot*, alguma informação deve ser passada no código de como o *bot* pode encontrar o servidor C&C. Essa informação disponibilizada pode conter o endereço final do servidor C&C ou apenas um ponto de partida para a obtenção do endereço deste servidor. Os endereços IP também podem ser disponibilizados para os *bots* através do uso de nomes de domínio. No caso de uso de domínios, o *bot* possui previamente um conjunto de potenciais domínios a serem buscados ou o domínio direto para o servidor C&C.

O endereço IP do servidor pode ser imutável, acessível e alterável apenas pelo código – o mesmo vale para os domínios. Nesse modelo "*hardcoded*", o uso de engenharia reversa do arquivo executável do *bot* poderia identificar os endereços dos servidores C&C e interromper um ataque, através do bloqueio ao acesso a estes IPs ou domínios por um administrador de rede. Embora o uso direto do endereço IP elimine o uso de DNS, o uso de endereços e domínios codificados é um método primitivo de obter o endereço IP do servidor C&C. Uma maneira de driblar o uso de endereços e domínios expressos diretamente no código é com o uso de endereços e domínios gerados durante a execução. Embora o uso diretamente de endereços IPs gerados dinamicamente não tenha sido observado em casos reais, as mentes por trás das *botnets* poderiam gerar uma série de IPs para potenciais servidores C&C, principalmente se o *botmaster* possuir um bloco de endereços IPs ao invés de endereços esparsos.

Para encontrar os endereços dos servidores C&C dinamicamente, usualmente se utilizam nomes de domínio. O uso de domínios facilita a operação das *botnets* pois pode-se manter uma gama de servidores C&C para cada domínio. O servidor de domínio retorna para o *bot* o endereço IP em que o servidor C&C se encontra no momento e sua atualização pode ser frequente e rápida. *Botnets* e *bots* podem gerar domínios por algoritmos conhecidos apenas por eles. Derrubar um domínio é um trabalho penoso de maneira legal, cheio de formalidades. Quando um domínio é derrubado, a *botnet* é capaz de migrar sua operação para outro domínio. Deve-se notar, porém,

que mesmo nessa situação, alguma informação no momento da codificação deve ser passada ao *bot* para iniciar a busca, usualmente um conjunto de domínios. Porém, o uso de domínios algoritmicamente gerados pode tornar a detecção destes para uso escusos mais rápida. Estes domínios dificilmente terão nomes que são inteligíveis para um ser humano, mais se aproximando da junção de caracteres randômicos. Nesse sentido, (Nadler et al., 2019) mostrou ser capaz de detectar domínios que não seriam utilizados por humanos mas para roubo de dados.

Por fim, algumas *botnets* operando em uma topologia P2P (*Peer-to-Peer*) possuem codificado não o único endereço do servidor C&C mas um conjunto de pares desta rede. Estes pares (que, também, podem ser encontrados por domínios) mantém uma lista de outros pares que estão ativos e são regularmente atualizados. Dessa forma, encontrar o servidor C&C pode se tornar mais fácil para o *bot*, haja visto que a rede é capaz de se manter mesmo com a saída de algum nó e não se sabe, para um observador externo, qual dos integrantes da rede que é ou não mantenedor da informação, dificultando o seu bloqueio ou interrupção de seu funcionamento.

Os *bots* se não se comunicarem com o servidor C&C são apenas um conjunto de *malwares* espalhados em diversas máquinas infectadas. A comunicação entre o servidor C&C e os *bots* é que possibilita a operação da *botnet*. Esta comunicação pode se utilizar de protocolos conhecidos, já consolidados e menos propensos a *bugs* ou protocolos novos, utilizando de mecanismos próprios para identificar os comandos a serem executados e as atribuições do *bot*. No início das *botnets*, o IRC era utilizado para troca de dados na *botnet*. É um protocolo amplamente utilizado na Internet, baseado em texto e sintaxe simples, com uma baixa latência. Porém, o uso do IRC não é comum em redes empresariais, o que facilita a sua detecção nestas redes. Com a evolução de novos protocolos, o IRC passou a ser substituído por comunicação HTTP. Do ponto de vista de defesa, bloquear a comunicação HTTP é impraticável pois toda a navegação através de navegadores utiliza esse protocolo. Periodicamente, os servidores C&C disponibilizam novos comandos nestes servidores HTTP e os *bots* realizam a captura destes.

O uso dos protocolos IRC e do HTTP provêm uma solução centralizada para a comunicação. Caso haja um número massivo de *bots*, o *botmaster* deve inclusive se preocupar e buscar uma maneira de escalonar essa comunicação para que o servidor de comando e controle não seja sobrecarregado. As redes P2P possibilitam a comunicação descentralizada, de difícil detecção e a informação pode ser disseminada e mantida por qualquer nó desta rede. Dessa forma, notou-se uma evolução das *botnets* para esta comunicação descentralizada que possibilita uma maior resiliência da informação sendo transmitida. Mais recentemente, diante da explosão de serviços que possibilitam a divulgação de conteúdo (como o Twitter), notou-se uma nova vertente de *botnets* se utilizando destes serviços para seus ataques. Os perfis nestes serviços são públicos e um novo comando é transmitido através de um novo "tweet", por exemplo. O sucesso dessa comunicação é mantido enquanto os atacantes conseguem manter seus perfis ativos. Notadamente, essa comunicação se utiliza de protocolos HTTP, porém o servidor C&C não é mantido pelos atacantes e o bloqueio para sites legítimos é mais improvável.

A topologia indica como os *bots* se comunicam entre si e com o servidor C&C. Esta comunicação cria uma rede sobreposta à Internet. Os modelos de comunicação podem ser centralizados, descentralizados ou híbridos. No modelo centralizado, todos os *bots* recebem comandos de um único servidor de comando e controle. É uma implementação simples e fácil, apresentando, porém, um ponto de falha único para toda a *botnet*. Neste modelo centralizado, em estrela, o *botmaster* emite os comandos diretamente no servidor C&C, com rápida velocidade de comunicação com os *bots*. No modelo centralizado hierárquico, com o intuito de tentar esconder o *botmaster*, diversas camadas de *proxies* são incorporadas entre o *botmaster* e o servidor de C&C e entre os *bots* e o servidor C&C. Estes *proxies* usualmente são máquinas comprometidas

compondo a própria *botnet*. Dessa forma, a análise das atividades da *botnet* fica mais complicada, com prejuízo para a latência da comunicação.

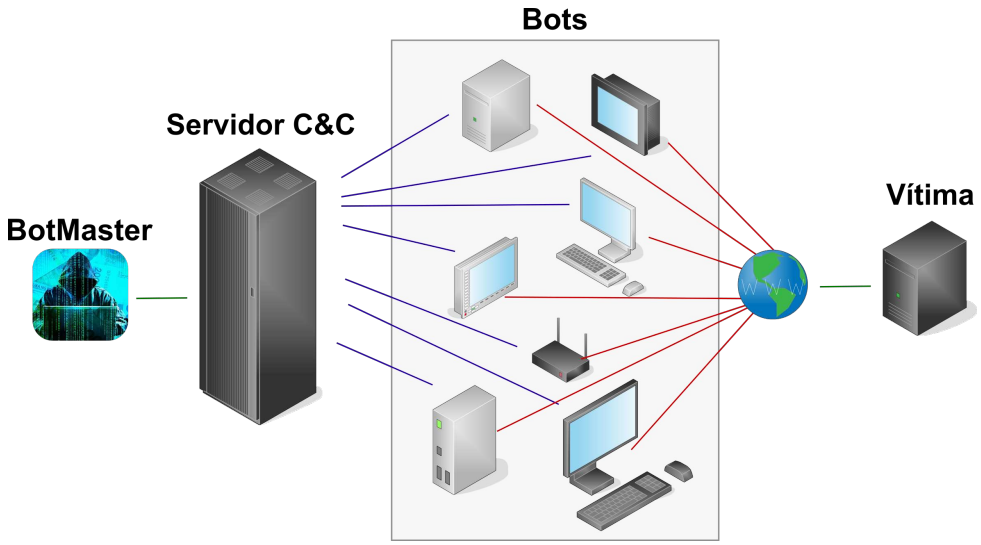


Figura 2.2: Típica *Botnet* de Ataque de Negação de Serviço Distribuído

No modelo descentralizado, não há um único dispositivo responsável para encaminhar os comandos para os *bots*. A gerência dos *bots* é distribuída entre múltiplos servidores C&C ou possuem uma arquitetura aleatória, sem uma relação clara de mestre-escravo. No modelo distribuído, vários servidores se comunicam entre si e cada um controla um conjunto fixo ou variável de *bots*. A carga de comunicação fica reduzida neste modelo, aumentando a resiliência da rede. Distribuindo estes servidores geograficamente, ao redor da Terra, melhora-se a redundância e disponibilidade, reduzindo a latência na comunicação com *bots* mais próximos. Entretanto, nota-se uma crescente complexidade quando comparada com o modelo centralizado. Já no modelo descentralizado aleatório, típico de redes P2P, não há uma relação entre mestre e escravo entre os pares. Os *bots* são capazes de receber instruções e comandos de outros *bots* e, também, enviar comandos. Assim, o *botmaster* envia os comandos a um dispositivo pertencente a rede e não há uma rota pré-determinada entre ele e os *bots*. Este tráfego sendo roteado de maneira não pré-determinada dificulta a localização do *botmaster* e a derrubada da *botnet*, com o contraponto de que a latência entre a emissão dos comandos e sua atuação é imprevisível.

De qualquer maneira, diversos serviços e protocolos são utilizados pelos atacantes para encontrar os comandos necessários para realizar o ataque. As *botnets* são um alvo em movimento e precisam, portanto, de uma solução capaz de identificar suas ações e agentes, não utilizando apenas os protocolos de comunicação mas por diversas outras características. Estas características, que possibilitam a identificação da forma de comunicação, serão discutidas posteriormente e fundamentam a proposta do BotFetcher.

2.1.4 Propósito

Caso apenas infectem novos dispositivos (aumentando o tamanho da *botnet*) sem realizar as ações necessárias para completar um ataque, os *bots* não têm função prática para o seu controlador. O ataque é a atividade maliciosa que dá propósito a existência da *botnet*. Pode-se depreender diversos propósitos para a existência das *botnets*. O primeiro e mais comum é o ganho financeiro do seu operador. Porém, propósitos específicos podem ser observados, tais como o roubo de dados, espionagem e guerra cibernética. Vale ressaltar que o propósito pode não ser único e, também, mudar de acordo com os comandos efetivados pelo *botmaster*.

Por se utilizar de dispositivos infectados, pertencentes a usuários inocentes, distribuídos ao redor do globo, a probabilidade de encontrar o *botmaster* realizando um ataque é reduzida. Uma maneira do operador lucrar com sua rede de *bots* é utilizando o poder de processamento e armazenamento destes dispositivos e alugá-los a terceiros para distribuição de arquivos ou computação distribuída, como para a quebra de senhas. Algumas *botnets* podem, inclusive, se servir da rede e armazenamento dos dispositivos para realizar a distribuição de outros *malwares*, servindo como uma plataforma de distribuição para terceiros, tais como as botnets *ZeuS* e *Pushdo* (*Cutwail*) (Business, 2010; SecureWorks, 2010).

O envio de *spam* também é realizado por *botnets*. O *botmaster* disponibiliza um *template* de *e-mail* para os *bots* que enviam *e-mails* baseados nestes *templates*. Este tipo de *e-mail* indesejado, normalmente com envio de promoções e propaganda já se mostrou mais efetivo e mais barato que ações tradicionais de marketing. Porém, foi abusado por muito tempo e diversas ações foram tomadas para restringir o seu volume e propagação. De qualquer forma, com as *botnets*, o envio de *spam* encontrou uma alternativa para propagar seu alcance. Entre os mais conhecidos estão *Rustock*, *Pushdo*, *Bagle*, *Bobax*, *MegaD* e *Virut* (Security, 2010).

Informações podem ser coletadas das redes e dos dispositivos para ganhos financeiros. Os *bots* podem coletar credenciais, senhas e informações das máquinas infectadas, tais como números de cartão de crédito e dados bancários. Além disso, o roubo de informações por um adversário (empresa ou nação, por exemplo) pode ser altamente lucrativo, focando em um alvo específico. Normalmente, ataques direcionados (e não em massa) são bem financiados, possuem mais conhecimento do alvo pelos seus operadores e são organizados, se utilizando das mais avançadas técnicas de intrusão. As *botnets Aurora* (Post, 2010) e *GhostNet* (Group, 2009) são conhecidas por suas ações de espionagem e coleta de informações.

Com a mudança de comportamento para o mundo digital, nações e corporações passaram a operar neste novo paradigma. A guerra cibernética já é uma realidade. Normalmente envolve uma nação que tem interesse em derrubar ou prejudicar a operação de ativos em outra nação. Atacando alvos estratégicos, como empresas de energia elétrica, tratamento de água e complexos industriais, estes ataques podem prejudicar a economia de outras nações de maneira significativa para que os países ganhem vantagem competitiva sobre outro. Usualmente, é um ataque altamente especializado e sofisticado, direcionado exclusivamente a um alvo específico. A guerra cibernética se tornou tão relevante que empresas seguradoras deixaram de cobrir custos de perda gerada, caso seja identificada a atuação de uma nação, considerando a ação como um ato de guerra (Times, 2019). O primeiro caso de uso de uma arma digital foi a *Stuxnet*, com o intuito de atingir centrífugas de enriquecimento de urânio das usinas do Irã (Wired, 2014).

Fraudes também podem ser realizadas por *botnets* para trazer ganhos financeiros a seus operadores. Uma forma de fraude é o *phishing*, no qual a vítima é redirecionada a uma página controlada pelo atacante ao invés da página legítima. O *bot*, presente na rede da vítima, é capaz de redirecionar o acesso legítimo para uma página falsa, onde informações pessoais e dados de cartão de crédito podem ser inseridos e serão armazenadas em servidores dos *botmasters*. Outra forma de fraude é o direcionamento de *bots* para influir em resultados de enquetes e jogos online. Também, fraudes em cliques podem ser realizadas, direcionando os *bots* para clicar ilegitimamente em anúncios que pagam no formato *"pay-per-click"* (Security, 2006).

Os *bots*, quando combinados, possuem a sua disposição um grande poder computacional, além de uma imensa banda disponível nas diversas redes em que estão presentes. Dessa forma, ao direcionar os *bots* para um determinado serviço e exaustivamente realizar solicitações de acesso a este serviço, o *botmaster* é capaz de exaurir os recursos do serviço e impossibilitar o acesso legítimos aos usuários. Esse tipo de ataque é conhecido como ataque de negação de serviço distribuído (*Distributed Denial-of-Service* (DDoS)). A extorsão também faz parte do

leque de possíveis ataques dos *bots*. Diferentemente do mundo físico, a extorsão é requisitada para que um determinado serviço não seja atacado. Normalmente, um ataque inicial é feito a um serviço mostrando o potencial de ataque. Após isso, uma extorsão é exigida para que o ataque não prossiga. Grandes empresas preferem pagar o valor da extorsão do que perder vendas e credibilidade. Além disso, o sequestro de dados também pode ser lançado por *bots*. Nesse caso, os dados da máquina infectada são criptografados e uma quantia é exigida para que a chave para descriptografar os dados seja fornecida. O *WannaCry* fez com que diversas companhias e usuários tivessem seus dispositivos infectados em um curto espaço de tempo, interrompendo operações de indústrias, serviços e até hospitais (Rapid7, 2017).

2.1.5 Evasão

As manobras de evasão permitem a persistência e continuidade da operação da *botnet*, buscando se esconder das diversas técnicas de detecção. Caso nenhuma tentativa de evadir a detecção fosse realizada, a detecção do *malware*, que compromete o dispositivo alvo, seria facilmente realizada por aplicativos de segurança da máquina hospedeira e, consequentemente, eliminando os *malwares*, antes que um ataque seja realizado. O *botmaster* também poderia ser identificado caso não tentasse esconder as suas ações de controle dos *bots*, possivelmente levando-o a responder criminalmente por seus feitos.

Para que os *bots* fiquem disponíveis pelo maior período de tempo para seu *botmaster*, eles precisam evitar a detecção pelos dispositivos infectados. Dessa forma, a primeira preocupação é com *softwares* de segurança que analisem os binários que são distribuídos para as máquinas infectadas. A maior parte dessas aplicações (anti-vírus) realizam a detecção através de reconhecimento de padrões nos binários. Para ofuscar os binários, o atacante busca usar polimorfismo. Polimorfismo consiste na habilidade do binário existir de diversas maneiras, utilizando criptografia e empacotamento do binário. Entretanto, após ser carregado na memória do dispositivo, aplicações de segurança podem detectar a presença de um executável malicioso, pelo mesmo mecanismo de reconhecimento de padrões. Dessa forma, utiliza-se também metamorfismo, que permite que um binário seja reescrito de maneira diferente, mas com código semanticamente similar, mantendo as funções originais. Assim, os binários distribuídos entre os diversos dispositivos são diferentes, embora mantenham suas funcionalidades, dificultando a detecção e compartilhamento de informações entre estes aplicativos.

Para evitar que pesquisadores tenham acesso a informações sobre o *software* do *bot*, nota-se a implementação de detecção de *sandbox*, uma análise do ambiente em que o binário está sendo executado. Caso o *bot* esteja sendo executado em um ambiente virtualizado ou em uma *sandbox*, o código não realiza suas funções esperadas para a realização do ataque e o contato com o servidor C&C (*rallying*) não é estabelecido, evitando a análise detalhada do comportamento do *bot*. Buscando manter a sua presença na máquina infectada, o *bot* pode buscar desabilitar as aplicações de segurança existentes na máquina. Isso pode ser feito em máquinas que tenham sido comprometidas por outros *malwares* ou que estejam mal configuradas, haja visto que essa requisição normalmente exige a ação de um administrador da máquina. Além disso, pode-se evitar que os aplicativos e serviços da máquina hospedeira se comuniquem com serviços externos de segurança que os atualizam, mantendo-se uma lista de bloqueio de acesso no dispositivo, impedindo acessos legítimos. O *bot* pode também buscar a instalação de *rootkits* que alteram o comportamento do próprio sistema operacional, através de um usuário privilegiado, que permite ignorar regras de segurança do sistema, para que o *bot* possa persistir no sistema.

Do outro lado da comunicação com os *bots* está o *botmaster*. Os *botmasters* são os operadores das *botnets* e sua identificação pode os levar a responder legalmente pelo seus atos e, também, ocasionar na perda do controle da *botnet*, cessando seus lucros. Dessa forma, *botmasters*

escondem a sua real identidade utilizando dispositivos intermediários entre o servidor C&C e ele. Estes dispositivos intermediários podem fazer redirecionamento de serviços (como HTTP, SOCKS ou SSH) e podem ser dispositivos controlados pelo *botmaster* ou infectado por *bots*. Também podem ser utilizadas técnicas de anonimização (como a rede Tor), o uso de servidores intermediários em países com leis mais flexíveis quanto a cooperação na Internet e o uso de múltiplas redes criptografadas, como VPNs (Redes Privadas Virtuais - *Virtual Private Networks*).

O ponto central de uma *botnet* está no servidor de comando e controle, pois ele é quem comanda as ações dos *bots*. Para encontrar um servidor C&C, o *bot* precisa identificar o seu endereço IP. Para evitar a detecção dos servidores C&C, *botmasters* precisam frequentemente alterar o endereço IP dos servidores, pois permite que o endereço não seja bloqueado ou inserido em listas de bloqueio. Dessa forma, o uso direto no código de endereços IPs é possível mas está cada vez menos sendo utilizado devido a sua fácil detecção e possibilidade de derrubada por agentes da lei. O uso de IPs gerados, como visto na Subseção 2.1.3, é impraticável. Dessa forma, o uso de domínios ou redes P2P é o caminho normalmente escolhido pelos *botmasters*.

Uma das maneiras de trocar frequentemente os endereços IPs e manter o controle de um domínio é o uso de servidores DNS próprios, distribuídos em várias localidades. Assim, com uma multitude de servidores DNS e domínios, pode-se manter a *botnet* operacional e direcionar o tráfego para os servidores C&C, podendo mudar de endereço constantemente. Outra forma é o uso de servidores de DynDNS (DNS dinâmico). Estes serviços possibilitam que o mesmo domínio ou subdomínio possa alterar seu IP e, dessa forma, ter seu serviço direcionado para um novo servidor C&C, de maneira rápida. Uma técnica mais avançada para evitar a detecção do servidor C&C e esconder seu endereço IP é conhecida como *Fast Flux* (Salusky e Danford, 2007). Esta técnica se caracteriza por ter de centenas a milhares de endereços IP se registrando e desregistrando com uma entrada DNS, normalmente com um pequeno TTL (*Time-To-Live*). As entradas do DNS encaminham o tráfego para dispositivos infectados e estes dispositivos infectados mantêm uma conexão com o real servidor C&C, através de requisições HTTP. Dessa forma, os *bots* são capazes de identificar os comandos a serem executados para realizar o ataque sem, no entanto, ter de expor os reais servidores C&C.

Por fim, busca-se esconder a comunicação do servidor C&C com os *bots*. Caso essa comunicação seja facilmente identificada, pode-se verificar quais os dispositivos foram comprometidos e removê-los ou isolá-los na rede. Ainda assim, a identificação da comunicação também poderia expor os servidores C&C. Portanto, uma maneira de evitar a detecção dessa comunicação é o uso de criptografia, não permitindo que monitores de rede baseado em conteúdo sejam capazes de analisar o tráfego, forçando o uso de outras características do tráfego para a análise. Além disso, busca-se alterar os protocolos de comunicação, usando-os como caminho para outros: pode-se utilizar o protocolo HTTP para encaminhar mensagens enviados via SSH, por exemplo, já que HTTP dificilmente é bloqueado por *firewalls*.

A criação de tráfego de ruído também pode ser feito por *bots*, de modo a torná-los indistinguíveis entre outros processos e nós. Como a comunicação entre *bot* e servidor C&C é tipicamente de baixo volume, criação de volume de tráfego sem propósito mas com aparência legítima pode evitar a sua detecção. Ainda, é possível a utilização de serviços de terceiros como técnica de envio de comandos para os *bots*. Serviços como *Twitter* já foram utilizados como intermediários de servidores C&C. O *bot* se inscreve ao conteúdo de um perfil, através de RSS, por exemplo, e recebe a notificação de novos comandos a serem executados.

2.2 TRÁFEGO DE REDE

O tráfego gerado na Internet se utiliza de estruturas e padrões bem conhecidos e disseminados. Essas formas padronizadas de comunicação entre os dispositivos possibilitam que tenhamos uma rede global de comunicação: a Internet. Os dispositivos de comunicação, assim, apresentam interoperabilidade entre os sistemas, com protocolos padrões. A informação trocada entre dispositivos é transmitida de maneira padronizada. Dessa forma, para que um dispositivo seja capaz de se comunicar nesta rede, deverá obedecer os padrões definidos. (Tanenbaum e Wetherall, 2011) definiu a Internet como um conjunto de computadores autônomos interconectados por uma única tecnologia. Todos os padrões da Internet são mantidos e desenvolvidos pela IETF - *Internet Engineering Task Force* (Força de Trabalho de Engenharia de Internet). Os documentos padronizados IETF são denominados RFC - *Request for Comments* (Pedido de Comentários). A IETF define através dos RFC protocolos como TCP, IP, HTTP (para web) e SMTP.

Mesmo que técnicas de criptografia sejam utilizadas para esconder o conteúdo do tráfego, ainda há a necessidade de informações para que a mensagem de um dispositivo chegue a seu destinatário. Sendo assim, como as *botnets* operam na Internet, elas também tem de submeter aos padrões e protocolos definidos. Caso não o fizessem, o tráfego dos seus *bots* não seriam encaminhados pelos dispositivos e acabariam descartados, não afetando a rede que se pretende atacar (Bykova e Ostermann, 2002). Por outro lado, o uso de criptografia para a comunicação entre *bots* e servidores C&C, como descrito na Subseção 2.1.5, impossibilita a análise do conteúdo e a identificação de um tráfego típico uma *botnet*. De qualquer forma, espera-se que o comportamento de um *bot* seja diferente de um dispositivo que acessa serviços online, por exemplo. Para a análise de *botnets*, esta padronização do tráfego da Internet é extremamente importante pois possibilita a análise do comportamento dos nós - e não do conteúdo.

Nesta seção, serão apresentados estes padrões, o modelo OSI e o modelo TCP/IP de representação do tráfego. Estes modelos permitem extrairmos informações que possibilitam a análise do comportamento do tráfego sem a necessidade da análise do conteúdo do tráfego.

2.2.1 O modelo OSI, o modelo TCP/IP e a Internet

Para que os diversos dispositivos conectados na Internet fossem capazes de se comunicar, deu-se a necessidade de padronizar protocolos. Protocolos são regras que definem o formato, ordem das mensagens enviadas e recebidas pelos dispositivos e quais ações devem ser tomadas quando da transmissão ou recepção de mensagens. Para que essa comunicação seja bem sucedida, os dispositivos devem cooperar e agir de acordo com os padrões. Ao invés de realizar a implementação em um único módulo, capaz de realizar todas as funções necessárias para a comunicação, dividi-las em módulos facilita o entendimento e a detecção de erros. Assim, a tarefa de comunicação é dividida em funções que podem ser implementadas separadamente. Estas funções, menores e mais especializadas, podem ser implementadas em camadas, fornecendo interfaces para as camadas inferiores e superiores. Uma camada não precisa se preocupar como as camadas adjacentes operam e quais seus objetivos, desde que realize a sua função adequadamente.

O modelo OSI (*Open Systems Interconnection*) definiu um modelo 7 camadas que propiciam as funcionalidades necessárias para a comunicação entre dispositivos. Foi um esforço da indústria que concordou em um utilizar um padrão comum, nos primórdios da Internet, para que os dispositivos fossem capazes de interoperar entre dispositivos de *hardware* e *software* de diferentes fabricantes. O modelo OSI também trouxe benefícios no ensino de conceitos de rede e promoveu uma maior consistência do modelo em camadas. Este modelo serviu como referência para os desenvolvedores programarem redes que pudessem operar independentemente de fabricante e fornecedor. Por ser feito em camadas, uma camada de um dispositivo irá interagir

com a mesma camada de outro dispositivo. Assim, as funcionalidades das camadas adjacentes podem ser abstraídas para cada camada. Uma PDU (*Protocol Data Unit* - Unidade de Dados de Protocolo) é enviada para a mesma camada do dispositivo que se deseja comunicar, sendo esta apenas uma carga a ser carregada pela camada inferior. Na camada inferior, cabeçalhos e/ou rodapés são inseridos ao PDU da camada superior, formando um novo PDU, desta nova camada. Este processo continua até que o nível mais baixo seja atingido e a mensagem é transmitida para o dispositivo recipiente. Neste dispositivo, a mensagem é desempacotada da mais baixa até a mais alta, onde os cabeçalhos e/ou rodapés que compõem o PDU da camada são removidos e apenas o PDU da camada superior é encaminhado à camada acima.

As sete camadas do modelo OSI são: Aplicação, Apresentação, Sessão, Transporte, Rede, Enlace de Dados e Física. A camada 1 ou camada Física tem como objetivo movimentar os bits individuais que estão dentro do quadro de um nó para outro. Os quadros enviados pela camada superior, de Enlace, são transformados em sinais compatíveis com os meios onde os dados deverão ser transmitidos. A camada 2 ou camada de Enlace realiza o processo de enquadramento, preparando os pacotes oriundos da camada superior, de Rede, inserindo informações de controle, gerando um quadro (*frame*). Este processo é chamado de enquadramento (*framing*) e, assim, cada quadro é transmitido para a camada física para o dispositivo recipiente. A camada 3 ou camada de Rede fornece o esquema de endereçamento lógico utilizado para identificar os dispositivos de maneira única dentro das diversas redes existentes e o roteamento. Os pacotes da camada de Rede são conhecidos como datagramas. Nesta camada que os endereços lógicos são convertidos em endereços físicos, para que possam atingir o seu destino corretamente. A camada 4 ou camada de Transporte realiza a segmentação dos dados das camadas superiores (se necessário), faz controle de fluxo e transporte confiável de dados, não importando o meio físico. Os dois protocolos de transporte mais utilizados na Internet são TCP e UDP. A camada 5 ou de Sessão estabelece serviços de segurança, gerenciando a comunicação contínua entre dois nós de modo a formarem uma única sessão. A camada 6 ou de Apresentação é responsável pelas transformações ou traduções adequadas nos dados antes do seu envio à camada de sessão, tais como compressão de textos, criptografia e conversão de codificação de textos. A camada 7 ou de Aplicação é a mais superior e é a responsável pela interface com as aplicações dos computadores. Nesta se encontram os protocolos HTTP, SMTP, FTP, DNS, SSH, entre outros.

O modelo OSI foi o primeiro a se tornar uma referência para os estudos de redes e as nomenclaturas utilizadas. Entretanto, na prática, a *Internet Protocol Suite* (conjunto de protocolos para internet) é a arquitetura que foi de fato implementada e é utilizada nas redes atuais, tanto nas redes locais como na Internet. Esta arquitetura, que também é conhecida como arquitetura TCP/IP, possui apenas 4 camadas. A camada de Aplicação no modelo TCP/IP engloba as três camadas mais altas do modelo OSI (Aplicação, Apresentação e Sessão) e as duas camadas mais baixas do modelo OSI (Enlace de Rede e Física) constituem a camada de Acesso a Rede do modelo TCP/IP. A camada de Rede no modelo OSI foi renomeado para Internet, no modelo TCP/IP. No modelo TCP/IP também não é costumeiro se referir às camadas pelos números, sendo a referência pelos números das camadas utilizadas no modelo OSI.

No entanto, nota-se que na academia e livros de autores renomados (como Tanenbaum (Tanenbaum e Wetherall, 2011) e Kurose (Kurose e Ross, 2017)), ainda se utiliza a nomenclatura definida pela arquitetura OSI, mesmo para a Internet. As camadas que não são utilizadas ou foram integradas em outras camadas nas redes atuais não são citadas. Essa variedade de definições e modelos levou a criação de um terceiro modelo, um modelo híbrido, geralmente apresentado em 5 camadas, como mostrado na Figura 2.3. Usualmente, a camada de Aplicação, neste modelo híbrido, passa a englobar as três camadas superiores do modelo OSI (Aplicação, Apresentação e Sessão), mantendo a nomenclatura para as camadas inferiores.

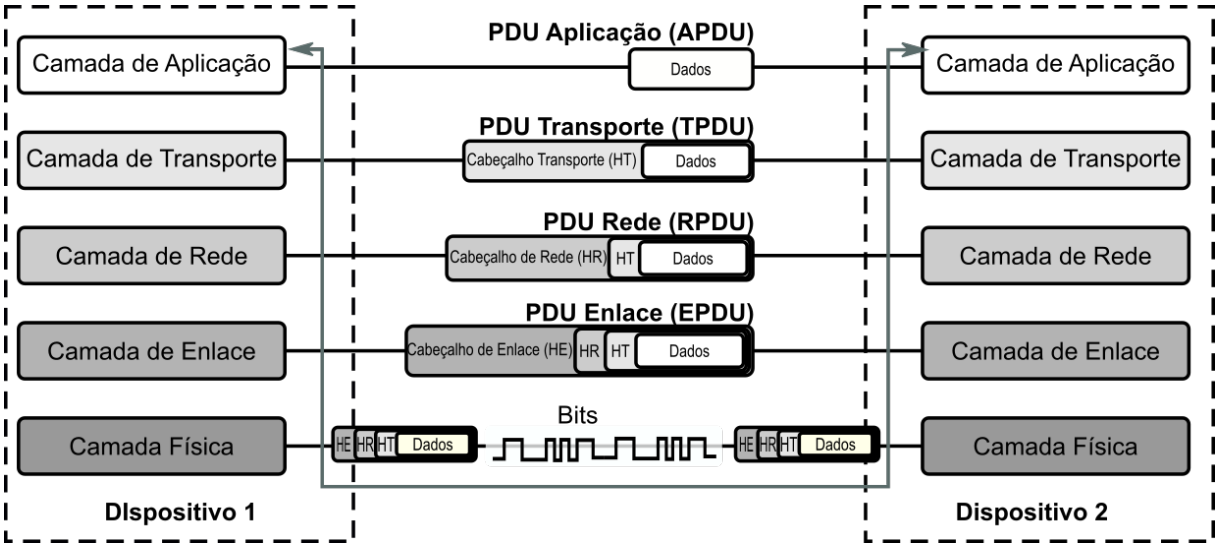


Figura 2.3: A camada inferior encapsula a camada superior

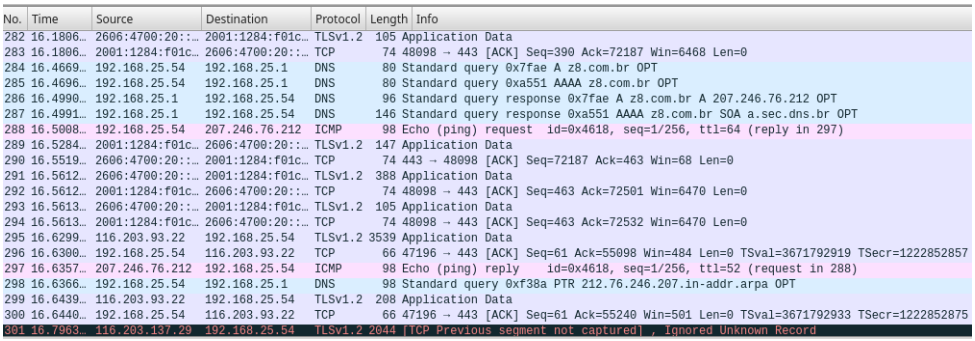
De qualquer forma, os pacotes enviados pela camada Física, os cabeçalhos e rodapés que são inseridos em cada camada são acessíveis para um observador de rede. Cada PDU de uma dada camada pode ser acessada e armazenada por um monitor de rede que queira coletar informações e processá-las de modo a analisar o comportamento da rede. Embora o acesso ao conteúdo da mensagem possa ser criptografado (na camada de Aplicação), como já dito, as informações contidas nas camadas abaixo não podem. Dessa forma, é possível realizar a coleta de informações que possam formar o comportamento dos dispositivos que trafegam dados na rede. É neste cenário que a extração de características do tráfego opera.

2.2.2 Extração de Características

Como descrito na Subseção 2.2.1, cada camada do modelo apresenta características que permitem identificá-la como aquela camada se comporta. Dessa forma, um monitor de rede, que analisa os pacotes trocados em uma determinada rede, é capaz de identificar estas características e distingui-las entre pacotes diferentes. A extração de características consiste na projeção do conjunto original de dados em um novo espaço onde a dependência das variáveis é minimizada, reduzindo o número de dados necessários para realizar uma análise. Com a extração de características, não se tem mais o conjunto inicial de dados mas informações diferentes e referentes a este conjunto de dados iniciais.

Na Figura 2.4, é apresentada uma captura de tela do monitor de rede Wireshark. É possível notar diferenças entre os protocolos utilizados, endereços de origem e destino, o tamanho do pacote e a mensagem destinada no pacote. As ferramentas de captura, quando implementadas na rede, permitem que diversos dados sejam coletadas dos pacotes sendo trafegados. Com a captura em um longo período de tempo, é possível extrair informações sobre os dados, montar estatísticas e diferenciar o comportamento de um dado nó da rede de outro. Vale ressaltar que a captura e análise de pacotes é apenas uma maneira de visualizar o tráfego sendo transmitido. Pode-se realizar a análise através de fluxos.

Os fluxos de tráfego são definidos pela RFC 3697 (Rajahalme et al., 2004) como uma sequência de pacotes enviados de uma determinada origem para um destino particular, de modo que a origem deseje classificar o conjunto como um fluxo. O fluxo consiste de todos os pacotes de uma conexão específica não sendo necessariamente uma relação 1:1 para o pacote da camada de Transporte. Dessa forma, a análise por fluxo também é possível, haja visto que nela contém as



No.	Time	Source	Destination	Protocol	Length	Info
282	16.1806…	2606:4700:20::…	2001:1284:f01c…	TLSv1.2	105	Application Data
283	16.1806…	2001:1284:f01c…	2606:4700:20::…	TCP	74	48098 → 443 [ACK] Seq=390 Ack=72187 Win=6468 Len=0
284	16.4669…	192.168.25.54	192.168.25.1	DNS	80	Standard query 0x7fae A z8.com.br OPT
285	16.4696…	192.168.25.54	192.168.25.1	DNS	80	Standard query 0xa551 AAAA z8.com.br OPT
286	16.4990…	192.168.25.1	192.168.25.54	DNS	96	Standard query response 0x7fae A z8.com.br A 207.246.76.212 OPT
287	16.4991…	192.168.25.1	192.168.25.54	DNS	146	Standard query response 0xa551 AAAA z8.com.br SOA a.sec.dns.br OPT
288	16.5008…	192.168.25.54	207.246.76.212	ICMP	98	Echo (ping) request id=0x4618, seq=1/256, ttl=64 (reply in 297)
289	16.5284…	2001:1284:f01c…	2606:4700:20::…	TLSv1.2	147	Application Data
290	16.5519…	2606:4700:20::…	2001:1284:f01c…	TCP	74	443 → 48098 [ACK] Seq=72187 Ack=463 Win=68 Len=0
291	16.5612…	2606:4700:20::…	2001:1284:f01c…	TLSv1.2	388	Application Data
292	16.5612…	2001:1284:f01c…	2606:4700:20::…	TCP	74	48098 → 443 [ACK] Seq=463 Ack=72501 Win=6470 Len=0
293	16.5613…	2606:4700:20::…	2001:1284:f01c…	TLSv1.2	105	Application Data
294	16.5613…	2001:1284:f01c…	2606:4700:20::…	TCP	74	48098 → 443 [ACK] Seq=463 Ack=72532 Win=6470 Len=0
295	16.6299…	116.203.93.22	192.168.25.54	TLSv1.2	3539	Application Data
296	16.6300…	192.168.25.54	116.203.93.22	TCP	66	47196 → 443 [ACK] Seq=61 Ack=55098 Win=484 Len=0 TSval=3671792919 TSecr=1222852857
297	16.6357…	207.246.76.212	192.168.25.54	ICMP	98	Echo (ping) reply id=0x4618, seq=1/256, ttl=52 (request in 288)
298	16.6366…	192.168.25.54	192.168.25.1	DNS	98	Standard query 0xf38a PTR 212.76.246.207.in-addr.arpa OPT
299	16.6439…	116.203.93.22	192.168.25.54	TLSv1.2	208	Application Data
300	16.6440…	192.168.25.54	116.203.93.22	TCP	66	47196 → 443 [ACK] Seq=61 Ack=55240 Win=501 Len=0 TSval=3671792933 TSecr=1222852875
301	16.7963…	116.203.137.29	192.168.25.54	TLSv1.2	2044	[TCP Previous segment not captured] , Ignored Unknown Record

Figura 2.4: Monitor de rede Wireshark - detalhe para os diferentes protocolos apresentados

informações necessárias para reconstruir e/ou analisar o tráfego. Neste trabalho, optamos pela análise por pacotes por ser uma técnica mais antiga e consolidada, capaz de expressar o tráfego e pela existência de base de dados para testes que se utilizam dessa estrutura, em detrimento de fluxos. A realização da análise por fluxos é possível, com alterações na implementação.

Na Figura 2.5, é possível notar o empilhamento dos protocolos de cada camada. Foi capturado um *frame* (camada de enlace – *Data Link*), identificado pelo número da sequência 284, com origem do dispositivo com endereço MAC *3c:a0:67:af:ea:35*. O protocolo da camada de rede utilizado é o IP, com origem no endereço IP *192.168.25.54* e destino *192.168.25.1*. Na camada de Transporte, é utilizado o protocolo UDP (*User Datagram Protocol*). A camada de Aplicação utiliza as portas definidas na camada de transporte para manter a conexão e saber onde deve ser enviado a requisição e a resposta entre programas e dispositivos. A porta de origem é 36662 e a de destino é a porta 53. O protocolo utilizado nesta camada é o DNS (*Domain Name System*).

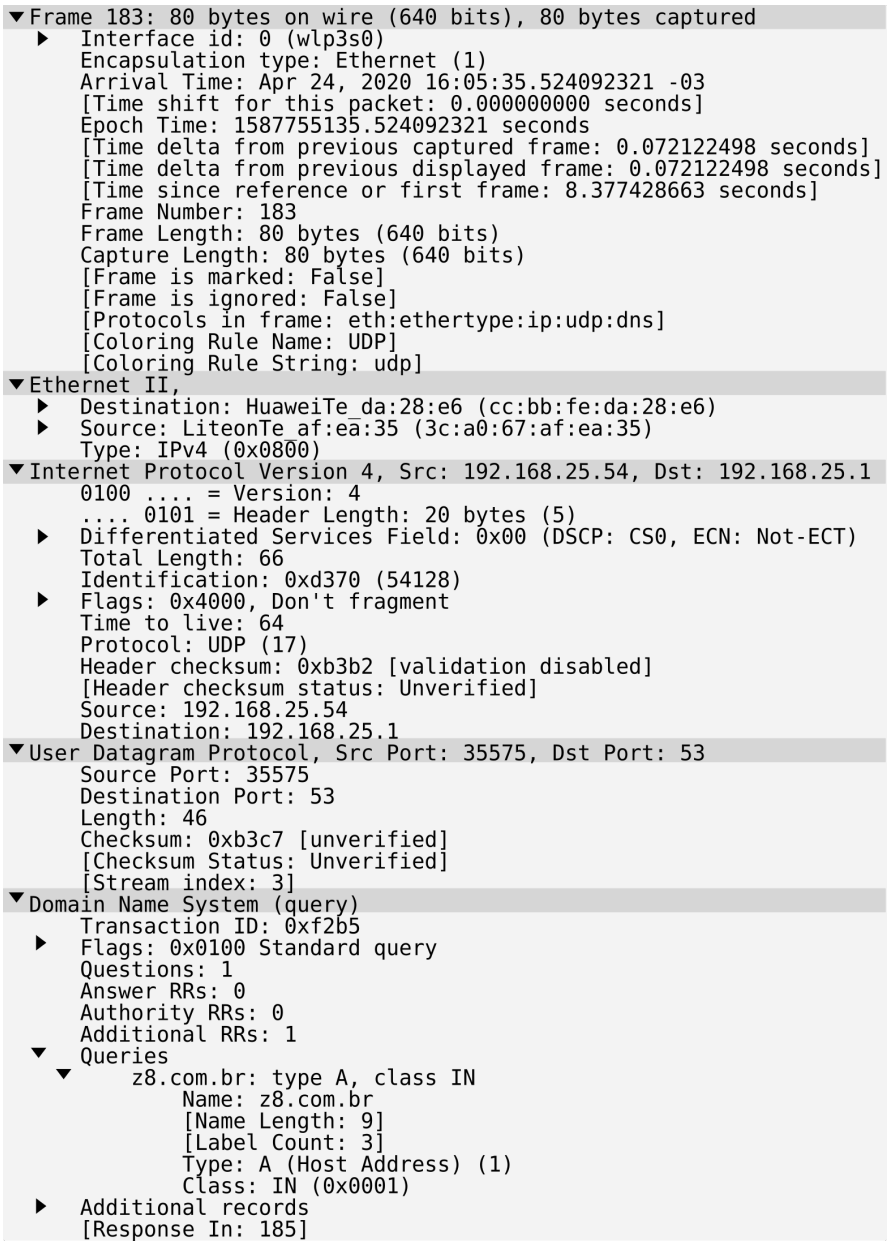


Figura 2.5: Monitor de rede Wireshark - requisição DNS para o domínio z8.com.br

Em (Iglesias e Zseby, 2015), os autores utilizam 41 características que podem ser extraídas e construídas a partir de pacotes de rede. Estas características foram divididas em: básicas, que podem ser extraídas diretamente do pacote; de tráfego, que podem ser inferidas para o mesmo nó ou serviço; de conteúdo, que é extraída da análise do dado sendo trafegado. Assim, para cada nó de rede, identificado por seu endereço IP, por exemplo, pode-se gerar informações que podem ser relevantes em uma análise de comportamento de rede. Quais os protocolos utilizados por um dado nó? Com quantos nós esse nó se comunica? Quantos pacotes ele troca com cada nó? Qual o tamanho dos pacotes? A relevância de cada informação que pode ser extraída para uma dada análise é relativa ao que se pretende analisar. O Capítulo 4 descreve as características elencadas a partir dos pacotes que são relevantes na análise de *botnets*.

2.2.3 Seleção de Características

As características do tráfego serão utilizadas com o propósito de realizar uma classificação. No caso do presente trabalho, classificar um determinado nó em *bot* ou não. Embora pareça que quanto mais características a serem analisadas de um tráfego, melhor será a análise, reduzir o número de características analisadas tem como benefício a redução do uso de recursos computacionais. Do ponto de vista teórico, incluir variáveis sem sentido ou redundantes não deve piorar o resultado esperado na performance de classificadores. Porém, do ponto de vista prático, técnicas de aprendizagem de máquina ignoram a distribuição dos dados e tendem a aproximar resultados na presença de características redundantes.

Outro problema relativo ao excesso de características é conhecido como "maldição da dimensionalidade", *i.e.*, a complexidade de uma entrada multi-dimensional cresce exponencialmente com cada nova variável, tornando o espaço de busca espaço no enorme espaço de possibilidades criado (Bellman, 1966). Este problema de adição de variáveis irrelevantes ou redundantes tem um grande efeito em técnicas de classificação de dados, com a piora percebida dependendo da técnica utilizada. Assim, a adição de uma nova característica não significa necessariamente um bom resultado de classificação. Depois de um certo ponto, adicionar novas características pode piorar o desempenho do classificador. Assim, a seleção de características realiza a exclusão de características que sejam irrelevantes ou redundantes para posteriores processos de representação dos dados ou classificação. A utilização de características discriminantes influi no sucesso da construção de um bom classificador. Alguns aspectos devem ser considerados na escolha de características, quando buscamos representar um conjunto de dados: desempenho, tempo de processamento e tamanho dos dados de entrada.

O problema de seleção de características não é um problema trivial. Dado um conjunto de N características, o espaço de busca de subconjuntos que representam estes dados é de 2^N. Para cada subconjunto escolhido, deve ser feita uma avaliação se este subconjunto é adequado. Assim, o problema consiste na busca de um conjunto de características que seja o menor subconjunto necessário e suficiente para resolver um dado problema e que melhore o desempenho de uma dada análise. Em problemas reais, características discriminantes não são conhecidas a priori. Além disso, características raramente são totalmente independentes. Inclusive, duas características irrelevantes, quando unidas podem formar uma nova característica relevante e com grande capacidade de discriminar os dados, podendo ser pertinente à análise.

Para tanto, deve-se haver uma avaliação se um dado subconjunto é ótimo. Embora a análise exaustiva, avaliando os 2^N certamente encontraria um subconjunto ótimo, na prática esta análise é inviável. Como dito, o uso destas características está intrinsecamente ligado ao problema de realizar uma classificação. Portanto, de acordo com o critério de avaliação dos subconjuntos podemos dividi-los em três categorias: *filter*, *wrapper* e *embedded*.

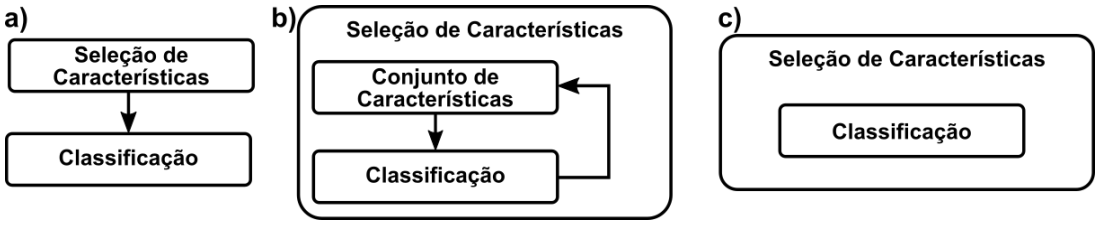


Figura 2.6: Métodos de seleção de características: a) filter (b) wrapper (c) embedded

O método *filter* é o mais rápido e direto dos três. Ele não se utiliza de classificadores para identificar a melhor característica, junto com uma base de treinamento. Se utiliza exclusivamente de propriedades intrínsecas a elas, não sendo afetadas pelas fases posteriores de treinamento e

inferência. Embora torne a seleção rápida, é sua principal desvantagem pois este método não enxerga o efeito da sua seleção na performance da classificação. Neste sentido, o método *wrapper* atua conjuntamente com um algoritmo de classificação para identificar o melhor subconjunto de características. Esta abordagem, em geral, produz melhores resultados que os do método *filter*, com a desvantagem de ter um custo computacional mais alto. Além disso, esta seleção pode incorrer em *overfitting* dos dados quando da classificação, sendo necessário utilizar um conjunto da base de dados para a validação. Já o método *embedded* é um método híbrido, combinando as abordagens *filter* e *wrapper*. Utiliza algoritmos de aprendizagem que tem a capacidade de avaliar características ao mesmo tempo que realiza a classificação, como por exemplo, árvores de decisão. Como resultado, a exploração dos dados é otimizada como um modelo preditivo, não sendo necessário um retreinamento a cada novo conjunto de dados.

O tipo de dado que é extraído e selecionado é determinante para decidir o método de detecção apropriado. No conjunto original de dados, as informações podem ser numéricas (tamanho do pacote), textuais (*hostname*) ou mistas (endereços MAC). Dessa forma, um tratamento pode ser feito de modo a se adequar a um método que satisfaça a proposta de seleção. A realização de uma normalização dos dados também é comum e incentivada.

2.3 MÉTODOS DE CLASSIFICAÇÃO

Vivemos uma explosão de geração, medição e armazenamento de informações, criadas em uma velocidade estonteante nas mais diversas disciplinas, desde o mercado financeiro, exploração de petróleo, mídias sociais e ciências biológicas. Serão gerados, em 2025, mais de 160 zettabytes (= 1 000 000 000 000 000 000 000 bytes = 1 bilhão de terabytes) de dados em todo o mundo (Reinsel et al., 2018). Esse crescimento se dará principalmente por dispositivos embarcados e da Internet das Coisas (IoT - *Internet of Things*). A interação dos humanos com esses dispositivos vai ocorrer de maneira mais frequente, sem que haja deliberadamente uma ação dos usuários, mas em plano de fundo. Isto acarretará em um aumento vultuoso no volume de dados que serão armazenados e precisarão ser processados. É impossível para o ser humano lidar com todo esse volume de dados. É necessário a utilização de mecanismos computacionais que sejam capazes de receber e processar esse conjunto de dados e agregar uma inteligência aos dados, tornando-os em uma informação útil para tomada de decisões. Naturalmente, a segurança destes dados vai continuar sendo um desafio, tanto no armazenamento quanto durante o transporte, fazendo com que a análise e detecção de *botnets* cresça em relevância.

Embora as técnicas de classificação de dados existam desde 1950, não havia na época a disponibilidade de dados que temos hoje, além do poder computacional e as ferramentas necessárias para a análise. Além disso, o volume de dados gerados não era um empecilho para a análise, sendo especialistas humanos capazes de classificar os dados. Dessa forma, com o crescimento exponencial do volume de dados, deu-se a necessidade da transferência de conhecimento dos humanos para dispositivos computacionais.

Neste sentido, com o aumento do volume de tráfego gerado entre dispositivos computacionais e a necessidade de prover soluções de segurança, deve-se buscar uma maneira de podermos classificar o tráfego e os dispositivos em agentes maliciosos ou não, para que a rede permaneça segura. Deve-se notar que a maioria dos dados de rede disponíveis que serão avaliados não possuem uma classificação, a priori, de serem benignos ou maliciosos. Aprendizagem de máquina supervisionada é utilizada para criar modelos matemáticos quando tanto o conjunto de dados de entrada quanto a saída esperada estão disponíveis (no nosso caso, agente malicioso ou não). Essa classificação da saída esperada usualmente exige uma classificação manual, por um humano, para posterior utilização do modelo criado. Já a aprendizagem de máquina não

supervisionada é um tipo de aprendizagem de máquina que procura por padrões não detectados no conjunto de dados, sem que haja uma pré-classificação indicando qual era a saída esperada. Também, esta técnica usualmente necessita de pouca intervenção humana.

Os dados coletados da rede, continuamente, formam um conjunto de séries temporais. Estas séries temporais podem ser extraídas de qualquer tráfego de rede, tendo um apelo maior na Internet das Coisas, devido ao volume massivo de dados gerados, sendo uma fonte gigantesca de coleções de séries temporais. Além disso, a diversidade e a natureza não estruturada dos dados, na Internet das Coisas, exige a necessidade de criarmos modelos capazes de representá-los. Neste sentido, serão apresentadas duas técnicas que permitem a extração de modelos e nos permitem classificar dados, mesmo com sua natureza não estruturada: a aprendizagem de máquina não-supervisionada e processamento de sinais em grafos.

A aprendizagem de máquina não-supervisionada permitirá que comportamentos similares no tráfego sejam identificados. Dessa forma, comportamento de usuários poderão ser diferenciados de dispositivos que possuam alguma atividade maliciosa. Assim, esta técnica propiciará um conjunto de nós que possuem comportamento similar, dadas características extraídas da rede. Entretanto, esse agrupamento em conjuntos não necessariamente classifica os nós. Para tanto, utilizaremos o processamento de sinais em grafos que possibilitará, através da análise da série contínua dos dados (e não apenas das características), extrair uma representação de baixa dimensionalidade que permitirá fazer inferências, estatísticas e quantificações que possibilitarão a detecção de um comportamento malicioso na rede.

2.3.1 Sistema de Predição - *Early Warning Signals*

A predição é o ato de saber antecipadamente sobre o resultado de um evento, baseado em um conhecimento específico (Huifang Feng e Yantai Shu, 2005). Na natureza, doenças são evitadas com a identificação precoce de fatores de risco, como em Lesões Renais Agudas (Kate et al., 2016). O mesmo acontece em doenças cardiovasculares onde a diabete *mellitus* está associada com um aumento significante no risco de doenças cardiovasculares, mas este risco não é uniforme (Wander et al., 2020). Em doenças infecciosas, a avaliação de fatores internos e externos - que podem desencadear eventos emergenciais - são cada vez mais úteis na predição destes eventos (Ogden et al., 2017). Portanto, o objetivo da predição consiste em "estar a frente da curva", permitindo que indivíduos e sistemas de saúde possam reagir de acordo e prontamente em relação aos resultados que virão. O mesmo acontece em tráfego de redes computacionais, onde tendências podem ser observadas e preditas.

Quando deseja-se detectar anomalias, algoritmos são capazes de observar um comportamento normal e inferir um comportamento anômalo. Estas metodologias tem um número alto de falso-positivos e podem atrapalhar na performance do sistema subjacente, se um número muito alto de falsos diagnósticos for detectado (imagine considerar que todos indivíduos que possuem diabetes *mellitus* desencadearão em morbidades ou morte). Assim, ao invés de tratar todos os possíveis pacientes com fatores de risco, outros sinais indicadores são analisados para indicar a necessidade de posterior análise na condição do paciente. Em sistemas de saúde, testagem sanguínea e tomografias podem auxiliar em casos críticos, indicando a necessidade (ou não) de, por exemplo, uma cirurgia investigatória. Mas estas ações são disparadas por indicadores precoces do sistema de saúde do indivíduo. Indicadores precoces focam em uma fração de órgãos enquanto as próximas análises são mais amplas e podem focar em todo o sistema (o corpo humano). Investigações iniciais podem focar em um órgão que não é a causa da deficiência, necessitando ter como alvo mais órgãos para encontrar (detectar) os agentes ruins.

O sistema de predição aqui empregado se baseia no conceito de metaestabilidade de sistemas. A metaestabilidade pode ser uma característica global de um sistema, quando variações

de parâmetros não causam alterações severas no comportamento do sistema. De maneira geral, porém, sistemas possuem diversos pontos de metaestabilidade e podem migrar de um ponto de metaestabilidade para outro através de um fenômeno conhecido como "transição crítica". Para exemplificar o conceito de estabilidade, considere um elástico. Uma vez que uma força de tração é aplicado a ele, ele se expande, de maneira gradual e reversível (uma vez retirada a força, o elástico volta a seu estado original). Uma pequena força causa uma pequena variação no tamanho do elástico, enquanto forças maiores causam maior variação, de maneira quase linear. Em outros casos, pequenas variações nos parâmetros e condições podem causar uma mudança desproporcional no estado do sistema. Essa mudança, porém, é contínua e pode ser revertida, desde que as condições iniciais sejam restauradas aos níveis iniciais. Há situações, porém, em que minúsculas alterações nas condições podem causar uma resposta não contínua e extrema no estado do sistema que não são facilmente reversíveis ou até irreversíveis. Isto acontece quando um limite é atingido e um sistema abruptamente muda para um estado diferente. Estas mudanças abruptas causadas por pequenas alterações nas condições são descritas como transições críticas. Transições críticas também surgem em sistemas com estados alternativos estáveis: sistemas que têm mais de uma configuração sob as mesmas condições externas.

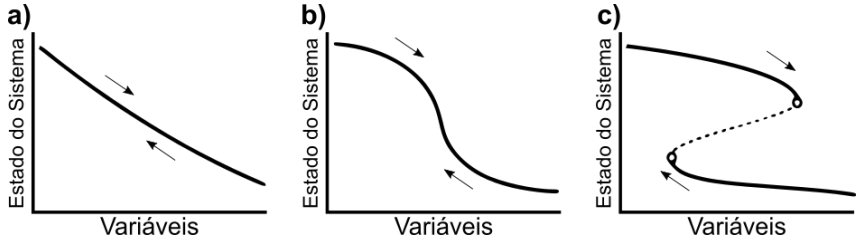


Figura 2.7: Sistemas sem transição crítica (sem grandes variações no estado (*a)* e com grandes variações no estado (*b)*) e com transição crítica (*c)*

Quando se quer detectar anomalias, algoritmos podem agrupar dados de um comportamento normal e inferir um comportamento anômalo. Estas metodologias têm uma alta percentagem de falso-positivos e podem interromper o funcionamento ou degradar a performance do sistema avaliado se um número alto de falso-positivos for encontrado (imagine tratar todos os pacientes com Lesão Renal Aguda como se fosse resultar em comorbidades e mortalidade). Ao invés de tratar todos os possíveis pacientes com sinais precoces de fatores de risco, os indicadores estatísticos indicam a necessidade de uma análise na condição do paciente. Em sistemas de saúde, testes sanguíneos e tomografias podem auxiliar em casos críticos, mostrando ou não a necessidade de uma intervenção cirúrgica, por exemplo. Mas estas ações mais profundas partem-se de um conjunto de indicadores iniciais de fácil obtenção que propiciam uma análise mais profunda. De maneira similar, em redes de computadores, foram utilizados indicadores estatísticos para indicar a possível ocorrência de ataques DDoS para indicar uma transição crítica na rede, de um estado metaestável para outro, de maneira irreversível (Dakos et al., 2012). Embasado na análise do tamanho dos pacotes trafegados nas redes, os estados futuros da rede dependem apenas do conhecimento do estado atual, podendo-se desprezar coletas passadas. Considera-se o fenômeno da metaestabilidade ocorrendo em processos estocásticos, em que a coleção de variáveis é aleatória. Para tanto, utiliza-se indicadores estatísticos (*leading indicators*). Assim, uma vez identificada uma transição crítica, pode-se analisar de forma mais detalhada o comportamento da rede para identificar os possíveis causadores desta transição.

2.3.1.1 *Indicadores Estatísticos - Leading Indicators*

Na técnica de *Early Warning Signals*, quatro indicadores estatísticos são utilizados para identificar e predizem uma mudança de estado crítico nas redes computacionais: taxa de retorno, autocorrelação, coeficiente de variância e assimetria. O fenômeno de metaestabilidade é associado a uma rede de computadores sob ataque. Em uma rede que está em um estado metaestável, um ataque DDoS, por exemplo, produz uma perturbação no estado da rede. Estas perturbações podem trazer o estado da rede para valores que são próximos de superar a barreira entre dois estados. Tendo como entrada os quatro indicadores estatísticos citados, os estados são calculados. Se a perturbação for grande o suficiente, a rede computacional sofrerá uma transição crítica. Os estados da rede são analisados tomando por base o tamanho dos pacotes na rede, em uma janela de tempo. Portanto, um estado da rede é caracterizado pelo comportamento de um conjunto de pacotes de rede durante uma janela de tempo.

O primeiro indicador estatístico é a taxa de retorno. Este indicador mede a taxa com que a rede retorna ou se recupera até um estado metaestável de equilíbrio. Ele é calculado pelo autovalor dominante da matriz composta pelas observações em uma janela de tempo. Autovalores dominantes caracterizam a taxa de retorno para o estado metaestável. Ele pode indicar a proximidade de uma transição crítica, indicando a irreversibilidade de uma transição, *i.e.*, se o estado da rede está migrando de um estado metaestável para outro, quebrando a barreira que os separa. Quanto maior a taxa de retorno, mais rápido a rede se recupera de pequenas perturbações ao redor de seu estado atual e, consequentemente, a rede tem maior resiliência para mudanças. A taxa de retorno é reduzida quando a rede se aproxima de uma transição crítica, reduzindo-se suavemente para zero conforme a transição crítica se aproxima(Scheffer et al., 2009). A redução suave da taxa de retorno próximo a uma transição crítica é denominada Desaceleração Crítica (*Critical Slowing Down (CSD)*, um termo cunhado da teoria de sistemas dinâmicos.

O segundo indicador estatístico é a autocorrelação. É uma métrica empregada para avaliar a correlação entre observações. Mede quanto um estado da rede se torna cada vez mais similar entre duas observações consecutivas. Em (Scheffer et al., 2009), os autores observaram que a autocorrelação em séries temporais aumenta quando transições críticas se aproximam. Transições críticas tendem a aumentar a correlação em baixas defasagens (*low lags* - também conhecido como memória de curto prazo) entre observações da série temporal. A autocorrelação é calculada com uma defasagem (*lag-1*), *i.e*, a correlação entre a série temporal e ela defasada por uma janela de tempo para trás. Na Equação 2.1, as variáveis z_t e z_{t+1} representam duas observações consecutivas nos tempos t e $t+1$, respectivamente, μ é a média em uma janela de tempo e σ é a variância da variável z_t.

$$\rho_1 = \frac{E[(z_t - \mu)(z_{t+1} - \mu)]}{\sigma_z^2} \tag{2.1}$$

O terceiro indicador estatístico é o coeficiente de variação. É uma medida estatística que indica o nível de variância na série temporal. É calculado como $CV = \frac{SD}{\mu}$ onde SD é um desvio padrão empírico, calculado como $SD = \sqrt{\frac{1}{n-1}\sum_{t=1}^{n}(z_t - \mu)^2}$. Transições críticas e fenômenos de desaceleração crítica aumentam a variância da série temporal (Scheffer et al., 2009). Com o auxílio dos outros indicadores estatísticos, o coeficiente de variação permite predizer tendência de uma transição crítica, como no caso de um ataque de negação de serviço distribuído (DDoS).

O quarto indicador é a assimetria. Este indicador é um conhecido valor estatístico que mensura a assimetria na distribuição das observações. Se a assimetria está aumentando ao longo do tempo, a distribuição das observações está ficando cada vez mais assimétrica. A assimetria é calculada como mostrada na Equação 2.2. Assim como a variância, a assimetria pode aumentar

devido uma transição crítica, significando que a assimetria nas observações aumenta. Isto acontece porque a dinâmica próxima a barreira da transição se torna lenta. (Scheffer et al., 2009) observou um aumento na assimetria em diversas séries temporais quando eles estavam próximos de uma transição crítica. A assimetria também é empregada como um indicador estatístico e é analisada em conjunto com os três anteriores.

$$\gamma = \frac{\frac{1}{n}\sum_{t=1}^{n}(z_t - \mu)^3}{\sqrt{\frac{1}{n}\sum_{t=1}^{n}(z_t - \mu)^2}} \tag{2.2}$$

Estes cálculos estatísticos são empregados como indicadores de tendência de ataques. A análise individual de um único indicador estatístico não deve ser considerada para a tomada de decisões, de modo a reduzir o número de falso-positivos e falso-negativos. A análise deve ser feita com todos os indicadores, de maneira conjunta. Em (Scheffer et al., 2009), os autores demonstraram que este conjunto de indicadores estatísticos definem um padrão específico de um sistema na iminência de sofrer uma transição crítica. Eles observaram que na iminência de uma mudança disruptiva (i) a taxa de retorno diminui, (ii) a autocorrelação com uma defasagem aumenta, (iii) o coeficiente de variação aumenta e (iv) a assimetria aumenta. Este conjunto de indicadores é capaz de predizer o começo de um ataque de negação de serviço volumétrico.

2.3.2 *Self-Organizing Map*

De maneira geral, qualquer método que incorpore informação de um conjunto de treinamento em um modelo utiliza aprendizagem. Aprendizagem se refere a criação de classificadores que, utilizando dados de treinamento, aprendam ou estimem padrões desconhecidos do modelo. Assim, o algoritmo de aprendizagem busca reduzir algum tipo de erro ao estimar uma classificação para um conjunto de dados. Como já dito, na aprendizagem de máquina supervisionada, há a indicação da classificação de uma categoria no conjunto de dados e busca-se reduzir o erro ao estimar novas entradas. Porém, na aprendizagem de máquina não-supervisionada, também conhecida como clusterização, não há a classificação explicitamente mostrada e o sistema de classificação forma grupos (*clusters*) dos padrões de entrada.

A vantagem do uso da clusterização é que a aquisição e captura dos dados é muito menos custosa que na aprendizagem supervisionada. Não há a necessidade de classificação de cada uma das entradas para a realização do treinamento. Também, em muitas aplicações, a caracterização dos padrões pode mudar gradualmente ao longo do tempo e essas mudanças podem ser percebidas por um classificador não-supervisionado. Por fim, para a realização de uma análise inicial de uma investigação, pode ser valioso entender como os dados estão estruturados, descobrindo subclasses distintas ou similaridades entre padrões, deixando a classificação final para outro método (como métodos de aprendizagem de máquina supervisionada ou, no nosso caso, o processamento de sinais em grafos).

A aprendizagem de máquina não-supervisionada é um tipo de aprendizado de máquina que busca por padrões previamente desconhecidos em um conjunto de dados, sem a pré-existência de classificações e com o mínimo de interação humana. A aplicação desta técnica em análise de *bots* se mostrou frutífera em trabalhos anteriores (Daya et al., 2019). Entre as técnicas de clusterização, encontram-se *k-means*, *DBSCAN* (*Density-based spatial clustering)* e *Self-Organizing Map*. Com base em resultados obtidos na revisão bibliográfica, será detalhada a técnica *Self-Organizing Map*, que é a que apresentou melhores resultados.

A técnica *Self-Organizing Map*, SOM, é uma técnica de aprendizagem de máquina não-supervisionada, baseada em redes neurais, que produz uma saída bidimensional (de baixa dimensão mas tipicamente bidimensional), discretizando a representação do conjunto de dados de

entrada, permitindo uma útil visualização dos dados, multidimensionais. O mapa bidimensional criado pelo SOM é dividido em nós (ou neurônios). O tamanho deste mapa é definido antecipadamente, usualmente definindo o espaço em um *grid*[1] finito de nós ou retângulos. A cada nó é associado a um vetor "peso", que indica a sua posição no espaço de entradas. Enquanto os nós ficam fixos no *grid*, os vetores "pesos" são ajustados de acordo com as entradas apresentadas ao algoritmo, buscando reduzir a distância euclidiana entre eles. As entradas são apresentadas para o algoritmo e o conjunto de pesos são ajustados até não haver mudança de posição dos pesos com as entradas, mantendo o *grid* na sua forma final ou atingindo um número de iterações. O treinamento consistem em 5 passos:

1. Inicializar (aleatoriamente) os conjuntos de pesos;
2. Aleatoriamente escolher uma entrada;
3. Escolher o nó vencedor daquela entrada, de acordo com a distância euclidiana;
4. Atualizar os pesos dos nós;
5. Repetir a partir de 2, até o fim do treinamento

Na Figura 2.8, nota-se as entradas, que são o conjunto de pesos que serão ajustados e a coleção de nós, do *grid*. Vale lembrar que a quantidade de pesos é igual ao tamanho do *grid* definido. Portanto, o processo de treinamento busca definir e ajustar o valor destes pesos. Como a inicialização dos pesos é usualmente aleatória, duas execuções sucessivas do algoritmo não necessariamente trarão o mesmo resultado. Porém, como o intuito desta técnica não é realizar uma classificação (que será realizada posteriormente) mas sim dar um entendimento aos dados, permitindo inferir e analisar padrões nos dados. Embora a classificação não seja realizada, a técnica de aprendizagem de máquina não-supervisionada agrega informações e conhecimento aos dados, sem que a sua aplicação prejudique a análise.

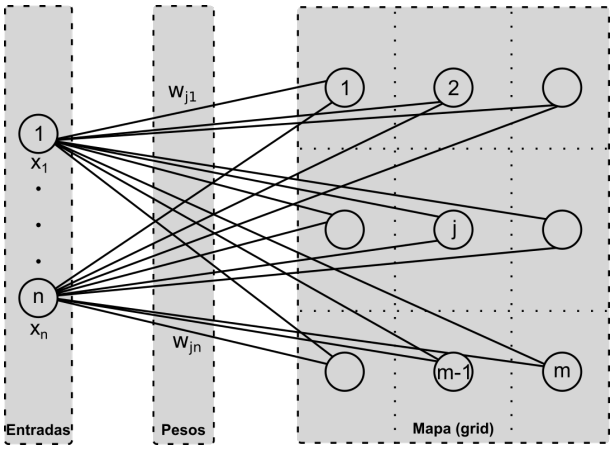


Figura 2.8: Entradas, pesos e o mapa do *Self-Organizing Map*

Ao escolher uma entrada aleatoriamente, calculamos a distância euclidiana entre o valor da entrada x_i e o seu peso correspondente, através da equação 2.3. Considere n, o número de

[1] grid: um padrão regularmente espaçado horizontalmente e verticalmente

nós (ou neurônios), x_i o valor correspondente a uma dada entrada $x = \{x_1, x_2, \ldots, x_m\}$, com m componentes e $w_{m,n}$ o peso correspondente a entrada n do vetor para o nó m.

$$d_m = \sqrt{\sum_{i=1}^{n} (x_i - w_{m,i})^2} \tag{2.3}$$

Uma vez calculada a distância deste vetor de entrada x_i para todos os nós, a entrada será atribuída àquela que possuir menor distância calculada. Após isso, os pesos são atualizados para todos os nós, utilizando a equação 2.4. t representa a época sendo avaliada; i e j são neurônios; $I(x)$ representa o neurônio vencedor; $\eta(t)$ é a taxa de aprendizagem do algoritmo e pode variar ao decorrer das épocas ou se manter constante durante toda avaliação; $T_{j,I(x)}$ representa a topologia da vizinhança, variando de acordo com a distância entre nós vizinhos e o tamanho da vizinhança.

$$\Delta w_{m,i} = \eta(t) * T_{j,I(x)} * (x_i - w_{k,i}) \tag{2.4}$$

Para entender como o SOM funciona, é importante entender como a redução de dimensionalidade acontece, trazendo uma representação de dados multi-dimensionais para uma representação bidimensional. Essa redução é inteiramente dependente da distância entre neurônios, pois ela pode ser usada para mapear as distâncias entre os nós e os vetores de entrada em um plano bidimensional. Como o SOM calcula apenas a distância entre os nós e as entradas, fica claro que soluções são possíveis para o mesmo conjunto. Para um número grande de entradas e pequeno de nós, as entradas se agruparão e se adequarão ao redor deste pequeno conjunto de nós, indicadas pela distância euclidiana.

Dessa forma, uma vez adequadas as distâncias, o conjunto de entradas é distribuído entre os diversos nós do SOM, cada um representando um *cluster*. Dois parâmetros foram apresentados e que merecem atenção: a taxa de aprendizagem e o número de *clusters* (ou nós). A taxa de aprendizagem, $\eta(t)$, por ser um fator multiplicativo do peso, indica a rapidez com que os pesos serão ajustados. Uma taxa de aprendizagem com valores pequenos pode trazer a solução para mínimos locais, onde a solução ótima ainda não foi encontrada mas, para esta taxa de aprendizagem, não há outra solução possível. Para taxas de aprendizagem grandes, pode ocorrer um *overshoot* na busca da solução ótima: ao se aproximar da solução, o fator multiplicativo leva a próxima iteração para um erro muito maior do que se tinha na iteração anterior. Dessa forma, idealmente, a taxa de aprendizagem é adequada com o decorrer das iterações: no início, quando os nós estão muito desorganizados, uma taxa de aprendizagem alta é incentivada. Já quando estamos próximos à solução, taxas de aprendizagem menores são mais adequadas.

Já quanto ao número de *clusters* que devem ser utilizados não há uma panacéia, na qual um valor funcione para todas as análises possíveis. Esta análise está intrinsecamente ligada aos dados sendo analisados e devem ser avaliadas caso a caso. Obviamente, um número muito pequeno de nós levará a uma classificação e agrupamento menos discriminante. Já um número muito alto, levará a muitos *clusters* com poucas entradas associadas a ele, tornando a análise esparsa. Algumas análises indicam o uso da fórmula $NumeroDeNos = 5 * \sqrt{entradas}$ (Tian et al., 2014). Obviamente, o objetivo desta técnica é dar uma representação para os dados. Assim, busca-se reduzir ao máximo o número de nós, para simplificar o conjunto de dados, mas se reduzirmos muito, a variabilidade dos dados é perdida. Assim, a conclusão é que o número de nós deve ser avaliado para o conjunto de dados que quer se estudar.

2.3.3 Processamento de Sinais em Grafos

Diversas aplicações coletam e se utilizam de séries temporais: mercado financeiro, estações meteorológicas, novos casos em uma pandemia, etc. Estes dados são muitas vezes não

estruturados. Dessa forma, para realização de uma análise destes dados, o primeiro passo é extrair uma representação de baixa dimensão que descreva bem as interrelações entre diferentes séries temporais e sua intrarelações ao longo do tempo. (Mei e Moura, 2017) propôs uma forma de gerar um grafo, computacionalmente viável, que estime as estruturas de dados sobre as séries temporais. O grafo resultante é um grafo orientado e ponderado, possivelmente capturando relações causais entre as séries e não apenas correlações recíprocas.

Na análise de *botnets*, esta análise possibilita calcular uma influência entre os nós da rede, podendo-se inferir seu comportamento. Para tanto, o tráfego deve ser transformado em uma série temporal, coletando informações referentes ao tráfego ao longo de uma janela de tempo. Para uma dada característica extraída, uma série temporal pode ser formada. Uma janela de tempo adequada permite realizar captar na análise mudanças de comportamento ao longo do tempo, sem causar uma sobrecarga com diversas análises.

(Sandryhaila e Moura, 2013) propõe um *framework* determinístico capaz de realizar Predição Linear, Compressão de Sinal e Classificação de Dados. O *framework* é chamado de DSPg (*Discrete Signal Processing on Graphs*). A classificação de dados permite a captura de relações causais, possibilitando a classificação de dados de tráfego. O método para realização desta classificação é o *Causal Graph Process* (CGP). O CGP é um processo autoregressivo sobre uma série temporal, no qual seus coeficientes são filtros de grafos (Sandryhaila e Moura, 2013). Assim, ao analisar-se uma característica, uma série temporal para aquela característica, para cada um dos nós é formada. A técnica CGP evidencia as relações entre os nós com relação a essa característica. Dessa forma, espera-se, para o sucesso desta técnica, que o estado da entrada na série temporal no tempo k seja influenciado pela mesma característica no tempo $k-1$, sendo k a janela de tempo da série temporal, como é o caso em um ataque coordenado.

Desta maneira, o CGP calcula a estimativa de causalidade entre os nós da rede, para uma dada característica, durante o intervalo analisado. A vantagem do CGP é que ele não necessita de uma etapa de treinamento e/ou aprendizado, podendo ser utilizado diretamente sobre o tráfego analisado, diferentemente de abordagens de aprendizado de máquina supervisionado. Assim, o CGP permite a modelagem de relações causais entre os elementos através de cálculos autoregressivos sobre os dados da série temporal, provendo mais informações agregadas ao modelo. Como entrada para análise do CGP, toma-se a série temporal de cada um dos nós que serão avaliados quanto à característica proposta. A saída do método é uma matriz de adjacências NxN, onde N é o número de nós que estão sendo avaliadas naquela janela de tempo. Os valores da matriz indicam a magnitude de influência de um nós sobre outro. Calcula-se o grau de influência entre os nós identificados no tráfego desconhecendo *a priori* a relação entre os dados. Porém, uma característica em uma janela de tempo k é influenciada de alguma forma pela característica em uma janela de tempo $k-1$. Esta é a influência que a autoregressão identifica, explicitada na matriz de influências $\mathbf{A}$. A Equação 2.5 descreve o detalhamento matemático para determinar a matriz de influências (Sandryhaila e Moura, 2013).

$$\begin{aligned} x\,[k] &= w\,[k] + \sum_{i=1}^{M} Pi(\mathbf{A},\mathbf{c})x\,[k-1] \\ &= w\,[k] + \sum_{i=1}^{M}\sum_{j=0}^{i}(c_{ij}\mathbf{A}^{j})x\,[k-1] \\ &= w\,[k] + (c_{10}\mathbf{I} + c_{11}\mathbf{A})x\,[k-1] + \\ &(c_{20}\mathbf{I} + c_{21}\mathbf{A} + c_{22}\mathbf{A}^{2})x\,[k-2] + \ldots \\ &+(c_{M0}\mathbf{I} + \ldots + c_{MM}\mathbf{A}^{M})x\,[k-M] \end{aligned} \tag{2.5}$$

onde k é uma amostra, x*[k]* é o valor da característica no tempo k, $Pi(\mathbf{A}, \mathbf{c})$ é um polinômio da matriz em **A** de ordem i (isto é, $Pi(\mathbf{A}, \mathbf{c})$ são filtros (Sandryhaila e Moura, 2013)); w*[k]* é ruído estatístico, utilizado na avaliação de precisão da autoregressão; c_{ij} são coeficientes polinomiais escalares, sendo $\mathbf{c} = (c_{10}\ c_{11}...c_{ij}...c_{MM})^{(T)}$ um vetor de todos os c_{ij}, e M é a ordem da autoregressão (Mei e Moura, 2017). Assim, pode-se notar que todos os termos apresentados são conhecidos, exceto a matriz de influências **A**, que indica a influência entre os nós.

Dada a existência de correlação entre os dados, temos um grafo G_1=(V,**A**) que define esta correlação, com V os vértices que correspondem aos nós de rede e **A** desconhecido, que define essa correlação. Para determinar **A**, emprega-se:

1. Resolver para $R_i = P_i(\mathbf{A},c)$: Aplicam-se os filtros de grafo tendo como entrada a série temporal

2. Recuperar a estrutura de **A** com uma das duas abordagens: usando $\hat{\mathbf{A}}$=$\hat{\mathbf{R}}$ como em

$$\hat{\mathbf{R}}_i = \underset{\hat{\mathbf{R}}}{\operatorname{argmin}} \frac{1}{2} \sum_{k=M}^{k-1} \parallel x[k] - \sum_{i=1}^{M} \mathbf{R}_i x[k-j] \parallel_2^2 + \lambda_1 \parallel vec(\hat{\mathbf{R}}_1) \parallel_1 + \lambda_3 \sum_{j \neq i} \parallel [\mathbf{R}_i, \mathbf{R}_j] \parallel_F^2 \tag{2.6}$$

   ou usando todos os $\hat{\mathbf{R}}_i$ em conjunto para encontrar **A**,

$$\hat{\mathbf{A}} = \underset{\mathbf{A}}{\operatorname{argmin}} \parallel \hat{\mathbf{R}}_1 - \mathbf{A} \parallel_2^2 + \lambda_1 \parallel vec(\mathbf{A}) \parallel_1 + \lambda_3 \sum_{i=2}^{M} \parallel (\mathbf{A}, \hat{\mathbf{R}}_i) \parallel_F^2 \tag{2.7}$$

3. Estimar todos os c_{ij} em uma das duas maneiras: estimando **c** ou a partir de $\hat{\mathbf{A}}$ e $\hat{\mathbf{R}}_i$ como em

$$\hat{\mathbf{c}}_i = \underset{\mathbf{c}_i}{\operatorname{argmin}} \frac{1}{2} \parallel vec(\hat{\mathbf{R}}_i) - \mathbf{Q}_i \mathbf{c}_i \parallel_2^2 + \lambda_2 \parallel \mathbf{c}_i \parallel_i \tag{2.8}$$

   onde

$$\mathbf{Q}_i = (vec(\mathbf{I}) vec(\hat{\mathbf{A}}) ... vec(\hat{\mathbf{A}}^i)), \mathbf{c}_i = (c_{i1} c_{i2} .... c_{ii}) \tag{2.9}$$

   ou de $\hat{\mathbf{A}}$ e os dados **X** como em

$$\hat{\mathbf{c}}_i = \underset{\mathbf{c}}{\operatorname{argmin}} \frac{1}{2} \parallel \mathbf{Y}(\hat{\mathbf{A}}) - \mathbf{B}(\hat{\mathbf{A}}) \mathbf{c} \parallel_F^2 + \lambda_2 \parallel \mathbf{c} \parallel_i \tag{2.10}$$

Ao final deste processo, os graus de correlação entre os nós são identificados na matriz de adjacências **A**. Quanto maior a coordenação entre os nós, ou seja, quanto maior o grau de correlação entre eles, maior a possibilidade de ser identificada uma *botnet* (Mirkovic et al., 2002). A magnitude dos valores na matriz de adjacência indica a coordenação entre os nós. Quanto mais distante do zero, maior é a influência entre os nós identificados na linha e na coluna da matriz.

2.4 RESUMO

Neste capítulo foram apresentados os fundamentos que embasarão a proposta do BotFetcher. Foram apresentadas e detalhadas as *botnets*. O ciclo de vida de uma *botnet* se inicia com a concepção e desenvolvimento do seu *software*. Nesta etapa é planejada a arquitetura,

métodos de comunicação, métodos de propagação, o propósito da *botnet* e métodos de evasão de detecção. A arquitetura de uma *botnet* pode ser centralizada ou descentralizada (hierárquica ou P2P). Para a *botnet* ser efetiva, ela deve possuir um grande número de dispositivos em sua rede. Assim, os métodos de propagação angariam membros para a rede, através de identificação e ataque aos dispositivos vulneráveis, realizando a infecção do dispositivo.

A comunicação dos *bots* com os servidores C&C se inicia com a descoberta (*rallying*), podendo esta descoberta ser feita, a princípio, via nome de domínio ou endereço IP. Uma vez que o servidor C&C é descoberto pelo *bot*, a comunicação pode se utilizar de diferentes protocolos e mecanismos para manter a comunicação segura, ativa e não identificável por monitores de rede, tais como o uso de criptografia. Uma vez iniciado o ataque, diversos comandos são enviados pelo *botmaster* aos `bots` para a efetivação do ataque. Estes comandos podem de acordo com o propósito daquela *botnet*, buscando-se o envio de *spam*, realização de ataques de negação de serviço e/ou roubo de informações. Todas as ações são realizadas buscando evadir a detecção de *softwares* anti-vírus (que executam no dispositivo hospedeiro) e monitores de rede (que podem ser executados distribuidamente, monitorando diversos dispositivos).

Embora busquem se esconder e se esquivar de técnicas de detecção, o tráfego dos *bots* utiliza a infraestrutura de rede existente, com protocolos e formas de comunicação bem definidos e consolidados. Assim, foi apresentado os modelos que padronizam a comunicação na Internet. O tráfego, formado pelos pacotes de rede, possui cabeçalhos, rodapés e conteúdo dos quais é possível realizar a identificação de informações. Estas propiciam uma análise de comportamento dos nós mesmo sem a análise do conteúdo daquele tráfego. Esse conjunto de características, uma vez observadas condições específicas, pode sinalizar um possível ataque de negação de serviço, antes que o ataque tenha de fato iniciado e seus efeitos sentidos na rede.

Com um conjunto de características que possam ser informativas para aquele tráfego, pode-se realizar uma análise da relevância destas características. Em um determinado tráfego, algumas características podem ser mais discriminantes que outras. Três métodos para a seleção destas características existem: *filter*, *embedded* e *wrapper*. *Filter* se utiliza apenas dos das características, sem um algoritmo de classificação, *embedded* se utiliza de um algoritmo de classificação em conjunto com um algoritmo de seleção e *wrapper* se utiliza de algoritmos de seleção que inerentemente são, também, classificadores. Assim, pode-se selecionar características relevantes em um tráfego gerado continuamente.

O levantamento das características se dá para permitir a realização da classificação dos nós do tráfego. Três métodos de classificação são relevantes citar neste trabalho: métodos de aprendizagem de máquina não-supervisionados, métodos de aprendizagem de máquina supervisionados e métodos determinísticos. Nos métodos de aprendizagem de máquina não-supervisionados, as características são apresentadas ao método sem que haja um treinamento prévio e um conhecimento de um rótulo (que identifica o que está sendo classificado), pelo algoritmo classificador. Assim, este método nos permite inferir similaridades das características, permitindo um melhor conhecimento dos dados, sem a realização da classificação.

O *Self-Organizing Map* é uma técnica baseada em redes neurais que reduz e discretiza a dimensionalidade dos dados, permitindo uma melhor análise e compreensão dos dados analisados. Métodos de aprendizagem de máquina supervisionados possuem um rótulo que permitem ao método treinar seu algoritmo de modo a aprender a reconhecer um determinado padrão ou comportamento. Métodos determinísticos não necessitam de uma etapa de treinamento e/ou aprendizado, tornando-o diferente dos métodos de aprendizagem de máquina. O *Causal Graph Process* permite a classificação de dados de tráfego pela captura de relações causais. Este método, assim, estima a causalidade no comportamento dos nós, propiciando a detecção dos *bots*.

3 TRABALHOS RELACIONADOS

Neste capítulo, são descritos e discutidos trabalhos relacionados à proposta da dissertação. Espelhando-se em uma revisão sistemática, buscou-se reunir os trabalhos relevantes nos temas abordados nas seções seguintes: definição e análise de *botnets* e métodos de detecção de *botnets*. Dentro dos métodos de detecção de *botnets*, foram exploradas as técnicas que foram utilizadas na arquitetura do BotFetcher. Na Seção 3.2, fez-se um estudo dos textos que apresentavam as características das *botnets*: como as classificar, como se comunicam, seus propósitos e forma de agir. Abordou-se, também, a classificação das técnicas de detecção, como se utilizam do tráfego, como identificam o comportamento de um *bot* e como se distribuem na rede. A seguir, na Seção 3.3, são abordadas, as técnicas que efetivamente foram utilizadas para a proposta da arquitetura BotFetcher. A Tabela 3.1 resume os trabalhos expostos neste capítulo, comparando as técnicas utilizadas na proposta com as demais literaturas.

Elencaram-se sete temas que permeiam e subsidiam a proposta: Extração de Características de Grafos, Extração de Características do Tráfego, Predição e Sinais Precoces, Seleção de Características, Aprendizagem de Máquina Não-Supervisionada, Aprendizagem de Máquina Supervisionada e Detecção por Método Determinístico. Vale ressaltar aqui que, embora a arquitetura BotFetcher não se utilize de Aprendizagem de Máquina Supervisionada, o tema foi discutido por ser um tema de pesquisa frequente na comunidade científica. Portanto, esta técnica é apresentada aqui para que o leitor possa observar onde a proposta se adequa na literatura atual. Nota-se que, por vezes, estes trabalhos são apresentados separadamente, constituindo um tema de pesquisa apenas, que foi abordado e incorporado, dentro dos seus limites para o presente trabalho. Outros trabalhos, porém, incorporam mais de uma das técnicas apresentadas e, assim, são discutidas.

3.1 METODOLOGIA DA REVISÃO

Uma das metodologias de revisão bibliográfica comumente utilizadas para identificar, selecionar e analisar artigos científicos e outras publicações é a revisão sistemática. A revisão sistemática é uma metodologia de investigação científica com métodos de identificação, seleção e análises sistemáticos. O objetivo é, após a sua realização, ter uma revisão crítica e abrangente da literatura (Cordeiro et al., 2007). Para tanto, define-se uma série de restrições na busca de modo que os resultados podem ser replicados e comparados por demais pesquisadores ao se deparar com o trabalho. Como resultado de uma revisão sistemática, tem-se a possibilidade de uma publicação do tipo *survey*, onde são apresentados os trabalhos analisados e suas classificações dentro do escopo do problema tratado.

Entretanto, a revisão sistemática, por ser abrangente, exige um considerável dispêndio de tempo para ser realizado, sendo melhor aconselhável para mestrandos que já tenham o seu escopo definido desde o seu início ou, melhor ainda, dentro de um projeto de doutorado. Ela pode abranger revisões primárias, secundárias e terciárias, onde se busca expandir a análise e o estudo apenas nos textos principais, mas também em revisões e *surveys*. Embora os trabalhos doravante apresentados não tenham passado pelo escrutínio e rigor científico de uma revisão sistemática, tomou-se esta metodologia como guia e espelho para a apresentação dos trabalhos.

Sendo assim, embora não seja uma revisão sistemática com o rigor científico esperado, algum critério é melhor do que nenhum critério. Palavras e fragmentos de texto foram escolhidos dentro de cada tema de modo a propiciar resultados relevantes na análise. Os trabalhos foram

	Características de Grafos	**Características do Tráfego**	**Predição e Sinais Precoces**	**Seleção de Características**	**ML Não Supervisionada**	**ML Supervisionada**	**Método Determinístico**
(Neira et al., 2020)		✓	✓		✓	✓	
(Dakos e Lahti, 2013)			✓				✓
(Pelloso et al., 2018)		✓	✓				
(Collins et al., 2007)		✓	✓				
(Husák, 2020)	✓		✓				
(Moussa et al., 2019)		✓		✓			✓
(Ferreira e Nogueira, 2018)		✓					✓
(Lagraa et al., 2017)	✓					✓	
(Chowdhury et al., 2017)	✓				✓	✓	
(Daya et al., 2019)	✓				✓	✓	
(Wang et al., 2019)	✓	✓		✓	✓	✓	
(Dong et al., 2020)		✓				✓	
(Burghouwt, 2015)	✓					✓	✓
(Cai et al., 2020)	✓						✓
(Kayacık e Zincir-Heywood, 2006)		✓			✓		
(Ismail et al., 2020)		✓		✓		✓	
(Hossain et al., 2020)		✓		✓		✓	
(Joshi et al., 2020)		✓		✓	✓	✓	
(al Qerem, 2020)		✓		✓		✓	
(Iglesias e Zseby, 2015)		✓		✓	✓	✓	
BotFetcher	✓	✓	✓	✓	✓		✓

Tabela 3.1: Comparação dos Trabalhos de Detecção de *Botnets*

encontrados nas principais e mais relevantes revistas e conferências científicas na área de computação e adjacentes. Além disso, indexadores e buscadores foram utilizados de modo a expandir os resultados da pesquisa. Os trabalhos mais recentes tiveram prioridade na seleção para demonstrar o estado da arte. Trabalhos mais antigos que possuem intersecção com o tema e surgiram da busca também foram apresentados, dada a sua relevância ou por motivos históricos.

3.2 BOTNETS

Botnets são ferramentas com propósitos únicos. São compostas por uma rede de dispositivos compromissados por um *malware*, causando uma série de ameaças à Internet. As pesquisas e publicações pertinentes ao assunto usualmente classificam, definem e discutem *botnets* em conjunto com maneiras de detectá-las. Nestes artigos são apresentadas características das *botnets*, arquiteturas, identificação, maneiras de comunicação. A primeira *botnet* a ganhar notoriedade foi identificada no ano 2000. Um *spammer* enviou mais de 1 milhão de emails ao

longo de um ano. Também, utilizou-se da *botnet* para coletar informações de cartões de crédito e praticar fraude (Journals, 2003). Embora este tenha sido o primeiro ataque a ganhar notoriedade, somente a partir de 2003 começaram a surgir ataques de *botnets* mais frequentes e consistentes.

(Wainwright e Kettani, 2019) apresenta o conceito de *botnets*, seu histórico, as arquiteturas comumente observadas nas *botnets* e como elas se desenvolvem ao longo do seu ciclo de vida. As fases do ciclo de vida consistem na sigla CRIME (a sigla se mantém na Língua Inglesa): Concepção, Recrutamento, Interação, Marketing e Execução. Além disso, apresenta modelos de ataques de *botnets*, criticando a ênfase na detecção dos *bots* e não na sua modelagem e de como operam. Defende que caso modelos matemáticos fossem estabelecidos, possíveis defesas poderiam ser testadas antes do ataque (ao invés de durante ou depois). Usualmente, se testam as ferramentas de detecção para avaliar a acurácia, ignorando a sua adequação a um dado modelo a seu uso pretendido. Assim, apresenta os modelos: epidemiológico, de aprendizado de máquina, estocástico, de teoria dos jogos, *bayseanos* não-paramétricos e de grafos, correspondendo cada um destes modelos com o seu ciclo de vida proposto (CRIME).

3.2.1 Detecção de Botnets

No início dos estudos de ataques por *botnets*, as conexões domésticas eram, em quase sua totalidade, discadas (*dial-up*). Como as velocidades de transmissão, o volume trafegado e a quantidade de dispositivos eram baixos, os primeiros estudos para detecção de *botnets* podiam avaliar continuamente todo o tráfego entre os dispositivos, sem se preocupar com a quantidade de nós ou como agrupá-los. Neste sentido, (Binkley e Singh, 2006) propuseram um algoritmo de detecção de *bots* em que a entrada para a análise consistia de todo o tráfego da rede. O tráfego era comparado com perfis específicos, pré-computados, como a utilização do IRC para comunicação. (Binkley e Singh, 2006) também propôs uma heurística analisada sobre o protocolo TCP, que mostrou bons resultados à época, haja visto a ausência de mecanismos de evasão das *botnets* (Seção 2.1.5). Em (Karasaridis et al., 2007), foi proposto um algoritmo de detecção e caracterização de *botnets* que foi utilizado por um provedor de acesso *Tier-1* à Internet. (Karasaridis et al., 2007) também utilizou-se como entrada todo o conjunto de dados na análise. A detecção ocorria em *links* específicos, analisando características da camada de Transporte do tráfego, analisando portas de conexão, comparando a comunicação com um modelo e uso de heurísticas. Novamente, vê-se a utilização de heurísticas que poderiam ser facilmente evitadas pelos desenvolvedores das *botnets*.

Para que as atividades maliciosas causadas pelas *botnets* sejam interrompidas, é necessário o desenvolvimento de um mecanismo robusto para melhorar a detecção, a análise e o processo de remoção das *botnets*. (Karim et al., 2014) apresenta uma revisão sistemática das técnicas de detecção de *botnets* e seu estado-da-arte, apresentando-as de maneira abrangente e compreensiva. A Figura 3.1 apresenta a taxonomia proposta pelos autores, permitindo que as técnicas de detecção de *botnets* sejam classificadas e analisadas quanto aos seus aspectos qualitativos críticos de sua proposta. Por fim, apresenta tendências de pesquisas passadas e futuras, verificando um consistente aumento de interesse em detecção de *botnets*, indicando desafios persistentes e proeminentes que continuam em aberto.

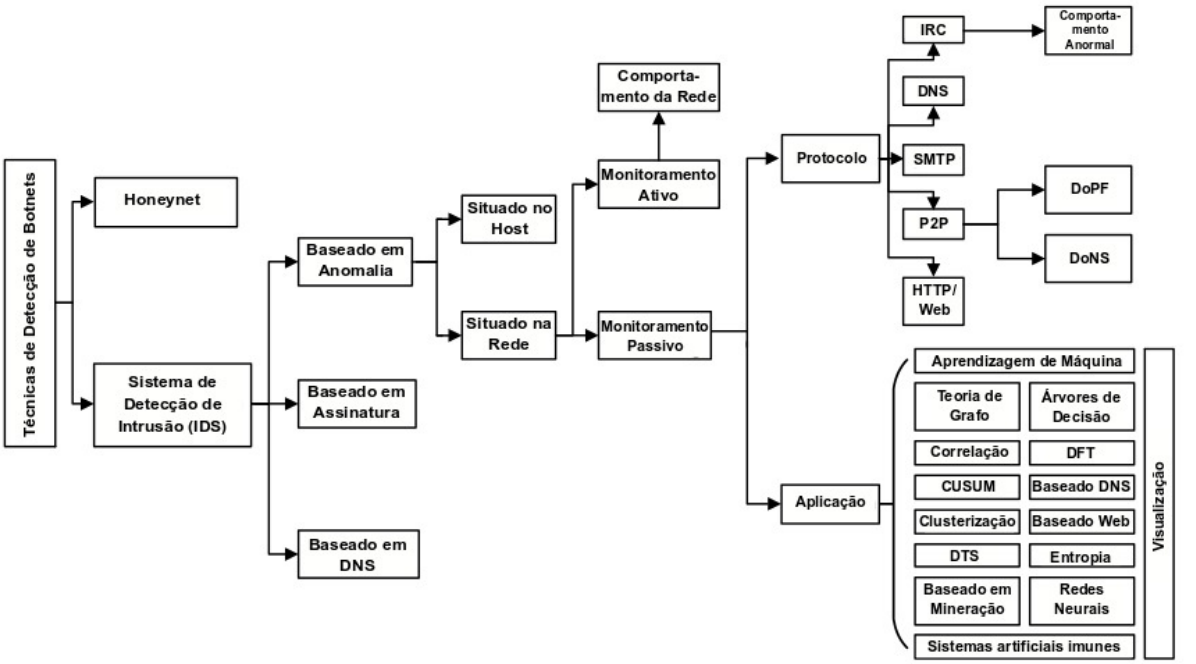


Figura 3.1: Taxonomia de técnicas de detecção de *botnets* apresentada por (Karim et al., 2014)

3.3 TÉCNICAS DE DETECÇÃO

Nesta seção, são apresentados trabalhos que se utilizam técnicas de detecção de *botnets* e que tenham alguma similaridade com a proposta desenvolvida. Alguns trabalhos levantados se adequam a mais de uma das categorias abaixo apresentadas, por se utilizarem de técnicas de seleção de características, aprendizagem de máquina supervisionada e não-supervisionada, por exemplo, tais como (Wang et al., 2019; Daya et al., 2019). Dessa forma, optou-se por apresentar o trabalho na categoria que mais ênfase deu-se na sua contribuição ou, então, caso o trabalho seja equânime em sua proposta, a última etapa da detecção foi considerada para sua apresentação.

3.3.1 *Early Warning Signals* e Predição

Predição é o ato de saber, antecipadamente, de um evento, baseado em um conhecimento específico (Huifang Feng e Yantai Shu, 2005). Assim, nesta subseção são apresentados trabalhos que buscam identificar a presente de *botnets* enquanto o seu ataque não começou. Esta detecção pode se dar nas etapas de propagação, quando a rede de *bots* está se expandindo, na descoberta, quando os *bots* tentam se comunicar entre si e/ou com o *botmaster*, se agrupando e coordenando para iniciar o ataque. (Neira et al., 2020) propõe o sistema ANTE, para identificação ANTecipada de botnEts, com base em algoritmos de aprendizagem de máquina. O sistema ANTE busca identificar as *botnets* antes que os efeitos de um ataque seja identificado, capturando o tráfego e realizando a análise de antecipação, identificando os possíveis *bots* naquela captura. Entretanto, apresenta diversas técnicas, sem que se haja uma maneira de selecionar qual a melhor para uma dada análise, de maneira automática.

Antes de que transições críticas (tais como o início de um ataque) ocorram, pode-se esperar que sinais precoces sejam identificados. Em (Dakos e Lahti, 2013), é discutido o que pode-se esperar destes sinais precoces, propondo, também, uma ferramenta para o cálculo destes sinais. Quando os sistemas chegam próximos de transições críticas, a tendência de que ele permaneça no mesmo estado diminui, assim como sua resiliência: uma pequena perturbação é

capaz de modificar o estado do sistema. Os autores discorrem sobre o que os sinais precoces, definindo-os como propriedades simples do sistema que variam de maneiras específicas antes de uma transição crítica. Com a ferramenta proposta, diversas características são extraídas de modo a permitir avaliar o comportamento de um sistema.

(Scheffer et al., 2001) interpreta e analisa transições críticas, sugerindo que o modelo se comporta como um "modelo mínimo de um ecossistema mostrando histerese". A estabilidade em sistemas pode ser um fenômeno local (quando uma pequena perturbação nos parâmetros não causa uma grande mudança no sistema, como um todo. Uma das maneiras de analisar este comportamento á com a análise do *Kendall tau*. (El-Shaarawi e Niculescu, 1992) apresenta a utilização do *Kendall tau* para detectar a presença de uma tendência em uma série temporal. (Pelloso et al., 2018) se utiliza do conceito de sinais de alerta precoces e suas características para a análise de *botnets* causadoras de ataques DDoS. (Pelloso et al., 2018) é capaz de identificar uma transição crítica na rede antes que um ataque seja iniciado. Entretanto, não identifica quais os nós ou dispositivos da rede estejam causando tal comportamento. BotFetcher é capaz de identificar os comportamentos anômalos causadores de ataques DDoS de cada nó.

Os *bots* são dispositivos que foram compromissados por um agente ou *software* malicioso. (Collins et al., 2007) apresenta o conceito de "impureza", que indica a propensão de dispositivos na rede serem compromissados por agentes externos. Testando hipóteses temporais (redes impuras conterem dispositivos compromissados por mais tempo) e espaciais (tendência de nós compromissados de agruparem em redes impuras), os autores analisam e testam tais características durante ataques de diversas modalidades (*spam*, *phishing*, varredura e monitoramento). (Collins et al., 2007) conclui que é possível predizer a localização (espacial e temporal) de um *bot* em ataques de *spam* e varredura, podendo-se criar uma facilidade para o bloqueio destes nós.

(Husák, 2020) busca identificar ataques criando uma consciência sobre a situação da rede, compartilhando informações da rede, de forma a projetar ataques. Para tanto, apresenta um modelo de predição e projeção de ataques baseado em grafos. Os grafos podem ser utilizados para modelar o ataque, modelar a topologia da rede, grafos conectados permitem a detecção de padrões maliciosos e correlação de alertas podem se utilizar de grafos. (Husák, 2020) apresenta o CyGraph, um modelo de dados baseado em grafos para conscientização da situação cibernética de uma rede. Com estes grafos apresenta uma representação detalhada de diversos aspectos da rede, podendo cooperar de maneira preditiva com outras ferramentas.

3.3.2 Extração e Seleção de Características

A utilização de características representativas se tornou uma necessidade com o aumento no volume de dados sendo trafegado na rede. (Iglesias e Zseby, 2015) propôs um método de detecção de anomalias em redes (não necessariamente voltado a *botnets*) em que 41 características do tráfego podem ser extraídas e construídas a partir dos pacotes da rede. Uma vez elencadas estas características, propôs-se uma divisão em: básicas, que podem ser extraídas diretamente do pacote; de tráfego, que podem ser inferidas para o mesmo nó ou serviço; de conteúdo, que é extraída da análise do dado sendo trafegado. Obviamente, cada uma destas características tem o seu grau de dificuldade para ser extraída e analisada. Características textuais, por exemplo, exigem um tratamento antes de serem levadas a algoritmos de aprendizagem de máquina. De qualquer forma, não houve, neste trabalho, um esforço para identificar variáveis que seriam mais úteis em uma análise. Utilizou-se cabeçalhos e rodapés das diversas camadas de comunicação, assim como *flags* para constituir as 41 características que foram extraídas. Entretanto, é importante notar que foi proposta uma análise computacional estatística da relevância destas características.

(al Qerem, 2020) foca em seleção de características em dispositivos IoT. Com o aumento no número destes dispositivos, eles passam a ser alvos de agentes maliciosos, visto o aumento

no número de ataques de *botnets* causadas por dispositivos IoT, como a *botnet* Mirai. Assim, uma detecção precisa e em um tempo adequado é necessária para mitigar os efeitos do ataque. (al Qerem, 2020) realiza a seleção de características do tráfego utilizando algoritmos genéticos multi-fases e otimização por enxame de partículas para identificar o melhor subconjunto de características para a classificação dos *bots*. Para a classificação, *Random forest* é utilizado, sendo, a proposta capaz de identificar o comportamento anômalo. Porém, há de se notar que não houve testes em cenários diferentes e com outras bases de dados. Alguns trabalhos focam em seleção de características para uma determinada arquitetura, protocolo ou *malware* causador dos ataques. (Joshi et al., 2020) realiza a análise de seleção de características para a melhora na detecção de *botnets* P2P (*Peer-to-peer*). Utilizando-se de técnicas de aprendizado de máquina, (Joshi et al., 2020) compara técnicas para encontrar o melhor conjunto de características para uma determinada base de dados. São comparadas as técnicas: *Principal Component Analysis (PCA)*, seleção uni-variável (tal como K-means), seleção recursiva de características, árvores de decisão e matrizes de correlação. Porém, apenas uma base de dados foi utilizada com um conjunto inicial extremamente restrito de apenas 6 características. BotFetcher utilizou-se de diversas bases de dados, além de um conjunto de 19 características.

O uso de características relevantes e discriminantes melhora a performance de uma análise de aprendizado de máquina. Neste sentido, aumentar indiscriminadamente (e até artificialmente) o número de características não é interessante. Pelo contrário, realizar uma análise de quais características são mais relevantes torna a detecção e a clusterização mais acuradas. (Moussa et al., 2019) apresentou os fundamentos teóricos do α-*investing*$^{+}$, um algoritmo para a análise estatística das características do tráfego. O objetivo foi identificar características estatisticamente independentes para a detecção de ataques DDoS. Porém, as 18 características apresentadas na proposta tornam difícil a sua aplicação em posteriores técnicas de aprendizado de máquina. Dessa forma, uma extração de características oriundas do tráfego e seu tratamento poderiam agregar mais informações para a análise. Além disso, poderia se utilizar características oriundas de grafos, como indicado nos trabalhos anteriores deste capítulo. O trabalho não focou na detecção de *bots*. Este método de análise constitui na abordagem *filter* de seleção de características, conforme apresentado na Seção 2.2.3, e foi utilizada na proposta desta dissertação para identificação de características relevantes de *botnets*.

Criptografia de ponta a ponta está se tornando o padrão nas comunicações pela Internet, evitando que agentes que estejam encaminhando o tráfego sejam capazes de ler e armazenar o que está sendo trafegado. Entretanto, a análise de *botnets* fica comprometida pois se pudesse verificar a comunicação entre os *bots*, interromper sua ação seria trivial. Assim, (Ismail et al., 2020) realiza a análise de seleção de características em tráfegos de *botnets* criptografados, propondo uma técnica baseada nos cabeçalhos dos pacotes do tráfego. Assim, uma análise do conteúdo do pacote não é necessária. Os autores realizam a classificação utilizando 7 características para treinamento, com 3 características para validação, com os resultados indicando um maior número de verdadeiros-positivos quando utilizadas apenas 3 características. Há de se notar neste trabalho o número baixo de características avaliadas, além da consideração do uso de tráfego criptografado, haja visto que esta é uma premissa comum e não pode ser desconsiderada nas comunicações atuais, não sendo um ponto forte "extra" do trabalho. Similarmente, (Hossain et al., 2020) se utiliza de cabeçalhos dos pacotes para a seleção de características, sem a análise do conteúdo do tráfego. Extrai 12 características e as avaliada com a técnica de redes neurais, através do método *wrapper* (a seleção e classificação atuam em conjunto). (Hossain et al., 2020) avalia a técnica na base de dados Botnet 2014, que é um agrupamento de outras três base de dados, conseguindo uma boa acurácia. Pesa contra o fato de que poderia ter avaliado um número maior de características e em outras bases de dados.

3.3.3 Aprendizagem de Máquina Não-Supervisionado - Clusterização

Nesta dissertação, utiliza-se a técnica de aprendizagem de máquina não-supervisionada *Self Organizing Map*, SOM, para a realização de agrupamento de nós com comportamento similar. (Kohonen, 2013) apresentou esta técnica identificando dois dos principais algoritmos e formas de calibrá-los. No trabalho, foram apresentadas diversas áreas de aplicação para SOM, como a análise de dados exploratória, que é aplicada neste trabalho. A utilização do SOM é baseada nos resultados obtidos em (Daya et al., 2019) que utilizaram este algoritmo para o agrupamento de nós de rede, apresentando melhores resultados quando comparado com outras técnicas de aprendizado de máquina não-supervisionado, tais como *k*-Means e DBScan. A utilização desta técnica para o agrupamento de nós de rede por comportamento também foi utilizada em (Chowdhury et al., 2017). (Kohonen, 2013) apresenta os fundamentos teóricos que subsidiam a implementação do método e propiciam resultados satisfatórios para o agrupamento de dados, sem focar, no entanto, no seu uso para tráfego de rede.

Com o avanço do poder computacional, a utilização de técnicas de aprendizagem de máquina começou a ser proposta. Trabalhos realizando a extração de características do tráfego e o agrupamento de nós, com técnicas de clusterização começaram a ser publicados mais recentemente. (Chowdhury et al., 2017) realizou a extração de sete características de um grafo formado a partir do tráfego. Após a extração, os nós da rede são agrupados por similaridade de suas características no grafo formado. A técnica utilizada foi a *Self-Organizing Map*. Cada agrupamento, permitiu a análise em um conjunto de dados menor, permitindo uma análise mais rápida. A detecção de *bots* utilizou a técnica de aprendizagem de máquina supervisionada *Support Vector Machine*. Entretanto, (Chowdhury et al., 2017) utiliza apenas características formadas a partir do grafo, não considerando características da rede. Também, assume como premissa que o maior *cluster* dos nós é necessariamente um *cluster* com nós benignos (sem *bots*) o que pode não ser verdadeiro dependendo dos parâmetros utilizados na análise.

(Kayacık e Zincir-Heywood, 2006) utiliza a técnica de aprendizagem de máquina não supervisionada *Self-Organizing Map* (SOM) para classificar o tráfego com ataques desconhecidos, tomando por base, ataques conhecidos. Os autores desenvolveram modelos de ataque de modo a propiciar uma melhor identificação de similaridades entre ataques. Utilizando-se a técnica *Self-Organizing Map* (SOM), uma relação entre ataques conhecidos e um ataque desconhecido (com suas características) é criada. A saída da análise é uma representação bidimensional que identifica o comportamento de ataques desconhecidos. Os resultados mostram que ataques desconhecidos, mas com comportamento parecido a um ataque conhecido foi agrupado junto, no mesmo *cluster*, no *grid* bidimensional. Embora o resultado tenha se mostrado satisfatório, (Kayacık e Zincir-Heywood, 2006) utilizou-se da base de dados KDD'99 que possui bastante críticas na literatura (Brugger, 2007). Baseado na análise de (Brugger, 2007; Tavallaee et al., 2009; Saporito, 2019; Guillén et al., 2014), a base de dados KDD'99 é conceitualmente inadequada, sem que uma conclusão possa ser extraída de qualquer experimento utilizando-a. Além disso, (Kayacık e Zincir-Heywood, 2006) se utiliza uma técnica de aprendizagem de máquina não supervisionada para classificar o tráfego que é rotulado. Assim, uma técnica de aprendizagem de máquina supervisionada se adequaria melhor à análise, tal como Redes Neurais.

3.3.4 Técnicas de Análise em Grafos e Causalidade

Com a melhora na infraestrutura de rede, conexões com banda larga e a disseminação global do acesso à Internet, deu-se a necessidade de identificar características mais representativas no tráfego, de modo a melhorar a detecção de *botnets*. A análise através de heurísticas e com a avaliação de todo o tráfego se mostrou insuficiente para detectar agentes maliciosos. Assim,

recentemente, estudos propuseram mecanismos de análise baseados em grafos, com avaliação das características que são utilizadas na análise e buscando agrupar os nós por comportamento, eliminando a necessidade da análise de todo o tráfego e propiciando uma melhora na performance de detecção. Em (Lagraa et al., 2017), os autores realizaram uma análise na forma de grafos e propuseram o BotGM. Os grafos foram formados pela sequência de dados trafegados pela rede. A sequência de pacotes de tráfego de rede formam pequenos grafos, com a utilização de características provenientes do tráfego, tais como endereços de origem e destino, portas de origem e destino, o protocolo utilizado e o total de pacotes. No total, 10 características foram apresentadas. Com a utilização destas características, os grafos eram formados e suas formas eram comparadas com grafos que, conhecidamente, possuíam um comportamento benigno, pré-definido ou treinado. Dessa forma, o BotGM era capaz de detectar *bots* através da identificação de *outliers*. A detecção de *outliers*, porém, espera um comportamento normal da rede, fazendo a detecção quando o comportamento desvia do padrão esperado, podendo levar discrepância na acurácia do método. Também, embora este trabalho tenha proposto uma avaliação sobre as características, o fez de maneira empírica, sem propor um método de identificação do melhor conjunto que foi proposto.

A utilização de grafos pode ser feita para a análise de mecanismos de defesa, da visualização da rede ou, ainda, para a visualização do comportamento e disposição dos *malwares*. Em (Cai et al., 2020), coletam-se os *malwares* de base de dados públicas e gera-se um grafo dinâmico para analisar a análise dos *malwares*. Foi possível identificar que os nós infectados se agrupam em *clusters* ao invés de uma rede única, havendo uma correlação positiva entre o número de *malwares* na rede e: o tamanho da rede e o número de eventos de segurança relatados. Também foi notado que componentes de infraestrutura de rede (tais como roteadores, *hubs*, *switches* e *bridges*) desempenham papéis relevantes na dinâmica de distribuição dos *malwares*. Porém, a proposta analisa e correlaciona os eventos e *malwares* a posteriori, não se utilizando da análise para uma classificação dos possíveis nós infectados na rede.

Em (Burghouwt, 2015), os autores apresentam técnicas de detecção de servidores de comando e controle para *bots* em redes corporativas. Uma das técnicas apresentadas, denominada Causalidade em Fluxos de Tráfego (TFC, *Traffic Flow Causality*) detecta o tráfego dos servidores de comando e controle pelas causas direta do tráfego de saída da rede. A técnica permite que seja implantada como um sistema de detecção de intrusão, em tempo real, identificando as causas diretas do fluxo do tráfego. O tráfego é organizado em fluxos bidirecionais e exige que todos os fluxos bidirecionais sejam iniciados por alguma atividade do cliente. Caso isso não aconteça, um sistema de detecção de anomalias pode atuar, bloqueando o tráfego. Entretanto, há de se notar que a rede corporativa possui, usualmente, total controle do administrador de rede, sendo, portanto, de mais fácil ação tanto nos nós (por exemplo, fazendo análise do *host* para *malwares*) quanto da rede, sendo capaz de suprimir e interromper o tráfego da rede. Embora este seja um cenário que BotFetcher seja capaz de operar, BotFetcher também é capaz de realizar a detecção em um cenário mais diverso e amplo como a Internet.

Em (Sandryhaila e Moura, 2013), os autores propuseram um *framework* determinístico para utilizado na área de Predição Linear, Compressão de Sinal e Classificação de Dados. Para a classificação de dados, o trabalho foi estendido para (Mei e Moura, 2017) que propôs a técnica *Causal Graph Process*, CGP. O CGP identifica inter-relações em séries temporais e intra-relações em uma série temporal ao longo do tempo. Como resultado, é gerado um grafo representando as relações causais entre pares na série temporal. A aplicação desta técnica para a identificação de *botnets* geradoras de ataques *DDoS* foi apresentada em (Ferreira e Nogueira, 2018). Entretanto, a análise era feita em todo o conjunto de dados fornecido, não se preocupando em dividir o tráfego em *clusters* menores e nem realizar uma análise estatística para identificar quais características seriam importantes na análise. Com a identificação das variáveis relevantes, o conjunto de

dados a ser analisado é reduzido. Além disso, a análise em *clusters* diminui a necessidade de processamento em um conjunto massivo de dados e possibilita a paralelização da detecção. A redução da necessidade de processamento propicia a aplicação desta técnica em cenários reais, oferecendo escalabilidade e eficiência na identificação das *botnets*.

3.3.5 Aprendizagem de Máquina Supervisionado

Em (Daya et al., 2019), os autores executaram a extração de características de grafos através do tráfego entre os nós. Após a extração de características, é aplicado uma clusterização e posterior detecção com uma técnica de aprendizado de máquina supervisionado. (Daya et al., 2019) compara técnicas tanto na clusterização quanto na detecção, identificando as que possuem melhores resultados. Porém, o conjunto de características utilizado é apenas oriundo do grafo, ignorando as que porventura pudessem ser extraídas e construídas a partir dos pacotes do tráfego. Além disso, utiliza, esse conjunto é fixo, não realizando uma análise de quais características seriam mais relevantes para a análise, fazendo-as variar de acordo com o tráfego gerado. A base de dados utilizada para testes foi a CTU-13. Eles utilizaram um cenário (dos 13) para a avaliação e sintonização de parâmetros, utilizando estes parâmetros na análise de demais cenários. Entretanto, tal abordagem não poderia ser tomada como ideal para todos os cenários, devendo os parâmetros serem avaliados em cada cenário.

Em (Dong et al., 2020), observa-se o BotDetector, uma técnica de inteligência artificial que utiliza *deep learning* para classificar os *bots* do tráfego. O BotDetector é baseado em aprendizagem de máquina extrema (*extreme learning machine* – ELM), defendendo que as características não precisam ser extraídas, podendo o algoritmo atuar diretamente no *stream* de dados aprendendo a partir dele, sem processamento dos dados. A técnica de aprendizagem de máquina extrema é uma proposta melhorada da rede neural com uma camada escondida (*single-hidden layer feedforward neural network* – SLFN). Muito embora defenda que o algoritmo do BotDetector possa atuar diretamente no tráfego, sem a necessidade de uma engenharia de características, os autores não discorrem sobre como isto é feito. Além disso, ao apresentar o algoritmo, há uma etapa de pré-processamento dos dados, antes da classificação. A entrada para o algoritmo é uma matriz bidimensional extraída do tráfego, sendo o vetor de características criado na etapa de treinamento. Os experimentos conduzidos indicam que o algoritmo possui uma boa performance, sendo capaz de identificar *bots* com poucos recursos computacionais e tempo, sendo aplicável para IoT. Entretanto, vale-se notar que o uso de redes neurais é custoso para o treinamento, sendo a sua classificação computacionalmente "barata".

Em (Wang et al., 2019), apresenta-se o BotMark, um sistema de detecção de *botnets* que se utiliza de características híbridas, provenientes do fluxo do tráfego e de um grafo baseado no tráfego, sendo extraídas 15 características do fluxo do tráfego e 3 características do grafo. A análise das características do tráfego e do grafo é feita separadamente, sendo atribuídos valores de pontuação para cada saída. A pontuação geral é utilizada para marcar os *bots*. Para a avaliação baseado no tráfego, *k-means* é utilizado, atribuindo pontuações para similaridade (baseado na distribuição dos tamanhos dos pacotes em um fluxo). As características baseadas no grafo são avaliadas utilizando mínimos quadrados e Fator Local de Ponto Fora da Curva (*Local Outlier Factor* – LOF), atribuindo pontuações para avaliações consideradas anômalas quando comparadas com os seus vizinhos. Assim, o BotMark avança literatura apresentando um algoritmo que se utiliza tanto de características de grafo quanto de tráfego, testando o algoritmo em 5 cenários de *botnets*. Porém, o uso de *k-means* faz um agrupamento dos nós, mas esse agrupamento é também uma avaliação. Além disso, peca em avaliá-las separadamente, com técnicas distintas.

3.4 RESUMO

Botnets se tornaram uma real ameaça a operação da Internet. Esta rede de dispositivos infectados por *malwares* foram criadas para conduzir atividades ilegais em larga escala, inclusive tomando o controle da operação de serviços públicos e privados em diversos países ao redor do mundo. Os trabalhos que tratam da solução usualmente realizam considerações sobre as características das *botnets*. Entretanto, alguns trabalhos mais focados e mais extensos neste ponto foram apresentados. Identificou-se as etapas do ciclo de vida de uma *botnet*, seus mecanismos de infecção e propagação, seus propósitos, formas de evasão de detecção, forma de comunicação, arquitetura e protocolos. Também, foi apresentada uma revisão dos mecanismos de detecção, concentrando a análise em detecção de *botnets* através do monitoramento passivo da rede, buscando realizar a detecção de anomalias.

Como pode ser observado na Tabela 3.1, nota-se uma ausência de trabalhos utilizando métodos determinísticos para detecção de *bots*, enquanto prevalece a utilização de aprendizagem de máquina supervisionada. Também, a predição de possíveis ataques com a identificação de *bots* na rede também é observada em um número diminuto de trabalhos. Em contrapartida, há em todas as categorias elencadas, a abordagem para identificação de *bots* ou identificação de ataques de negação de serviço distribuídos, sem, porém, integrá-los em uma arquitetura distribuída e única, capaz de identificar um possível ataque *DDoS*, além de identificar, individualmente, os nós maliciosos presentes na rede causadores de um ataque.

Os trabalhos apresentados que concernem à detecção de *botnets*, concentram-se em utilizar os conceitos apresentados e discutidos de modo a propor um mecanismo de detecção. São apresentados trabalhos que focam no monitoramento da rede de maneira passiva, identificando características do tráfego que permitam a análise de comportamento para classificar comportamentos maliciosos. Alguns textos utilizam-se características oriundas de grafos formados a partir do tráfego originário da rede. Outros, discutem características extraídas dos pacotes do tráfego, sumarizando e elencando uma série de características que auxiliam em definir um comportamento de um nó na rede. A detecção de ataques de maneira precoce também é abordada, de modo a identificar um possível ataque antes que ele se inicie. Diversos trabalhos de detecção focam em técnicas de aprendizagem de máquina supervisionada e não-supervisionada, com resultados satisfatórios na detecção de *bots* no tráfego analisado.

Buscou-se elencar e discutir artigos que sejam recentes, de modo a capturar o estado da arte tanto na caracterização de *botnets* quanto na sua detecção. Para tanto, foram discutidos os trabalhos que apresentam e propõem técnicas que serão utilizadas como subsídios para a presente proposta. Assim, técnicas *early warning signals*, α*-investing*$^{+}$, *Self-Organizing Map* e *Causal Graph Process* são apresentadas, indicando a literatura condizente e realizando a análise de benefícios do uso destas técnicas e possíveis pontos de melhoria.

4 BOTFETCHER

Este capítulo detalha o BotFetcher, uma arquitetura para predição de ataques de negação de serviço distribuídos e detecção de *bots*, capaz de identificar a mudança no estado das redes, predizendo ataques de negação de serviço distribuído. O BotFetcher é composto de cinco módulos: (*i*) Módulo de Coleta de Dados, (*ii*) Módulo de Extração de Características, (*iii*) Módulo de Processamento, (*iv*) Módulo de Análise e (*v*) Módulo de Notificação, conforme ilustra a Figura 4.1. Nas Seções 4.1 a 4.5, são descritos os módulos e seus componentes, detalhando suas funcionalidades e como os componentes interagem com outros módulos. Na Seção 4.6, é apresentado como esta arquitetura se disporia e operaria em um cenário hipotético, instanciando os módulos e componentes em dispositivos de rede reais.

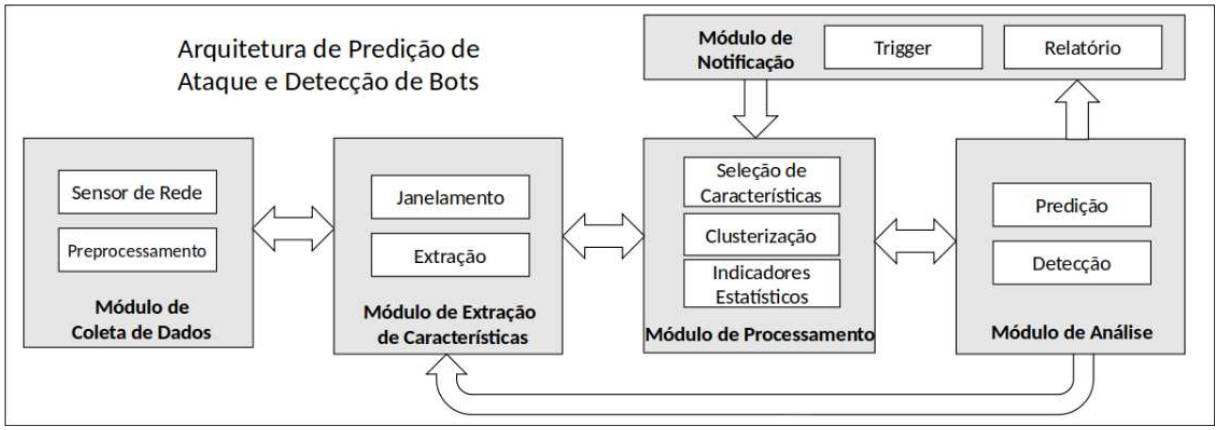


Figura 4.1: A Arquitetura BotFetcher

4.1 MÓDULO DE COLETA DE DADOS

O Módulo de Coleta de Dados integra sensores de rede e componentes de pré-processamento. Os sensores de rede continuamente capturam o tráfego da rede, utilizando-se de um sensor ou um *sniffer*. Os componentes de pré-processamento pré-processam o tráfego coletado da rede, transformando dados binários (*bots*) em dados capazes de serem processados por ferramentas (e também legível para humanos). Dessa forma, estes dados irão ser entradas relevantes para o Módulo de Extração de Características.

4.1.1 Sensor de Rede

Sensores de rede são elementos capazes de monitorar um valor medido na rede, tais como o tráfego em uma dada porta, o uso do CPU de um servidor ou o armazenamento disponível em um disco. Aqui, o interesse está nas informações do tráfego e quais dados podem ser extraídos dos pacotes trocados na rede (tanto UDP quanto TCP). Alguns sensores permitem que o tráfego seja capturado e armazenado em disco, sendo possível a reconstrução do tráfego e a simulação de um cenário condizente com o tráfego experimentado em determinada rede. Além disso, pode-se realizar a filtragem dos pacotes de acordo com o seu protocolo, tamanho, origem e destino, propiciando uma abrangente análise da rede.

Os dois tipos de sensores mais comuns são os *sniffers* de pacotes (em tradução livre, "farejador" de pacote) e os sensores NetFlow. Os *sniffers* capturam e decodificam os pacotes,

colocando o seu conteúdo em saídas como arquivos, enviando a um outro dispositivo para análise ou apenas saída visual para o operador. O sensor é capaz de analisar todo o pacote, desde os cabeçalhos até o conteúdo. Evidentemente, caso o conteúdo seja criptografado, este é inacessível de maneira legível pelo *sniffer*, mas, também, pode ser armazenado para futura tentativa de descriptografar os dados. Para que este sensor seja capaz de capturar o tráfego, o tráfego deve passar por ele, atuando como um intermediário na transmissão. Os sensores NetFlow são utilizados principalmente por dispositivos Cisco (o protocolo NetFlow foi desenvolvido por esta empresa). Os sensores NetFlow não atuam coletando e armazenando pacotes da rede, mas sim fluxos. Os fluxos são criados a partir das informações do tráfego IP, tais como endereços de origem e destino, duração da comunicação e janelas temporais. A partir dos dados NetFlow, visualiza-se o comportamento do tráfego da rede e seu volume.

Ambos os dispositivos necessitam da informação do tráfego de rede. Assim, tem-se um maior volume de informações capturadas e processadas em roteadores de borda e *gateways*. Na arquitetura proposta, é nestes dispositivos em que as informações do tráfego são capturadas. Entretanto, o tráfego capturado na rede é binário. Portanto, de pouco uso e de difícil análise se não for decodificado. Na Subseção 4.1.2, aborda-se esta decodificação e como os dados são convertidos para os Módulos de Processamento e Análise.

4.1.2 Pré-processamento

Os dados coletados das redes de computadores são transmitidos de forma binária. Uma codificação e decodificação pelo remetente e recipiente da mensagem trocada é necessária para que a informação seja legível pelas aplicações, em ambos os lados da troca da mensagem. Como o componente que analisa o tráfego não é, a princípio, parte das trocas de mensagens, o tráfego binário é capturado. Para que os Módulos de Extração de Características e demais sejam capazes de utilizar dados do tráfego, um pré-processamento é feito. O pré-processamento decodifica os dados binários em dados capazes de serem lidos pelos demais módulos.

Uma maneira de decodificar o tráfego de rede é capturando-o e armazenando-o em um arquivo. Este arquivo é uma captura completa e independente dos pacotes capturados previamente em um dado tempo. Outra possibilidade é a captura em tempo real, podendo alimentar os Módulos subsequentes e realizar a análise e a possível atuação na rede de maneira quase que imediata na rede. A habilidade de decodificar e analisar o tráfego é igualmente importante quanto a habilidade de coletá-lo. Os resultados obtidos e apresentados no Capítulo 5 se utilizaram de arquivos para a análise do tráfego, por constituírem de bases de dados *offline*, já coletadas.

Dependendo do modo em que o tráfego é capturado pelo sensor de rede e o protocolo utilizado, uma ferramenta específica pode ser necessária para esta decodificação. Aqui, utilizou-se o *tcpdump* como ferramenta de análise dos pacotes. Esta ferramenta é capaz de ler o tráfego binário armazenado em arquivos ou, também, capturar o tráfego diretamente de uma interface de rede. O *tcpdump* permite exibir e filtrar pacotes e seus diversos cabeçalhos, podendo-se visualizar as origens e destinos dos pacotes, os tamanhos destes e os protocolos utilizados nas diversas camadas. Estas informações são capturadas da interface de rede ou lidas de um arquivo (dependendo de como a análise se dá) e armazenadas em um arquivo que será lido pelo Módulo de Extração de Características, que prosseguirá a análise. Esta transmissão para o Módulo de Extração de Características pode também ser feita apenas na memória do dispositivo de análise em que a arquitetura está executando. Prefere-se a opção pelo arquivo, pois possibilita uma análise mais aprofundada caso necessário e a possibilidade de replicar os resultados.

4.2 MÓDULO DE EXTRAÇÃO DE CARACTERÍSTICAS

O Módulo de Extração de Características tem como entrada os dados pré-processados pelo Módulo de Coleta de Dados e continuamente divide-os em janelas, *i.e.*, conjuntos de dados geralmente determinados por espaços de tempo ou quantidade de elementos (por exemplo, número de pacotes de rede). Este módulo também realiza a extração de características que descrevem o tráfego daquela janela avaliada. Ele reúne informações do tráfego de rede e seus protocolos, sumariza e quantifica alguns destes dados, construindo uma característica que é representativa daquela tráfego.

4.2.1 Janelamento

Um aspecto que deve ser levado em conta na realização da análise de tráfego é quanto a geração dos dados. Uma análise de tráfego, implementada em um sistema real, necessita que a solução seja capaz de operar sobre um *stream* de dados, que são gerados continuamente (sem um início e um fim pré-determinado), em alta frequência e em ambientes dinâmicos. Porém, as soluções existentes operam, normalmente, com limites de início e fim bem definidos, tomando um conjunto finito de entradas e proporcionando uma saída. Os modelos de decisão devem ser capazes de incorporar novas informações à medida que os dados cheguem e detectar mudanças, se adaptando em função dos novos dados. Além disso, comportamentos antigos, que não são mais relevantes devem ser esquecidos. Os seguintes aspectos são importantes com relação aos sistemas de aprendizagem no contexto de *streams* de dados: uso da memória, detecção de mudança e adaptação do modelo.

Como o resultado é reportado a cada intervalo de tempo, mesmo em um sistema de análise contínuo, buscou-se uma maneira de realizar a análise em um intervalo de tempo. A abordagem mais comum na realização de análises em *streams* de dados é o uso de janelamento. As janelas são deslizantes e tomam como entrada na análise o conjunto de dados naquele intervalo de tempo especificado pela janela. Dados que estão fora da janela acabam por se tornarem inutilizados, enquanto dados novos ganham relevância. Em um sistema dinâmico tal fato é esperado, haja visto que deve-se priorizar a informação nova, que pode indicar uma mudança de comportamento em detrimento ao comportamento antigo, que pode não mais ser adequado.

O tamanho da janela influi no desempenho do classificador. Uma janela pequena traz para o classificador uma adaptação rápida a mudança de conceito, porém em cenários mais estáveis pode afetar o desempenho do classificador. Além disso, exige um número maior de classificações, incorrendo em maior custo computacional. Já uma janela grande é boa em cenários estáveis, porém demora para detectar mudanças de conceito. Neste trabalho, foram utilizadas janelas de tempo para a análise dos dados, buscando levar em consideração tanto a velocidade da análise quanto a dinâmica do contexto. Este janelamento é importante para a formação de séries temporais, que são utilizadas na técnica de processamento de sinais em grafos (Subseção 2.3.3). Também, foram avaliadas as bases de dados como um todo, considerando-a como uma única janela que provém os dados à análise.

4.2.2 Extração de Características

Neste etapa diversas características são extraídas do tráfego de rede e outras são formadas a partir do tráfego. Dois tipos de características são extraídas de todo o tráfego: (i) características do tráfego propriamente dito e (ii) características que buscam evidenciar as relações entre os nós. As características extraídas do tráfego são oriundas de cabeçalhos e rodapés das diversas camadas do protocolo de comunicação (Figura 4.2). O dado da camada de Aplicação é encapsulado na

camada de Transporte onde recebe cabeçalhos específicos para esta camada. O mesmo acontece nas camadas inferiores de Rede e Enlace.

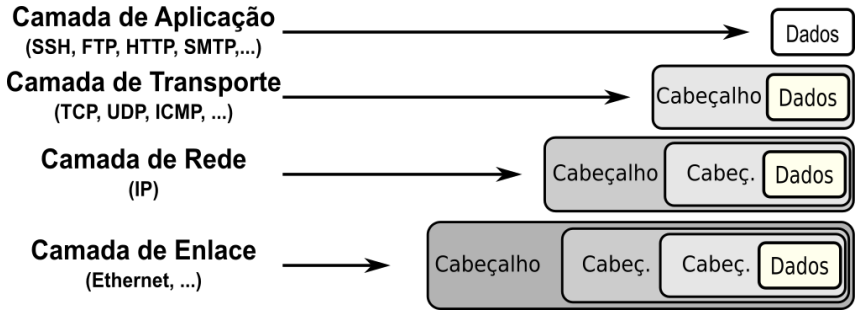


Figura 4.2: Cabeçalhos, rodapés e dados das diversas camadas de comunicação

Como estas informações são observáveis na comunicação, monitores da rede as capturam e enviam ao BotFetcher. O BotFetcher processará estas informações e as transformará nas características (*features*) utilizadas na análise. Como o intuito do método é detectar os nós que possuem comportamento de *bot*, as características serão associadas ao endereços deles. Cada nó, associado a um endereço IP, formará o conjunto de características e sumarizadas na Tabela 4.1. Com exceção de redes locais que se utilizam do protocolo *ARP* (camada de Enlace), associado ao endereço MAC (*Media Access Control*), toda a Internet é baseada no protocolo IP (*Internet Protocol*), a associação de um nó a um endereço IP é adequada.

A Figura 4.3 apresenta as diversas informações que podem ser extraídas dos pacotes IP (na camada de Enlace) e dos pacotes TCP e UDP (na camada de Transporte). O pacote mais simples é o UDP, que tem como informações apenas tamanho do pacote e portas de origem e de destino como informações relevantes para análise. *Checksum* é um bloco de dados para verificação de erros na mensagem, não sendo relevante para análise devido a aleatoriedade inerente a este dado, assim como a mensagem transmitida em si. Já o protocolo TCP apresenta 11 informações que podem ser incorporadas à análise. Isso porque este protocolo apresenta recursos de garantia de entrega e outras funcionalidades inexistentes no protocolo UDP. Entretanto, algumas destas informações também são irrelevantes, como o conjunto de *Flags* e, também, o *Checksum*, por exemplo. Esses pacotes TCP e UDP são encapsulados no protocolo IP, que, por si só, também apresenta uma śerie de informações a serem avaliadas – 12 no total. Algumas destas, no protocolo IP podem ser ignoradas, como a versão do protocolo e as *flags*. Entretanto, algumas podem ser incorporadas à análise, como o TTL (*Time-To-Live*) do pacote e os endereços IP de origem e destino do pacote, que é utilizada no BotFetcher para identificar os nós. Porém, extrair estas características e utilizá-las sem nenhum tratamento é impraticável, pois aumenta significativamente o dicionário de possibilidades para cada característica. Por exemplo, pode-se ter até 65535 portas de origem e destino e inúmeros tipos de protocolo, para algumas das informações ali presentes. Tempos e tamanhos, também, podem ter uma gama grande de valores. Assim, uma sumarização, agregação e quantização foi realizada em cada característica.

A quantidade de protocolos contabiliza quantos protocolos cada nó utiliza para se comunicar com outros nós, da camada de Aplicação como HTTP, FTP, SSH, OpenVPN, IRC, DHCP, DNS, etc. Os protocolos ICMP, TCP, UDP e DNS são corriqueiros. O percentual do total de pacotes em que cada um destes protocolos é utilizado é contabilizado em *Percentual ICMP*, *Percentual TCP*, *Percentual UDP* e *Percentual DNS*, respectivamente. Certos ataques gerados por *botnets* utilizam prioritariamente destes protocolos, *e.g.,* o *Ping Flood* (protocolo ICMP) e Dyn Attack (protocolo DNS) e, portanto, estes percentuais são relevantes para a análise de *botnets*. Caso outros protocolos sejam utilizados (inclusive os da camada de Aplicação), o

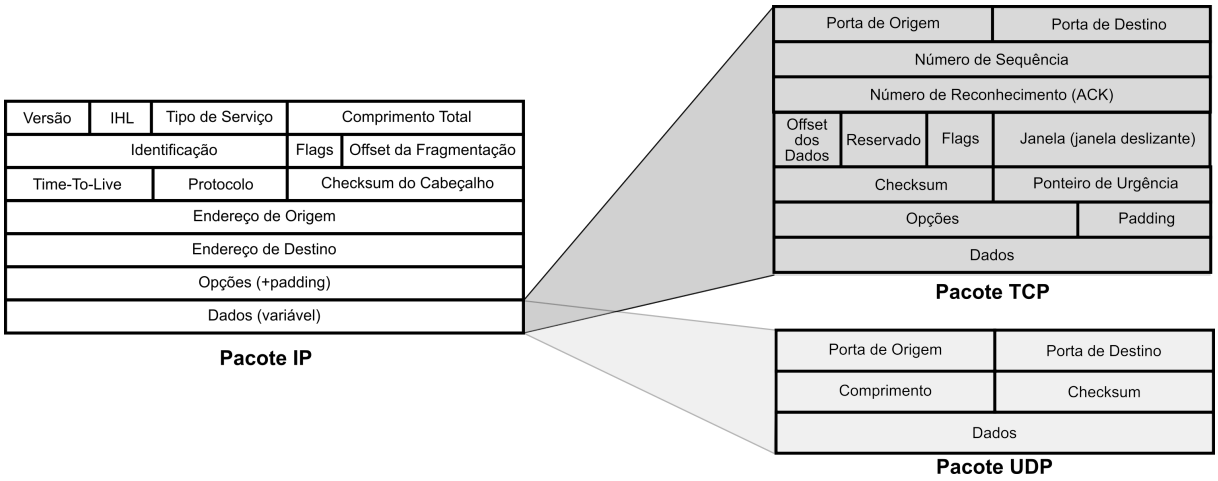


Figura 4.3: “Anatomia” dos Pacotes TCP/IP e UDP/IP

percentual é contabilizado no parâmetro *Percentual Outros*. Para as características *Tamanho da Janela TCP* e *Tamanho do Quadro (Frame Length)*, as médias de seus valores para cada nó são calculadas, pois podem apresentar características que se diferenciam de outros nós em caso de comunicação por *bot*. O *tamanho da janela TCP* indica a quantidade de *bytes* enviados antes de um *ACK* ser recebido e o *tamanho do quadro* se refere ao quadro da camada de enlace. Certos tipos de *bots* se beneficiam de privilégios no dispositivo infectado e enviam pacotes a partir de portas privilegiadas (menores que 1024), enquanto outros só conseguem enviar a partir de portas não privilegiadas (maiores que 1024). As características *Portas de Origem Privilegiadas* e *Portas de Origem Não Privilegiadas* contabilizam a quantidade de pacotes que tem origem em portas privilegiadas e não privilegiadas, respectivamente. Também, a quantidade de portas de destino é contabilizada para discriminar *bots* de outros nós, pois alguns *bots* se comunicam prioritariamente com uma determinada porta de destino. A média do valor de TTL (*Time To Live*), *TTL Médio* do pacote é calculada pois os *bots* podem tentar se esconder enviando pacotes com TTLs muito pequenos ou muito grandes.

Além das características do tráfego, outras são extraídas buscando identificar as relações entre os nós. BotFetcher cria um grafo direcionado de modo a evidenciar essas interações. A cada pacote trocado entre o nó origem e o nó destino, os endereços IP de origem e destino são capturados e serão vértices do grafo construído pelo BotFetcher. Uma aresta direcionada é criada do nó de origem para o nó de destino com base nos endereços IP de origem e destino indicados em cada pacote. Havendo uma aresta existente entre as mesmas origem e destino, outra aresta não será adicionada. Os endereços de origem e destino dos pacotes enviados e recebidos são capturados e enviadas ao BotFetcher que realizará a construção de um grafo G_2=*(V,*A_r*)*, onde V é o conjunto de endereços IP e A_r, conjunto de arestas indicadas pelo par ordenado endereço IP de origem e endereço IP de destino de um pacote do tráfego. Cabe ao BotFetcher construir uma visão geral da rede através do grafo. Dessa forma, quando todo o tráfego for analisado, o conjunto de vértices do grafo consistirá no conjunto de nós de rede que trafegaram dados. As arestas direcionadas terão como origem o nó de rede que origina o pacote, identificado pelo seu endereço IP, tendo como destino o vértice (outro nó da rede) que o pacote tem como endereço IP de destino. Desse grafo, as seguintes características serão extraídas para buscar caracterizar o comportamento dos nós: *out degree*, *in degree*, *out degree weight*, *in degree weight*, *local clustering coefficient*, *node betweenness centrality* e *eigenvector centrality*.

In degree e *out degree* são a quantidade de nós que se comunicam com um dado nó. *In degree* indica a quantidade de nós que enviaram ao menos um pacote para o dado nó, enquanto

Características	Descrição
Quantidade de Protocolos	Número de protocolos que um nó se comunica
TTL Médio	*Time to live* médio dos pacotes enviados
Tamanho da Janela TCP	Tamanho da janela no protocolo TCP
Percentual TCP	Percentagem de pacotes que utilizam protocolo TCP
Percentual UDP	Percentagem de pacotes que utilizam protocolo UDP
Percentual ICMP	Percentagem de pacotes que utilizam protocolo ICMP
Percentual DNS	Percentagem de pacotes que utilizam protocolo DNS
Percentual Outros	Percentagem de pacotes que utilizam outros protocolos
Portas de Origem Priv.	Quantidade de portas de origem que são privilegiadas (<1024)
Portas de Origem Não Priv.	Quantidade de portas de origem que não são privilegiadas (>1024)
Quant. de Portas de Destino	Quantidade de portas de destino de pacotes enviados por um nó
Tamanho do Quadro	Tamanho do quadro do pacote
In Degree	Caracerística de grafo definida no texto
Out Degree	Caracerística de grafo definida no texto
In Degree Weight	Caracerística de grafo definida no texto
Out Degree Weight	Caracerística de grafo definida no texto
Node betweenness centrality	Caracerística de grafo definida no texto
Local clustering coefficient	Caracerística de grafo definida no texto
Eigenvector centrality	Caracerística de grafo definida no texto

Tabela 4.1: Características extraídas do tráfego para cada nó

out degree indica a quantidade nós em que o nó enviou ao menos um pacote de tráfego de rede. *In degree* e *out degree* são relevantes, pois se muitos *bots* tentam contactar um único nó atuando como *botmaster*, pode ter um alto valor de *in degree* para o *botmaster*. *Out degree weight* e *in degree weight* são similares às características anteriores, se diferenciando, porém, por contabilizar a quantidade de pacotes trocados de e para um dado nó, respectivamente. A cada pacote trocado, havendo uma aresta existente com a mesma origem e destino, a quantidade de um pacote é somada ao valor atual, senão a aresta é criada e estas características são inicializadas. *Node betweenness centrality* quantifica o número de vezes que um nó atua como elo para o menor caminho entre dois outros nós. Quanto maior o valor do *node betweenness centrality*, mais central no grafo um nó é, podendo ser relevante para a detecção de *botnets* em que os nós se comunicam em uma estrutura descentralizada, como uma uma rede de *bots* P2P (Chowdhury et al., 2017). *Local clustering coefficient* indica quão concentrada é a vizinhança de um dado nó, buscando identificar quão próximos no grafo os vizinhos de um nó estão. Por fim, *eigenvector centrality* é um critério que busca evidenciar a influência de um nó sobre outros.

Todas estas 19 características são extraídas continuamente e normalizadas, tendo o seu valor no intervalo entre 0 e 1. Após isso, são associadas ao nó e compõem o vetor de características daquele nó. Esse vetor de características é utilizado para a seleção que retornará um vetor com apenas as características selecionadas.

4.3 MÓDULO DE PROCESSAMENTO

O Módulo de Processamento realiza a seleção de características, a clusterização e o cálculo dos indicadores estatísticos do tráfego. O Módulo de Processamento usa dados das janelas geradas no Módulo de Extração de Características e calcula as características que serão utilizadas na análise, os *clusters* e os indicadores estatísticos. O componente de seleção de características identifica as características mais relevantes no tráfego e gera como saída um

conjunto de características apropriado para a detecção de *bots*. O componente de clusterização agrupa os nós de rede com comportamento similar, se utilizando de uma técnica de aprendizagem de máquina não-supervisionada. Os indicadores estatísticos fornecem informação sobre uma mudança no estado da rede. O cálculo destes indicadores é mais rápido e menos custoso do que a realização de uma análise profunda em todo o tráfego. Estes podem indicar um ataque DDoS se, no entanto, realizar a análise de cada saída e característica avaliada.

4.3.1 Seleção de Características

As 19 características identificadas e extraídas do tráfego podem ou não ser relevantes para a análise de detecção dos *bots*. Note que foram extraídas apenas 19 que pressupõe-se apresentar algum auxílio na detecção de *botnets*. Este número poderia ser expandido ou reduzido, a depender do julgamento. Embora ter mais características pareça interessante, este aumento pode incorrer na Maldição da Dimensionalidade (Bellman, 1966) em técnicas aprendizado de máquina (utilizado na etapa de clusterização – detalhes na Subseção 4.3.2). Esta "maldição" indica a existência de um limite para melhorias no resultado decorrente do aumento no número de características. A partir de um dado momento, a adição de novas características não melhora o resultado da análise, podendo, inclusive, degradá-lo. Buscar um bom – e menor – conjunto de características discriminantes que descrevem os dados é um tema de grande relevância em aprendizagem de máquina. Isto se deve ao fato que as interações entre diferentes características são mascaradas por variáveis correlacionadas (Wichers, 1975).

Assim, um conjunto estatisticamente relevante de características deve ser buscado para as análises subsequentes. Um fator importante para a seleção de características é o critério de seleção. Tradicionalmente, os algoritmos de seleção de características são classificados de acordo com a forma que executam essa tarefa. Além disso, os algoritmos também podem ser categorizados de acordo com a natureza dos dados em que estão sendo aplicados. Como descrito na Subseção 2.2.3, existem três modos de seleção de características: *filter*, onde a análise das características sem a participação de um classificador; *wrapper*, onde a avaliação se dá em conjunto com o classificador; *embedded*, onde o classificador também propicia uma avaliação das características. Cada um dos métodos traz vantagens e desvantagens. Optou-se aqui pelo o uso de um método *filter* por propiciar mais flexibilidade ao BotFetcher, sendo um passo de pré-processamento, dando a liberdade de utilizar-se outros algoritmos tanto para seleção de características quanto para a detecção, por ser independente do algoritmo de aprendizagem de máquina. O BotFetcher utiliza como algoritmo para seleção o α-*investing*$^{+}$.

O objetivo da seleção de características é buscar características mais relevantes e informativas em relação aos dados. Ainda, (Guyon e Elisseeff, 2008) afirma que essa tarefa ao diminuir o conjunto de características reduz a dimensionalidade dos dados, melhorando o desempenho, reduzindo o custo computacional da análise e aprimorando o conhecimento sobre os dados. o algoritmo α-*investing*$^{+}$ (Moussa et al., 2019) (Algoritmo 1) realiza a seleção de características. Este algoritmo objetiva reduzir a taxa de falsos positivos na descoberta de novas características. Um falso positivo, nesta etapa, consiste em uma característica erroneamente selecionada e adicionada ao conjunto de características relevantes até um dado momento. No α-*investing*$^{+}$, uma característica é adicionada ao conjunto de características selecionadas se ela não tiver relação de dependência com alguma característica pertencente ao conjunto.

Para avaliar a relação de dependência, um limiar α ($l.5$ no Algoritmo 1) determina a probabilidade de incluir uma característica irrelevante ao conjunto. Além disso, um peso w ($l.2$) é atribuído representando o número de possíveis falsos positivos que podem ser incluídos ao conjunto ao longo das iterações do algoritmo. Um falso positivo, então, é uma característica com alguma relação de dependência com outra no conjunto de características selecionadas. O

Algoritmo 1: Algoritmo α-*investing*$^+$, extraído de (Moussa et al., 2019)

```
   Dados: Logs de rede
   Resultado: Conjunto de características selecionadas
 1 W ← [0.5]; // Conjunto de pesos
 2 w_0 ← W_0; // Peso inicial
 3 F ← {}; // Conjunto de características detectadas
 4 S ← {}; // Conjunto de características selecionadas
 5 α_Δ ← 0.5;
 6 i ← 1;
 7 enquanto novas características forem detectadas faça
 8     adicionarFeature(f_i, F);
 9     α_i ← w_i/2 * i;
10     para cada s ∈ S tal que s ≠ f_i faça
11         tabela ← gerarTabelaContingencia(f_i, m);
12         resultado ← chiQuadrado(tabela);
13         se resultado[valorP] ≤ α_i então
14             w_{i+1} ← w_i − α_i;
15         senão
16             adicionarFeature(f_i, S);
17             w_{i+1} ← w_i + α_Δ − α_i;
18         fim
19     fim
20     i ← i + 1;
21 fim
```

valor p é a probabilidade de duas características serem independentes e é obtido a partir de um teste estatístico. No caso do α-*investing*$^+$, o teste é o Chi-Quadrado. Dessa forma, para adicionar uma nova característica ao conjunto de características selecionadas, é necessário que o valor do teste estatístico associado a esta característica seja maior que o limiar α atual (l.13 a l.18). Neste caso, o peso w atual aumenta (l.17) e, no caso contrário, onde a característica é dependente de outra, w diminui (l.18). O valor de w afeta diretamente o limiar α (l.9), de forma que quanto mais características são adicionadas ao conjunto de características selecionadas, o limiar torna-se mais rígido para impedir a admissão de falsos positivos.

Para cada uma das 19 características extraídas, um vetor de características (com os valores das características) é formado para cada nó. O algoritmo tem como entrada os vetores de características para cada nó e é executado até que não haja detecção de novas características. No Botfetcher, as 19 características são continuamente geradas. O α-*investing*$^+$ oferece uma redução do conjunto de dados a ser analisado, melhorando o desempenho do método, sem que a representatividade dos dados seja comprometida. Ao fim da execução, obtém-se um conjunto de características que melhor representam o conjunto de dados. Este conjunto de características será utilizado pelo algoritmo de aprendizado de máquina não supervisionado para agrupamento de nós de rede que tenham comportamento similar neste conjunto.

É válido ressaltar que a etapa de seleção de características é precedida pela etapa de extração de características. Também que seria possível inverter a ordem da extração e da seleção de características. A seleção poderia ser feita no conjunto de dados extraídos diretamente de uma ferramenta de captura, sem nenhum tratamento (que é feito na extração). Isto porque o algoritmo é capaz de operar em dados categóricos e não-paramétricos. A seleção é capaz de identificar variáveis que seriam independentes nesse cenário. Entretanto, para a aplicação da clusterização é adequado o uso de características quantizadas e sumarizadas, tais como as

descritas na Subseção 4.2.2. Além disso, caso a seleção fosse realizada antes da extração, não haveria garantia de quais características de tráfego seriam selecionadas para a realização da extração. Assim, sumarizar, quantizar e agregar estas características poderiam tornar os dados incompletos ou a análise incoerente. Dessa forma, optou-se pela execução da extração de características para sua posterior seleção.

4.3.2 Clusterização

Nesta etapa, o BotFetcher agrupa os nós utilizando uma técnica de aprendizagem de máquina não-supervisionada: *Self-Organizing Map* (SOM). Esta técnica oferece uma redução de dimensionalidade nos dados, pois tem-se um conjunto grande de dados que serão agrupados em um conjunto menor de *clusters*. Os vetores de características representam os nós. O SOM não conhece, *a priori* a classificação das entradas, possuindo para realizar a análise, apenas os dados de entrada, sem indicação do que, de fato, se tratam (sua classificação). Nesta etapa, portanto, o objetivo é entender e extrair informações sobre os dados através do seu agrupamento.

Os vetores de características dos nós são agrupados analisando as características selecionadas no procedimento descrito na Subseção 4.3.1. Desta forma, cada nó possui um vetor de características como entrada para a clusterização. A técnica SOM calcula a distância euclidiana desse vetor de características com o centro de um *cluster* e a compara com de outros vetores de características (de outros nós). A distância euclidiana é calculada pela raiz quadrada da soma dos quadrados das diferenças dos termos dos vetores. Os termos dos vetores são os valores das características selecionadas. O nó é atribuído ao *cluster* cujo seu vetor de características seja mais próximo do centro daquele *cluster*. A cada iteração, os centros dos *clusters* são ajustados de modo que as entradas contenham a maior quantidade de nós com a menor distância euclidiana para o centro do seu *cluster*. As iterações ocorrem até que o centro dos *clusters* não seja mais alterado (e, portanto, os nós em cada *cluster* também não se alteram) ou até um limite de iterações ser atingido (na implementação descrita a seguir, o limite de iterações é 1000) (Subseção 2.3.2).

As distâncias euclidianas entre estes vetores são mensuradas para associar um nó a um *cluster*. Calculadas as distâncias, o SOM retornará um *grid* 5x5. Cada entrada deste *grid* corresponde a um *cluster*. Os nós (identificados pelos seus endereços IP), assim, passam a ser associados a um *cluster*. Um *cluster* é uma coleção de nós similares e diferentes de outros nós pertencentes a outros *clusters*[1]. Um nó é similar a outro se pertencer ao mesmo *cluster*.

Com a clusterização, a dimensão da análise passa da granularidade do número de nós para a de número de *clusters*. Reduzindo significativamente o espaço de análise, com um conjunto muito menor de nós em cada *cluster*. Em um curto espaço de tempo, em uma análise, pode-se ter milhares de nós se comunicando. Dessa forma, o conjunto de análise de milhares de nós é reduzido para alguns *clusters*, com um conjunto muito menor de nós em cada *cluster*. Assim, a vasta distribuição de dados de entrada é rearranjada em um conjunto finito de *clusters*. Os *clusters* são distribuídos em uma grade (*grid*) que possibilita compreender as relações entre os nós, especialmente quando se trata de um conjunto bastante volumoso de informações, como é o caso da análise de tráfego de rede. No BotFetcher, os nós serão agrupados em uma grade 5x5 (totalizando 25 *clusters*), utilizando-se as características de rede extraídas e selecionadas para identificar as similaridades, ou seja, nós cujos vetores de características pertençam a um mesmo *cluster*. Uma quantidade muito pequena de *clusters* tornará os dados poucos discriminantes, enquanto muitos *clusters* deixarão muitos deles vazios.

[1]Deve-se observar aqui que não se trata de *clusters* de computadores (um conjunto que computadores que trabalham em conjunto e podem ser vistos como um único sistema)

É esperado que alguns *clusters* contenham um número maior de nós do que os outros. Os com maior número de nós (representados pelo vetor de características) são identificados com o comportamento benigno na rede (Chowdhury et al., 2017). Comportamentos anormais/maliciosos são pouco frequentes no mundo real. Na base de dados CTU-13, cenário 5, mais de 99% dos nós não são infectados por *bots* (e em outros cenários desta mesma base, temos dados similares). Modelos simulados indicam até 50% de nós que podem ser infectados (Lan et al., 2009). Vale ressaltar que a infecção e atuação pelos nós infectados não é imediata e instantânea, ocorrendo ao longo do tempo. No caso do *ransomware* Wannacry, que infectou rapidamente diversas empresas e máquinas ao redor do globo, estima-se até 30% dos dispositivos que tinha a porta 445 TCP aberta a Internet (meio de infecção dos dispositivos) eram vulneráveis (Rapid7, 2017).

Portanto, como resultado desta etapa, tem-se um conjunto de *clusters*, com cada *cluster* contendo um grupo de nós de rede (identificados pelos seus endereços IP) que apresentam comportamentos similares, segundo o algoritmo. Assim, pode-se iniciar a análise de detecção pelo menor *cluster*, acelerando a detecção dos *bots*, com um custo computacional muito menor do que realizar a análise sobre todo o tráfego. Parte-se da detecção no menor *cluster*, seguido pelo segundo menor, até se esgotar toda a análise. Além disso, cada conjunto de nós em um *cluster* é independente dos nós de outro *cluster*, possibilitando que a análise seja realizada de maneira autônoma e paralelizada, propiciando um ganho na velocidade de detecção.

4.3.3 Indicadores Estatísticos

Os indicadores estatísticos são calculados utilizando-se o tráfego de redes locais, observado de maneira distribuída no tráfego global. Os quatro indicadores estatísticos utilizados são: taxa de retorno, autocorrelação, coeficiente de variação e assimetria, apresentados na Seção 2.3.1. Estes indicadores são calculados sobre a série temporal construída em uma janela de tempo, utilizando-se apenas uma característica extraída do tráfego: a soma do tamanho dos pacotes de uma janela, oriundo do Módulo de Extração de Características. O cálculo dos indicadores é leve em recursos computacionais, sendo realizado nas bordas das redes locais (como em roteadores), essa característica é fácil de ser extraída, haja visto que todo o tráfego tem de passar por esse dispositivo que deve, então, apenas contabilizar os tamanhos dos pacotes trafegados.

O conjunto proposto de indicadores é capaz de predizer o começo de um ataque de negação de serviço volumétrico (Pelloso et al., 2018). O seu cálculo deve ser feito de maneira rápida, haja vista a necessidade e atuação dentro de uma janela de tempo (antes que a próxima se inicie). Implementações em Python e em R existem para o cálculo efetivo destas características. Dentro de cada janela, os pacotes podem ser agrupados em intervalos, ignorando individualmente os pacotes individuais e suas estampas de tempo. Este agrupamento diminui o volume de dados a ser analisado sem perda na informação global utilizada (a soma do tamanho dos pacotes). Também, torna o cálculo dos indicadores computacionalmente mais barato, haja visto que o *array* de entrada, com pontos a serem analisados, reduz consideravelmente, possibilitando a análise dentro de uma janela de tempo e sem perda no resultado.

Este conjunto de indicadores estatísticos são capazes de indicar sinais precoces de uma mudança em um metaestado da rede. Essa mudança de metaestado na rede é associada a uma rede sob ataque de negação de serviço distribuído. Uma vez que os indicadores estatísticos são calculados, eles serão utilizados no Módulo de Análise, pelo componente de predição, para indicar uma possível transição crítica no estado da rede.

4.4 MÓDULO DE ANÁLISE

O Módulo de Análise continuamente realiza a predição de ataques e a detecção de *bots*. Ele tem como entrada os indicadores estatísticos calculados no Módulo de Processamento. Os *clusters* definidos no Módulo de Processamento serve como base para a detecção de *bots*.

4.4.1 Predição

O componente de predição irá analisar os indicadores estatísticos calculados no Módulo de Processamento, verificando as tendências nos indicadores estatísticos: taxa de retorno, autocorrelação, coeficiente de variação e assimetria. A tendência destes indicadores é avaliada pelo parâmetro *Kendall tau*. O *Kendall tau* é comumente utilizado para detectar a presença de uma tendência em uma série temporal de dados. Os resultados são obtidos utilizando média móveis, com uma ou duas janelas de atraso. A análise de apenas um destes indicadores ou de um par deles é usualmente inconclusiva, não sendo assertivo quanto a mudança de estado na rede, indicando possível um ataque. Caso um conjunto menor de indicadores fosse avaliado, o número de falso-positivos ou falso-negativos tornaria a análise inviável. Assim, a análise deve ser realizada no conjunto de todos os indicadores.

Os indicadores são avaliados através do Kendall tau, que varia de -1 a +1. O valor de Kendall tau positivo indica uma tendência de aumento na variável medida naquela janela, enquanto um valor negativo indica uma tendência de queda. Quanto mais próximo do valor absoluto unitário, maior é a tendência de queda ou aumento. Os indicadores estatísticos são comparados com a série temporal construída. Se a concordância entre a série temporal e a variável for perfeita (*i.e.*, as duas séries são a mesma), o valor de Kendall tau será +1. Se a discordância entre a série temporal e a variável for perfeita (*i.e.*, uma série é o inverso da outra), o valor de Kendall tau será -1. Se as séries forem independentes, então, o valor para Kendall tau será próximo de zero. A Figura 4.4 mostra estes indicadores estatísticos calculados para uma janela temporal de um tráfego, com os valores Kendall tau apresentados para cada indicador.

A avaliação dos indicadores busca detectar quatro condições em uma janela de tráfego: (i) a redução na taxa de retorno, (ii) o aumento na autocorrelação, quando medida com uma defasagem, (iii) um aumento no coeficiente de variação aumenta e (iv) um aumento na assimetria. Em (Scheffer et al., 2009), os autores demonstraram que estas condições nos indicadores estatísticos, quando avaliados em conjunto, indicam um comportamento específico de iminência de uma transição crítica. No conceito de redes, a rede está em um estado metaestável e um ataque DDoS produz uma perturbação no estado da rede. Assim, caso este prenúncio do ataque seja verificado, ele pode levar a indicação de mudança no comportamento da rede.

Avaliado o valor de Kendall tau para os quatro indicadores estatísticos e notada as condições indicando uma transição crítica na rede, o Módulo de Análise indicará (através do Módulo de Notificação) ao mecanismo central de detecção de *botnets* que uma transição crítica foi detectada em uma determinada rede. O Módulo de Processamento, assim, utilizará o tráfego das redes para então proceder a extração e seleção de características além da clusterização. A avaliação de predição ocorre de maneira distribuída, em redes locais, em um conjunto menor de nós avaliados que a rede de detecção de *botnets*, de maneira centralizada.

4.4.2 Detecção

Para a detecção de *bots*, utiliza-se como entrada os *clusters* identificados na etapa de clusterização (Subseção 4.3.2) e o tráfego dos nós presentes nos *clusters*. Assim, busca-se correlacionar o tráfego entre os nós e compreender as interrelações entre os dados coletados e

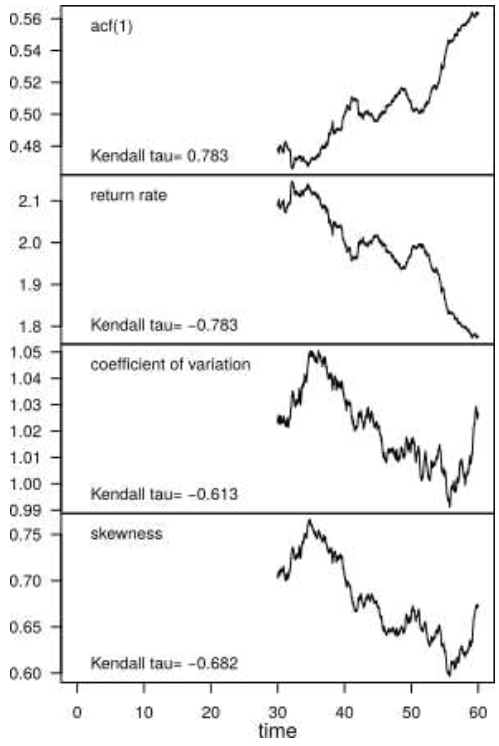


Figura 4.4: Indicadores estatísticos calculados para uma janela temporal de um tráfego

suas intrarelações ao longo do tempo. Neste processo, utiliza-se a técnica *Causal Graph Process*, CGP (detalhada na Subseção 2.3.3). O tráfego de cada nó é dividido em janelas de tempo, com as características definidas para cada intervalo, do início do período de avaliação até seu fim. Estas características, fragmentadas em intervalos de tempo regulares, formam uma série temporal para cada nó e cada característica. Assim o CGP realiza uma autoregressão sobre séries temporais. Os valores coletados ao longo do tempo são divididos em janelas de tempo regularmente espaçadas, de modo que o tamanho da amostra temporal é o mesmo para todos os nós avaliados e, consequentemente, que as séries contenham a mesma quantidade de dados. Das 19 características extraídas, cada uma possui uma série temporal correspondente. Para o BotFetcher avaliou-se a característica *out degree weight* (quantidade de pacotes enviados de um nó para outros), dado resultados prévios (Ferreira e Nogueira, 2018).

Definidas as séries temporais, a autoregressão é aplicada sobre a série temporal e estimará a matriz de influências. Os coeficientes da autoregressão são filtros de grafos que permitem reduzir ainda mais o espaço de busca dos dados (Sandryhaila e Moura, 2013). A matriz de influências permite capturar as relações causais entre os nós e suas correlações (Mei e Moura, 2017). A matriz de influências é uma matriz de adjacência ponderada que descreve o grafo, também ponderado e direcional. Assim, para cada *cluster* e seu conjunto de nós, as séries temporais são construídas e avaliadas pelo CGP. Como saída deste processo tem-se uma matriz de influências. Esta matriz de influências permite identificar as relações causais do tráfego gerado pelos nós e permitindo identificar o tráfego de *bots*. Cada *cluster* terá uma matriz correspondente. A correlação entre os nós de cada *cluster* é observada em sua matriz. Os termos da matriz correspondem ao peso atribuído à relação entre os dois nós identificados na linha e na coluna da matriz pelos seus endereços IP (os mesmos nós constam tanto nas linhas quanto nas colunas).

Como a detecção é realizada em cada *cluster*, o volume de dados a ser analisado é menor do que se todo o tráfego de todos os nós fossem avaliados entre si, como feito em (Ferreira e Nogueira, 2018). Na Figura 4.5, que apenas ilustra matrizes de influências, caso o tráfego dos

nós da rede não fosse segregado em *clusters*, a análise exigiria que os nós com endereços de *a* a *l* fossem analisados entre si. Neste cenário hipotético e ilustrativo, mostra-se que o volume de dados a serem confrontados com o uso de *clusters* é significantemente reduzido.

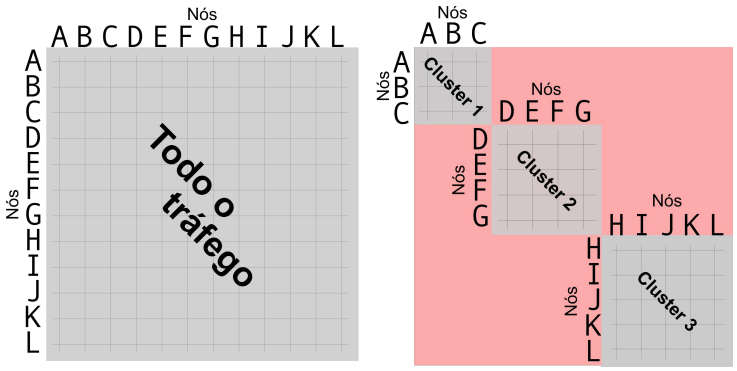


Figura 4.5: Comparação ilustrativa do volume de dados analisados no CGP com e sem o uso de *clusters*

Para cada *cluster*, as relações entre os nós deste *cluster* são calculadas, tendo como saída uma matriz $N \times N$, onde N é número de nós presentes no *cluster*. Calculadas as matrizes de influências, para cada *cluster*, os *bots*, cujas influências sobre os outros nós são significativas, terão valores de ordens de magnitude maior que os nós que não apresentam nenhuma correlação entre si. Por realizar a detecção em *clusters*, o BotFetcher pode paralelizar a análise, trazendo ganhos de desempenho para o método. As matrizes são avaliadas individualmente para cada *cluster*, identificando os *bots* em cada *cluster*. Destaca-se aqui que cabe ao BotFetcher apenas a classificação dos nós em *bots* ou não e não a atuação na rede para interromper o seu efeito malicioso. Dessa forma, os resultados podem ser informados parcialmente após a análise de cada *cluster*, para atuação imediata na rede ou após a análise de todos os *clusters*.

4.5 MÓDULO DE NOTIFICAÇÃO

O Módulo de Notificação é o módulo que realiza a comunicação dos Módulos com outros sistemas ou entre Módulos, mas de maneira assíncrona ou não-sequencial. Tanto o Módulo de Processamento quanto o Módulo de Análise se comunicam com o Módulo de Notificação. Este módulo é responsável por disparar a detecção de *bots* uma vez que um ataque DDoS é predito. O Módulo de Notificação também reporta resultados para administradores de rede ou para sistemas automatizados de modo que eles possam atuar na rede.

4.5.1 *Trigger*

O *trigger* é uma indicação feita pelo componente de predição, no Módulo de Análise, quando um possível ataque DDoS é identificado na rede. Este disparo indica para o Módulo de Processamento que este deve começar a analisar a rede para identificar os possíveis *bots* presentes. Como a detecção de um ataque DDoS se dá de maneira não determinada no tempo, este disparo é uma chamada assíncrona do Módulo de Análise para o Módulo de Processamento.

A partir da identificação da necessidade do início da análise da rede, o Módulo de Processamento utilizará as características do tráfego para realizar a seleção de características e a clusterização, de modo a propiciar ao Módulo de Análise a realização da detecção dos *bots* presentes na rede. Embora haja comunicação e coordenação entre os diversos módulos da arquitetura, as outras mensagens e informações trocadas se dão de maneira coordenada e

sequencial, de modo que não há a necessidade da utilização do Módulo de Notificação, dando sua comunicação por interfaces programadas.

4.5.2 Relatório

O relatório é uma sumarização dos resultados da análise. Este relatório identifica quais são os dispositivos que possuem comportamento de *bots* naquele tráfego analisado. A saída pode ter como objetivo identificar e informar ao administrador da rede quais dispositivos foram comprometidos, de modo que seu tráfego seja bloqueado ou seguir algum padrão específico àquela determinada rede. A saída pode ser padronizada para que um sistema atue diretamente na rede, impedindo que o tráfego malicioso cause dados à rede de origem e à rede atacada. Durante toda a análise, diversos registros são armazenados em arquivos, de modo que sejam auxiliares na análise ou sejam, de fato, entradas para o Módulo seguinte. Assim, a saída mais importante da arquitetura é a identificação de *bots* ou não de cada dispositivo. Entretanto, para uma análise mais aprofundada (como de um possível falso-positivo ou falso-negativo), estes registros também podem ser suplementados ao relatório de modo a munir o administrador da rede de insumos para uma confiável e segura operação da rede e seus dispositivos.

4.6 ESTUDO DE CASO

Esta seção detalha um estudo de caso para o BotFetcher - a arquitetura proposta para predizer ataques de DDoS e detectar seus *bots*. Quão antes o sistema sendo atacado sabe da iminência de um ataque DDoS e, sendo capaz de identificar os *bots*, melhor este sistema poderá agir defensivamente diante do ataque (Santos et al., 2017). Para antecipar *botnets* e mitigar os efeitos de ataques DDoS, a arquitetura proposta opera hierarquicamente em dois níveis: rede local e Internet. Na rede local, sinais identificam um ataque DDoS sem que este atinja estágios avançados do ataque. Estes sinais são baseados nos seguintes indicadores estatísticos: taxa de retorno, autocorrelação, coeficiente de variância e assimetria (apresentados na Seção 2.3.1.1). Assim como em (Pelloso et al., 2018) e diferentemente de outros trabalhos (Scheffer et al., 2009), os indicadores estatísticos são aplicados como um sistema de predição de tendências de ataques.

Assim, combinando estas duas técnicas, predição e detecção (como em sistemas de saúde), a arquitetura proposta é capaz de identificar atividades suspeitas, de maneira rápida e leve em recursos computacionais. A detecção é mais ampla e profunda, necessitando de mais recursos computacionais para executá-la. Utilizando predição e detecção, de maneira distribuída, é possível identificar *bots* de uma maneira escalável e confiável, sem a sobrecarga de continuamente realizar a custosa detecção de *bots*.

Para realizar a predição, os dados são coletados nas redes locais e pré-processados. Então, são agrupados em janelas e em séries temporais que são subsequentemente processados e analisados. Uma vez com os dados processados é utilizado um mecanismo de predição (com base nos indicadores estatísticos nas redes locais). A entrada para o componente de predição (Módulo de Análise) são os indicadores estatísticos (Módulo de Processamento) calculados sobre as séries temporais do tráfego. A saída do componente de predição indica uma mudança significativa no comportamento da rede. Assim, quando uma mudança no comportamento da rede é identificada, em qualquer rede local, um disparo é feito do componente de predição, através do Módulo de Notificação. A partir deste *trigger*, inicia-se o processamento para a detecção dos *bots*. Esta notificação pode partir de qualquer rede local e o módulo de processamento analisará os tráfegos das redes locais que tenham identificado esta mudança.

O componente de detecção de *botnets*, uma vez dado o disparo das redes locais, utiliza os dados coletados e processados, para identificar os *bots* que estejam causando um ataque

DDoS. Esta análise é mais ampla e profunda que a análise de mudança de estado da rede. Se utiliza de mais características e tem mais dispositivos como alvo da análise. A análise classifica o tráfego e identifica os dispositivos maliciosos. Assim, o componente de detecção de *bots* opera não apenas sobre o tráfego de uma única rede local, mas com a conjunção dos tráfegos de várias redes locais: a Internet. A mudança de comportamento em qualquer rede local pode disparar a detecção de *bots*. Como entrada, tem-se o tráfego coletado e processado, com suas características selecionadas e os dispositivos clusterizados. A detecção analisa a causalidade entre os nós de redes, em cada cluster identificado no Módulo de Processamento. Como saída, tem-se a classificação dos dispositivos em *bots* ou não.

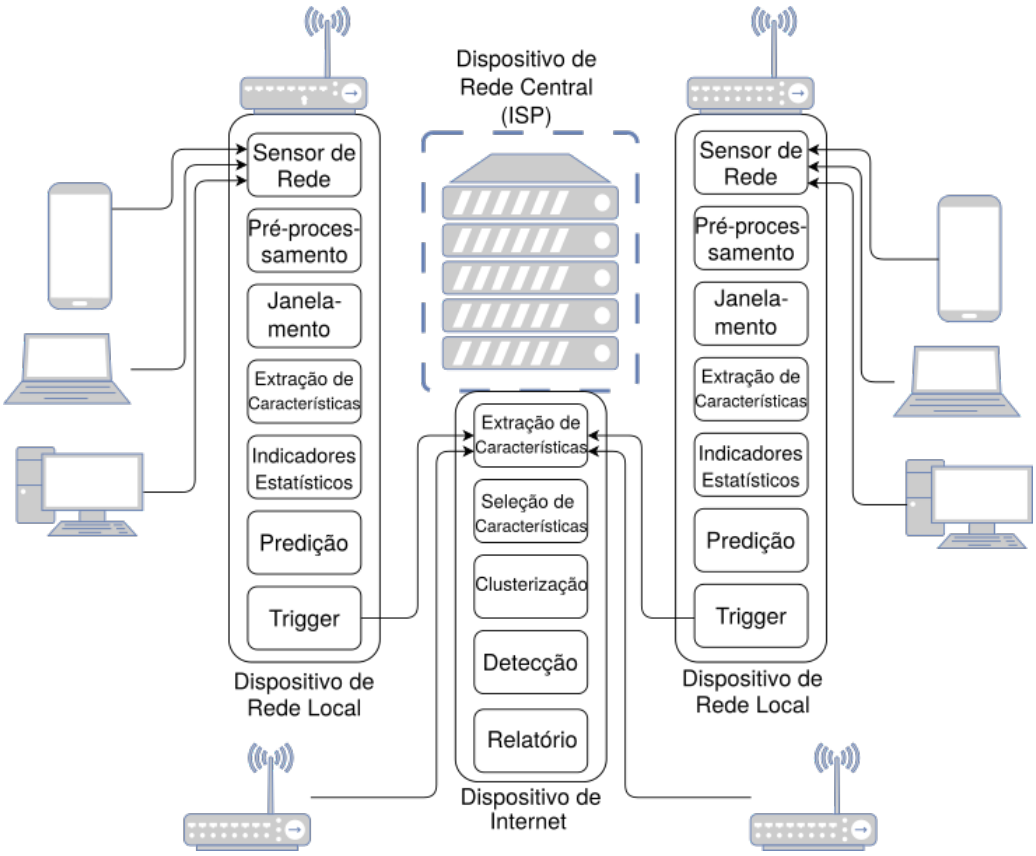


Figura 4.6: Visão geral da arquitetura em múltiplos dispositivos

A Figura 4.6 apresenta esta arquitetura em dois níveis, com seus componentes instanciados em diversos dispositivos da rede, tanto em redes locais quanto no nível da Internet. A figura mostra o tráfego senso coletado e analisado em dispositivos de redes locais, realizando a predição, enquanto a detecção dos *bots* é realizada em um dispositivo centralizado. O tráfego de dispositivos nas redes locais (tais como computadores, *smart* TVs, dispositivos móveis e dispositivos IoT) é encaminhado a um roteador, um *hub* ou um modem que, sendo parte da rede, captura o tráfego em tempo real. Este tráfego é coletado pelo dispositivo através de um sensor de rede. Os dados (binários) são pré-processados, permitindo a análise, em um formato legível para humanos. Todo o tráfego pré-processado é dividido em janelas que são analisadas e é o que caracteriza aquela janela de tempo. Do tráfego desta janela, ainda na rede local, os indicadores estatísticos são calculados e o componente de predição pode, enfim, indicar ou não uma transição crítica na rede. Se uma transição crítica é detectada, o dispositivo realizando a análise (na rede local) dispara o sistema de detecção de *botnets*. Este disparo acontece de maneira assíncrona,

dando início a coleta de dados centralizada. Esta predição é leve em recursos computacionais e fornece sinais precoces de um possível ataque DDoS.

Com uma transição crítica detectada, em qualquer estado das redes locais, o objetivo passa a ser investigar possíveis *bots* naquele tráfego. Com os dados coletados (Módulo de Coleta de Dados) e dada a arquitetura, pode-se identificar três etapas necessárias para a detecção: (*i*) extração e seleção de características, (*ii*) agrupamento (clusterização) e (*iii*) detecção de *bots* em cada *cluster*, conforme ilustra a Figura 4.7. Primeiro, é realizada a extração (Módulo de Extração de Características) e seleção de características (Módulo de Processamento), com o objetivo de trazer para análise características que possam ser relevantes na detecção de *bots* e identificar aquelas características mais relevantes a partir do tráfego de rede. Esta etapa tem como resultado um conjunto de características relevantes para a detecção de *bots*. Novas características (diferentes das usadas na análise da rede local) são extraídas. As características extraídas e selecionadas têm um papel fundamental no direcionamento da clusterização, sendo determinante para o funcionamento dos algoritmos de detecção. A etapa de clusterização (Módulo de Processamento) agrupa os nós da rede com comportamentos similares no tráfego. A técnica de aprendizagem de máquina não-supervisionada utilizada é a *Self-Organizing Map*. O comportamento é avaliado utilizando as características extraídas e selecionadas, ignorando as que não foram selecionadas na etapa anterior. Por fim, a etapa de detecção (Módulo de Análise) ocorre para cada *cluster* definido na etapa anterior, através da análise causal das relações entre os nós. A saída, identificando os dispositivos em *bots* ou não, é encaminhada ao Módulo de Notificação que reporta ao administrador de rede ou, possivelmente, a uma ferramenta automatizada que atuará na rede, evitando danos que o *bot* poderia causar.

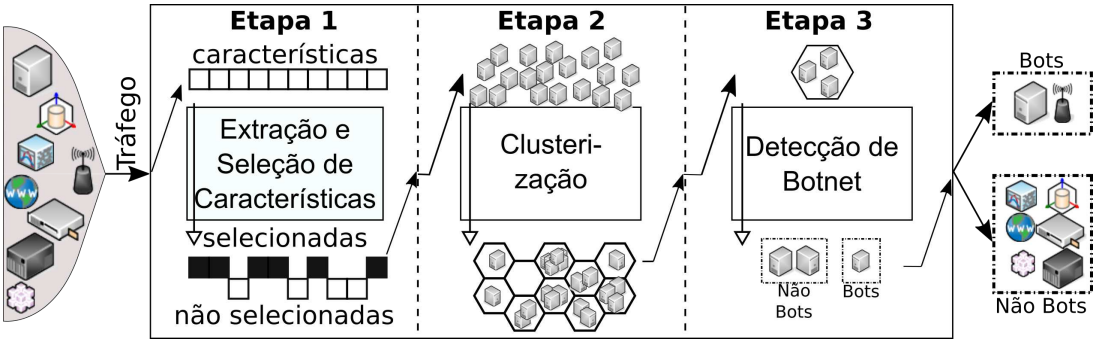


Figura 4.7: Etapas da detecção de *bots*

4.7 RESUMO

Neste capítulo são utilizados os fundamentos apresentados no Capítulo 2, instanciando-os para o uso no BotFetcher. A extração de características é mostrada, identificando as características de tráfego que são utilizadas na análise. Indica-se quais são as características oriundas dos pacotes do tráfego, seus cabeçalhos e rodapés, e como são sumarizadas e contabilizadas. Estas características são utilizadas na predição de ataques de negação de serviço distribuídos, através da identificação de indicadores estatísticos e como seus comportamentos ao longo do tempo, quando avaliados em conjunto, podem indicar um possível ataque. São mostradas também as características que são extraídas de um grafo formado utilizando os endereços IP de origem e destino dos pacotes. Assim, as características buscam evidenciar as relações entre os nó. Uma vez extraídas, as características são selecionadas utilizando-se o α-*investing*$^{+}$. Este algoritmo realiza uma análise estatística das entradas, selecionando as que possuem relação de independência com

as demais. A vantagem deste método é que ele opera no modo *filter*, *i.e.*, sem a necessidade de um método classificador operando em conjunto na análise dos dados.

Com as características selecionadas, uma técnica de aprendizagem de máquina não-supervisionada é aplicada: o *Self-Organizing Map* (SOM). O BotFetcher agrupa o conjunto de nós, se utilizando dos seus vetores de características, em *clusters*. O SOM propicia uma redução da dimensionalidade dos dados, migrando de uma visão multi-dimensional (com o número de dimensões do tamanho do vetor de características) para uma visão bidimensional, apresentada em um *grid*. No BotFetcher, um *grid* 5×5 é utilizado. Assim, todos os nós do tráfego que fazem parte da análise são inseridos em um dos 25 *clusters*, com cada *cluster* tendo nós que possuem comportamento similar entre si. O volume de dados a ser analisado é reduzido, pois nós em *clusters* diferentes não serão comparados. Além disso, uma paralelização da análise é possível. Dessa forma, cada *cluster* será investigado pelo BotFetcher para buscar detectar *bots*, com cada *cluster* sendo investigado separadamente.

Por fim, a detecção de *bots* é realizada em cada *cluster* através de um método determinístico: o *Causal Graph Process* (CGP). O CGP busca correlacionar o tráfego entre os nós e compreender as interrelações entre os dados coletados e suas intrarelações ao longo do tempo. Neste processo, utiliza-se autoregressão sobre séries temporais. Para cada nó e característica, tem-se uma série temporal, ou seja, uma coleção de valores distribuídos em intervalos de tempo regulares, do início do período de avaliação até seu fim. Os valores coletados ao longo do tempo são divididos em janelas de tempo regularmente espaçadas, de modo que o tamanho da amostra temporal é o mesmo para todos os nós avaliados e as as séries contenham a mesma quantidade de dados. Uma matriz de influências é a saída deste processo, possibilitando identificar as relações causais do tráfego gerado pelos nós e permitindo identificar o tráfego de *bots*. Cada entrada da matriz corresponde ao peso atribuído à relação entre os dois nós identificados na linha e na coluna da matriz. Quanto maior o valor na matriz, maior é a influência entre os nós identificados.

5 AVALIAÇÃO

Este capítulo apresenta a metodologia de avaliação do método proposto, as métricas utilizadas para a avaliação dos resultados e os resultados obtidos. Na Seção 5.1, são apresentadas as métricas utilizadas para avaliação dos resultados, a terminologia das métricas e os insumos utilizados para obtenção dos resultados. Ne Seção 5.2 são apresentados os resultados do BotFetcher. Para o cenário 10, na Subseção 5.2.1, da base de dados CTU-13 são apresentados os resultados e explicados minuciosamente, de acordo com os fundamentos e o estudo de caso apresentados nos Capítulos 2 e 4. Já na Subseção 5.2.2, são apresentados os resultados para os cenários 04 e 11, da mesma base de dados CTU-13, além da base de dados CAIDA. Por fim, a Seção 5.3 compara a proposta com outras literaturas, utilizando a base de dados Botnet 2014.

5.1 MÉTRICAS E METODOLOGIA DE AVALIAÇÃO

O objetivo principal do método é aumentar a detecção de *botnets*, com a redução da taxa de falsos-positivos e falsos-negativos, maximizando os verdadeiros-positivos e verdadeiros-negativos. Um verdadeiro-positivo é identificado quando a saída da análise corretamente prediz a classe positiva – no presente cenário, um *bot* foi corretamente classificado como um *bot*. Similarmente, um verdadeiro-negativo é quando a saída da análise corretamente prediz a classe negativa – um nó com tráfego benigno classificado corretamente. Já um falso-positivo é quando o modelo prediz incorretamente a classe positiva, *i.e.*, um nó é erroneamente classificado como *bot*. Por fim, um falso-negativo é quando a análise classifica erroneamente um nó benigno como *bot*. Na Tabela 5.1 pode-se observar estas classificações para o presente cenário.

		Real	
		Bot	Benigno
BotFetcher	*Bot*	Verdadeiro-Positivo	Falso-Positivo
	Benigno	Falso-Negativo	Verdadeiro-Negativo

Tabela 5.1: Matriz de confusão com a definição de verdadeiro-positivo, verdadeiro-negativo, falso-positivo e falso-negativo para *botnets*

Como as etapas do BotFetcher funcionam de maneira independente das outras, com limites e atribuições claramente definidos, pode-se, também, avaliar cada etapa do método separadamente. Apresentam-se os cenários e os resultados para todas as etapas, de modo a evidenciar o bom funcionamento das partes intermediárias. Dessa forma, serão apresentados os parâmetros utilizados, exemplos de entradas e as respectivas saídas em cada etapa. O tráfego utilizado na avaliação é oriundo da base de dados CTU-13, utilizando-se o tráfego anonimizado de rede da Universidade da República Tcheca. Os *bots* foram instanciados em máquinas virtualizadas, com 13 cenários diferentes de *botnets*. A máquina física utilizada por eles tinha o Debian como sistema operacional hospedeiro e as máquinas virtuais com sistema operacional Windows XP. Ao menos uma máquina virtualizada, em todos os cenários, estava infectada com um *malware* que gerava tráfego característico de uma *botnet*. As demais máquinas virtuais possuíam tráfego normal. Há, ainda, um tráfego de fundo na base de dados, de dispositivos que não eram de controle dos autores da base de dados.

Inicialmente, os dados de captura da base de dados CTU-13 são tratados pela etapa de extração e seleção de características. As características extraídas foram elencadas na Tabela 4.1. O método seleciona então as características mais relevantes para a análise, identificando quais foram

removidas e as que foram selecionadas para análise, por serem estatisticamente independentes, de acordo com o teste estatístico. Para a clusterização apresenta-se os cenários com a utilização da seleção de características e sem a sua utilização. Indicamos qual o *cluster* possui *bot(s)*. Além disso, mostra-se a redução alcançada por esta análise, partindo do menor *cluster* até o maior, com a detecção dos *bots*. Por fim, para detecção dos *bots*, avaliam-se as matrizes de influência para cada *cluster*, de modo a evidenciar a magnitude dos valores na matriz em um *cluster* com a presença do *bot* e sem a sua presença. Também, foi avaliada a base de dados CAIDA, proveniente do tráfego de um ataque DDoS real, pela Internet, não controlado pelos pesquisadores. Dada a magnitude deste ataque e o não controle dos nós realizando o ataque, esta base não é categorizada: não se sabe quais nós são ou não *bots*. Também, não há um período sem ataque. Todo o período analisado é do ataque, não havendo a necessidade do componente de predição. Dessa forma, faz-se uma análise desta base de dados, apresentando os resultados indicados pela detecção.

Os testes foram executados em uma máquina com processador Intel(R) Xeon(R) Silver 4114 CPU@2.20GHz, com 64 GB de RAM e sistema operacional Ubuntu Server 18.04.3 LTS. Para a extração dos dados e cálculo de características, foi utilizado Python, com a bibilioteca *NetworkX*. As implementações para a seleção de características foram feitas em Python e R. Para a clusterização, Python foi a linguagem utilizada com a bibilioteca MiniSOM. Por fim, para a detecção de *bots*, a implementação foi feita em objetive-C (arquivo *.m) que pode ser executado tanto em Matlab e Octave – aqui, foi utilizado Octave.

5.2 RESULTADOS

Nesta seção são apresentados os resultados da avaliação da arquitetura proposta. Da base de dados CTU-13, são utilizados os cenários 04, 10 e 11. Também, é utilizada a base de dados CAIDA, que apresenta um ataque real. A base de dados CTU-13 é categorizada: é sabido quais nós são *bots*. Já a base de dados CAIDA não é categorizada. Além disso, na base de dados CTU-13 há períodos em que há ataque e períodos de repouso da rede. Assim, a análise de predição se faz necessária. Já na base de dados CAIDA, tem-se apenas o tráfego do período em que o ataque está presente, sendo a análise de predição irrelevante. Na Subseção 5.2.1, é apresentado de maneira minuciosa os resultados para o Cenário 10 da base de dados CTU-13. Já na Subseção 5.2.2, os outros resultados são apresentados, sem que detalhes dos fundamentos sejam pormenorizados e detalhados, como foi feito para o Cenário 10.

5.2.1 Cenário 10 (CTU-13)

Nesta subseção são apresentados detalhadamente os experimentos do cenário 10 da base de dados CTU-13. Nas subseções seguintes, outros testes e resultados são apresentados sem discutir minuciosamente os detalhes aqui apresentados. O cenário 10, CTU-13, possui dez *bots* com o *malware* Rbot, causador de ataques DDoS. O *malware* Rbot explora vulnerabilidades em sistemas operacionais Windows, infectando os dispositivos e realizando ataques DDoS. O cenário total possui duração de 4,75 horas, com mais de 99 milhões de pacotes trocados na rede. O ataque acontece em múltiplos intervalos de tempo dentro deste período, permitindo que a análise seja efetuada: em um cenário com a rede sem ataque, em um cenário de repouso (sem a comunicação dos nós infectados); antes do início do ataque; e durante o ataque.

Como esta análise se deu através de um cenário pré-coletado, um arquivo binário (em formato **.pcap*) é a entrada para a arquitetura. O BotFetcher lê este arquivo e realiza a extração de características. Uma janela de 60 segundos é utilizada (de 0 a 59,99 segundos). A Figura 5.1 mostra o intervalo da captura para o cenário 10 da CTU-13 e realça o período da captura utilizado

para a análise da detecção dos *bots*. A arquitetura proposta implementada opera continuamente, calculando os indicadores estatísticos para cada janela de tempo. A detecção é realizada apenas quando uma indicação de um possível ataque DDoS é disparado. Embora o componente de predição utiliza o tráfego apenas de uma janela para a sua análise, uma vez indicnado um possível ataque DDoS, o componente de detecção acumula o tráfego das janelas, calculando a influência de um nó sobre outros, possivelmente indicando a presença de *bots*.

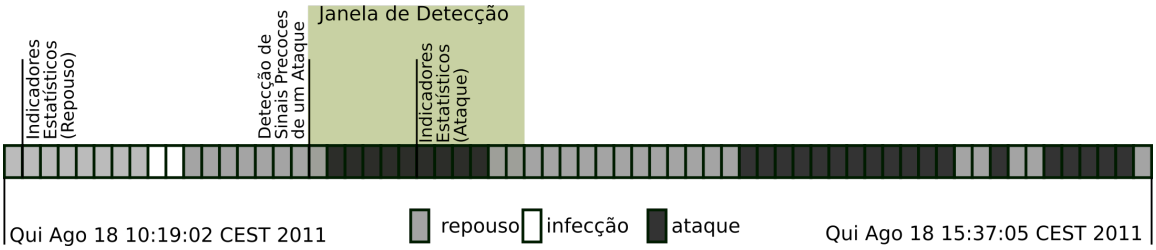


Figura 5.1: CTU-13 - Período de ataque do cenário 10

Para o cálculo dos indicadores estatísticos e posterior análise do comportamento da rede, séries temporais são construídas utilizando o tamanho do pacote trafegado na rede e sua estampa de tempo (dentro da janela de análise). O objetivo consiste em utilizar características simples mas relevantes para predizer a iminência de um ataque. Características tais como frequência de geração de pacotes e/ou o tamanho de pacotes fortemente caracterizam um ataque DDoS volumétrico. Alta frequência de geração de pacotes é geralmente associada a um ataque DDoS. O aumento no tamanho do pacote usualmente indica que o atacante está tentando causar a saturação da rede. Pacotes de rede são, normalmente, na média, menores que 250 *bytes*. No entanto, é possível observar pacotes atingindo 1500 *bytes*, sobretudo quando a rede está sob ataque.

É possível observar um rápido crescimento no tamanho do pacote na série temporal (Figura 5.2, à direita). Enquanto na primeira série temporal (Figura 5.2, à esquerda) a maioria do tamanho dos pacotes atinge cerca de 60 *bytes*, na segunda série temporal (Figura 5.2, à direita) é possível notar uma transição (indicada pela linha tracejada vertical vermelha) do período em que os pacotes eram majoritariamente de cerca 60 *bytes* para o período em que os pacotes são de 1500 *bytes* (o tamanho máximo de um pacote, definido pelo MTU (*Maximum Transmission Unit*), parâmetro da camada de Rede dos dispositivos). Outro aspecto interessante consiste nos dois picos (realçados nos retângulos vermelhos) apresentados na primeira série temporal (Figura 5.2, à esquerda). Apesar desta série temporal, neste instante, não estar sob ataque, alguns pacotes foram enviados com o tamanho de 1500 *bytes*. Este comportamento foi observado em outros períodos e pode indicar que os *bots* estivessem se preparando para iniciar o ataque ou, ainda, testando se estão operacionais, minutos antes do ataque ser de fato iniciado.

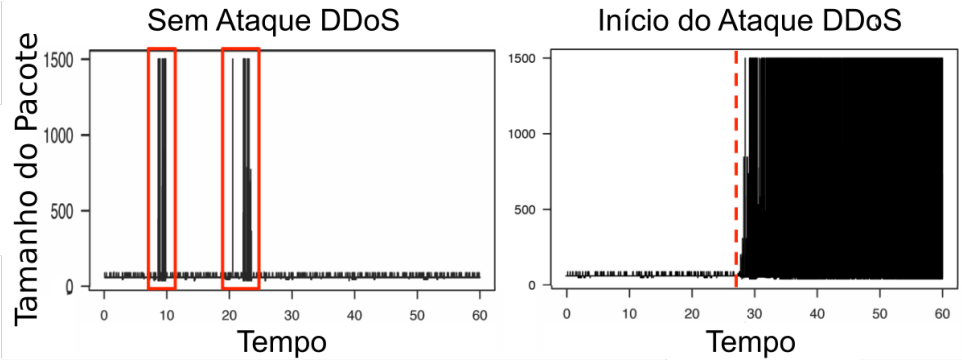


Figura 5.2: Ataque DDoS

O cálculo dos indicadores estatísticos é feito a cada janela de 60 segundos. Esta janela foi criada, para a análise, a partir do arquivo *pcap* com todo o tráfego da rede. Três comportamentos podem ser observados: a rede em repouso, a rede momentos antes de um ataque começar (onde já havia a comunicação de *bots*) e durante um ataque. Momentos antes do ataque, há uma mudança no estado da rede, calculado pelos indicadores estatísticos, provendo um sinal que alguma atividade maliciosa pode estar acontecendo na rede. Com isso, a detecção de *botnets* começa a se utilizar do tráfego coletado para identificar os *bots*.

A Figura 5.3 mostra os resultados dos indicadores estatísticos antes do início do ataque (de 18/08/2011 11:46:13 CEST a 18/08/2011 11:47:13 CEST). Os indicadores mostram o início iminente de um ataque DDoS, *i.e.*, há uma redução na curva da taxa de retorno enquanto há um aumento nas curvas de autocorrelação, coeficiente de variância e assimetria. A redução da taxa de retorno indica significantes oscilações no fluxo do tráfego de rede, assim, não sendo capaz de se manter em um estado metaestável. A tendência de alto aumento para o índice de autocorrelação (Kendall tau positivo no valor de 0.896) indica uma similaridade nos tamanhos dos pacotes ao longo do tempo, significando uma tendência de os pacotes permanecerem próximos a 1500 *bytes*. Este tamanho do pacote é usado pelo atacante para sobrecarregar a largura de banda da rede durante um ataque, já que este é o tamanho máximo de um pacote de rede. O aumento no coeficiente de variância indica uma forte instabilidade na rede, devido a presença de valores próximos aos seus limites e de alta variação nos tamanhos nos tamanhos dos pacotes. Finalmente, uma assimetria positiva, com sua tendência de aumento, aponta uma concentração de tamanho de pacotes próxima a 1500 *bytes*. Considerando os indicadores calculados, uma transição crítica na rede é identificada, apontando para um iminente ataque DDoS.

Para ilustrar o comportamento quando nenhuma transição crítica é encontrada a Figura 5.4, à esquerda, mostra o estado da rede em um estágio de repouso. Neste ponto, a captura do tráfego já havia sido iniciada (pois uma transição crítica havia sido verificada em janelas anteriores), mas sem a presença de *bots* realizando ataques DDoS. De maneira similar, a Figura 5.4, à direita, mostra os indicadores estatísticos durante um ataque DDoS, quando, também, não há nenhuma transição crítica (o ataque tende a continuar). A taxa de retorno indica a continuidade do estado metaestável devido a baixa variância dos dados. Ambas autocorrelação e coeficiente de variância indicam uma baixa similaridade dos dados, com pouca variância, indicando para uma mudança no tamanho da média do pacote.

A mudança de estado, mostrada na Figura 5.3 dispara o início da detecção de *bots*. A Figura 5.1 mostra a janela em que os dados foram acumulados e continuamente escaneados para a detecção de *bots*. A seguir, os resultados mostram a detecção de *botnets* realizando a extração e seleção de características, a clusterização e, finalmente, a indicação de quais dispositivos se comportam como *bots* ou não. Durante a análise, os tráfegos de 35477 nós foram observados naquela janela de tempo. Para todos os nós, dentro da janela analisada, 19 características são continuamente extraídas e analisadas. A Tabela 5.2 mostra o resultado da seleção de características, tendo como saída os conjuntos de características selecionadas e não selecionadas. Nota-se uma redução de 48% no número de características totais. Vale ressaltar que a característica *node betweenness centrality*, não selecionada, é a que tem maior custo computacional para ser extraída. Porém, não é sabido, antes da realização da análise de seleção de características, se essa característica deveria ou não ser selecionada. Em alguns casos, esta característica tomou 99,9% do tempo gasto para a extração, com a realização dos cálculos necessários. Em todos os casos, o tempo observado foi maior que 95% do tempo total de extração (também observado em (Daya et al., 2019)).

O componente de clusterização cria 25 *clusters* utilizando as características selecionadas. Todos os 10 *bots* desta base de dados foram agrupados no mesmo *cluster*, junto com um nó sem

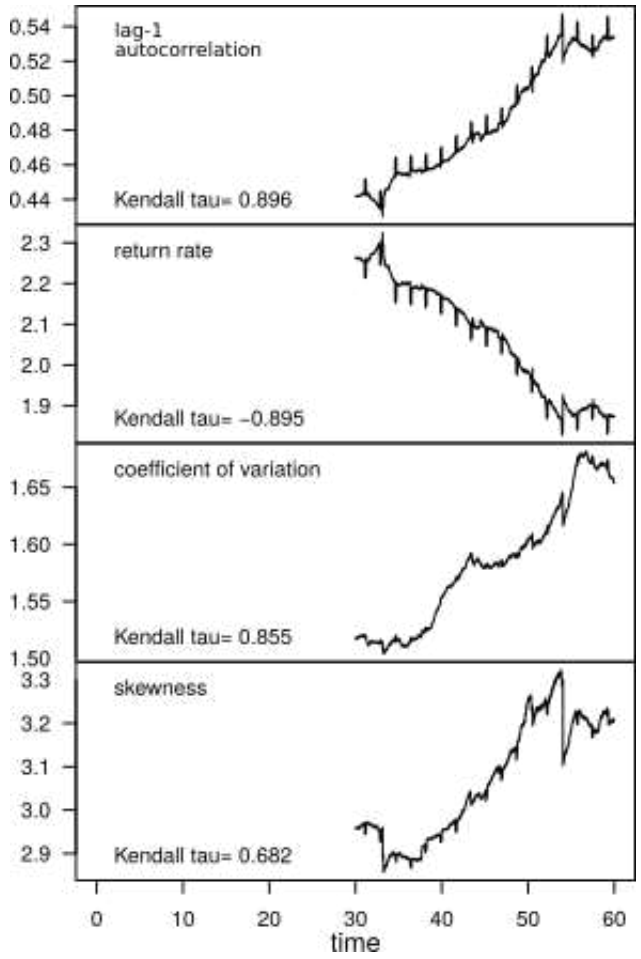


Figura 5.3: Indicadores estatísticos na iminência de um ataque

Não Selecionadas	**Selecionadas**
TCP Windows Size	Average TTL
Percent DNS	Destination Port Quantity
Percent TCP	Frame Length
Source Privileged Ports	Percent ICMP
Local clustering coefficient	Percent UDP
Out Degree Weight	Percent Outros
Out Degree	Source Not Privileged Ports
Node Betweenness Centrality	In Degree Weight
Protocol Quantity	In Degree
	Eigenvector Centrality

Tabela 5.2: Características selecionadas e as não selecionadas para a análise

comportamento malicioso. A Tabela 5.3 realça, em vermelho, o *cluster* com os *bots*. Iniciando-se a análise do menor *cluster*, a detecção seria capaz de identificar os *bots* após analisar apenas outros 27 nós (em 7 *clusters*). Isso acelera a detecção e reduz o tempo necessário para que uma ação seja realizada na rede, mitigando o ataque.

Por fim, a realização da detecção dos *bots*, no Módulo de Análise, se inicia pelo menor *cluster*. A matriz de influências, gerada para cada *cluster*, é avaliada, para identificar a influência de um nó sobre outros. A característica utilizada para a avaliação de influência é o número total de pacotes trocados entre os pares de nós. Quando nenhuma correlação é observada, os valores

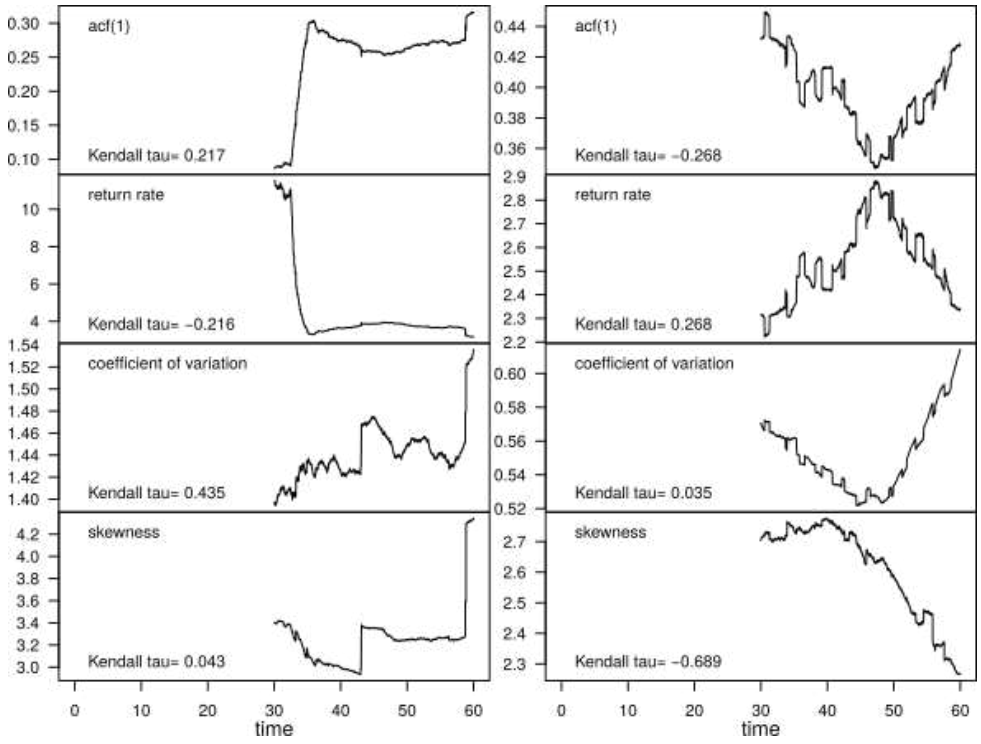


Figura 5.4: Indicadores estatísticos sem a iminência de um ataque

	0	1	2	3	4
0	21	23	9	0	3
1	100	27	3	**11**	3
2	2963	48	20	0	3
3	7570	132	23	13	3
4	24113	339	35	12	3

Tabela 5.3: CTU-13 Cenário 10: *Clusters* durante a análise do ataque

na matriz de influências é próximo a zero, em valores absolutos, para a linha e coluna na matriz correspondentes para este par de nós. A Figura 5.5 ilustra a matriz de influências e os valores altos para um nó, com grande influência sobre os pares.

Os resultados também mostram a influência entre nós benignos, em outros *clusters*. Quando esta influência é observada, um nó é identificado como *bot*. No *cluster* com os *bots* (10 *bots* entre 11 nós), a detecção classificou corretamente 9 dos 10 *bots*, corretamente identificando como benigno o nó daquele *cluster*. Assim, a Tabela 5.4 mostra a matriz de confusão para o cenário 10, da base de dados CTU-13, após a detecção ser concluída. 9 nós foram corretamente classificados como *bots* (verdadeiro-positivos), 35464 nós foram corretamente classificados como benignos (verdadeiro-negativos), 7 foram erroneamente classificados como *bots* (falso-positivos) e um *bot* foi erroneamente classificado como benigno (falso-negativo).

		Real	
		Bot	Benigno
Classificação	Bot	9	7
	Benigno	1	35460

Tabela 5.4: Matriz de Confusão após a detecção dos *bots*

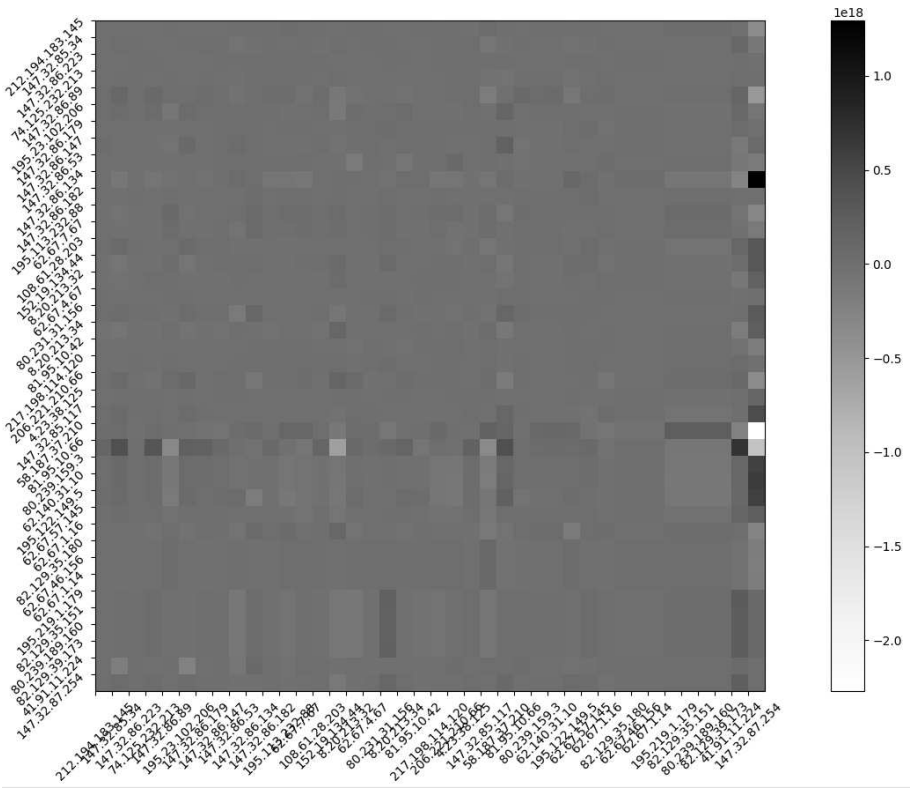


Figura 5.5: Matriz de Influência em um *cluster*. O nó na última coluna possui maior influência nos nós do que outros nós no *cluster*.

Também, vale ressaltar que a utilização dos *clusters* e da análise de predição permite uma detecção mais rápida. Se o componente de detecção tivesse que avaliar todos os nós da base de dados em busca de *bots*, o tempo de execução poderia tornar a análise impraticável. Esta abordagem baseada em *clusters* e realizando a detecção apenas quando disparado pela indicação de um ataque DDoS (pelos indicadores estatísticos) reduz consideravelmente o volume de dados, sem comprometer o resultado. Na Tabela 5.5 é ilustrado esse efeito, com todo o intervalo de coleta avaliado e agrupado em *clusters*. O tráfego de mais de 110000 nós precisariam ser avaliados, com muito mais dados e tráfego, possivelmente inviabilizando a análise.

	0	1	2	3	4
0	106956	779	105	7	9
1	1832	195	24	10	7
2	336	61	20	0	3
3	114	26	13	11	1
4	51	28	16	9	**17**

Tabela 5.5: CTU-13 Cenário 10: *Clusters* quando todo o tráfego é avaliado

5.2.2 Outros resultados

Esta subseção apresenta os resultados dos cenários 04 e 11 da base de dados CTU-13 e, também, da base de dados CAIDA. Estas bases de dados possuem ataques DDoS. São bases de dados consolidadas na análise e detecção de *botnets*, que, embora tenham sido criadas a mais de 10 anos, ainda possuem a sua relevância, tanto acadêmica quanto prática. A base de dados

CTU-13 é oriunda do tráfego da Universidade da República Tcheca. Já a base de dados CAIDA é proveniente de um ataque real. O cenário 04 da base de dados CTU-13 durou 4,21 horas, no dia 15/08/2011, das 10:59:23 CEST até 15:11:46 CEST. Este cenário gerou mais de 62 milhões de pacotes, sendo o *bot* malicioso o Rbot. O cenário 11 da base de dados CTU-13 durou apenas 15 minutos, gerando mais de 6 milhões de pacotes. A base de dados CAIDA possui pouco mais de uma hora de duração, com traços de tráfego anonimizado, de um ataque de negação de serviço distribuído que ocorreu em 04/08/2007, das 20:50:08 UTC até 21:56:16 UTC. Este ataque de negação de serviço buscava exaurir os recursos do servidor, tanto de processamento quanto de largura de banda da rede do servidor para a Internet, bloqueando acessos legítimos ao alvo.

Tanto na base de dados CAIDA quanto na base de dados CTU-13, cenários 04 e 11, foi possível verificar uma transição crítica no estado da rede antes do ataque efetivamente começar. A Figura 5.6 mostra os indicadores estatísticos para estas bases de dados. A transição crítica foi detectada antes dos ataques começarem, notada pela redução dos valores da taxa de retorno e aumento na autocorrelação, coeficiente de variância e assimetria.

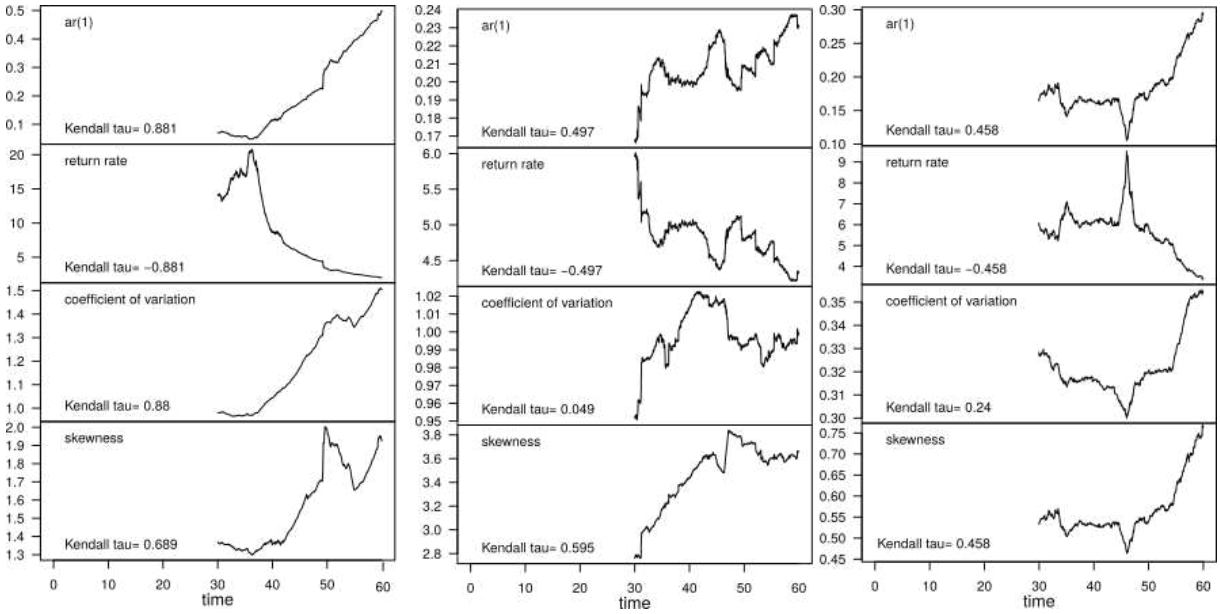


Figura 5.6: Indicadores estatísticos calculados para o cenário 04, CTU-13 (esquerda), cenário 11, CTU-13 (centro) e CAIDA (direita). Transições críticas identificadas.

A Tabela 5.6 mostra os resultados da seleção de características, realizada pelo Módulo de Processamento, para cada base de dados. O conjunto de características selecionadas não é estático, mas varia de acordo com cada tráfego. A característica *node betweenness centrality* não foi selecionada em nenhuma base de dados. Esta característica é a que possui maior custo computacional para seu cálculo e extração. Porém, não é possível saber antecipadamente se esta característica deve ou não ser selecionada. Na sequência, é feito o processamento da clusterização, que tem como saída o *grid* 5×5 com os nós associados a cada *cluster*.

No cenário 04 da base de dados CTU-13, há apenas um *bot*. O *bot* está no *cluster* junto com outros 71 nós, como pode ser visto na Tabela 5.7. No cenário 11 da base de dados CTU-13 (Tabela 5.8), há três *bots*: dois são observados em um *cluster* com outros cinco nós enquanto o outro *bot* é encontrado em um *cluster* com outros 456 nós. Ter os nós agrupados acelera a detecção pois o *bot* pode ser detectado antes de que *clusters* com alto número de nós (e, usualmente, sem *bots*) sejam avaliados, o que tomaria mais tempo computacional e processamento. O número de *bots* na base de dados CAIDA é desconhecido, haja visto ser uma base de dados proveniente de um ataque real. No entanto, ao invés de realizar a análise de todos

Características	CTU-13/S04	CTU-13/S11	CAIDA
Protocol Quantity			
Average TTL	✓	✓	✓
TCP Window Size			
Percent TCP			✓
Percent UDP	✓	✓	
Percent DNS	✓	✓	
Percent ICMP	✓	✓	✓
Percent Others			✓
Source Privileged Ports	✓	✓	
Source Not Privileged Ports			✓
Destination Port Quantity	✓	✓	✓
Frame Length	✓	✓	
In Degree			✓
Out Degree	✓	✓	
In Degree Weight			✓
Out Degree Weight	✓	✓	
Node betweenness centrality			
Local clustering coefficient		✓	
Eigenvector centrality	✓	✓	

Tabela 5.6: Características selecionadas para cada base de dados

	0	1	2	3	4
0	168268	3172	31	17	**72**
1	919	367	25	11	2
2	243	115	27	8	4
3	34	44	18	10	8
4	61	41	20	40	20

Tabela 5.7: Cenário 04 da base de dados CTU-13: *Clusters* - *bot* no *cluster* (0,4)

os 9251 em uma só análise, a clusterização permite uma rápida detecção possibilitando uma rápida ação na rede, reduzindo os danos causados pelas ações dos agentes maliciosos presentes. A Tabela 5.9 mostra o resultado da clusterização para a base de dados CAIDA.

O componente de detecção (Módulo de Análise) foi capaz de, com sucesso, encontrar *bots* em todas as bases de dados. No cenário 04 da base de dados CTU-13, o componente de detecção identificou uma significativa influência em todos os nós do *cluster* que continha o *bot*. Isso explica o alto número de falso-positivos na detecção neste cenário. Também, este cenário é o que teve mais nós analisados, dentre as bases avaliadas, totalizando 186 206 nós. A Tabela 5.10 sumariza o cenário 04 da base de dados CTU-13.

Para o cenário 11, da base dados CTU-13, o componente de detecção identificou 6 dos 7 nós como *bots* (no *cluster* com dois *bots* reais), sendo 4 incorretamente indicados. Também, a detecção não detectou o *bot* presente com os outros 456 nós (no *cluster* com apenas um *bot* real). Também, em *clusters* menores, foi verificada uma significante influência entre alguns nós, causando a incorreta classificação de 17 nós como *bots*. O número total de nós analisado no cenário 11 da base de dados CTU-13 foi de 41892.

	0	1	2	3	4
0	3	**7**	3	13	23
1	4	0	10	6	2
2	12	14	5	25	51
3	22	56	49	170	**457**
4	2735	3396	7183	4062	21547

Tabela 5.8: CTU-13 Cenpario 11: *Clusters* - 2 *bots* no *cluster* (0,1) e 1 *bot* no *cluster* (3,4)

	0	1	2	3	4
0	114	58	1	0	1
1	136	44	81	0	0
2	78	80	84	6	0
3	130	95	117	49	3762
4	1794	854	169	857	741

Tabela 5.9: CAIDA: *Clusters*

Na base de dados CAIDA, o tráfego de 9251 nós estava presente. Esta base de dados apresenta apenas o tráfego da rede em que a vítima fazia parte, enviando ou recebendo pacotes. O número de *bots* é desconhecido, pois esta base de dados não é rotulada . O mecanismo de detecção detectou alta coordenação entre 2738 nós, marcando-os como *bots*.

5.3 COMPARAÇÃO DE TÉCNICAS

Para comparar o BotFetcher com técnicas de aprendizagem de máquina, foi utilizada a base de dados Botnet 2014 (Biglar Beigi et al., 2014). Esta base de dados foi escolhida pois sua integralidade foi dividida em conjuntos de treinamento e teste, condição necessária para o treinamento das técnicas. Caso outra base fosse utilizada (como algum cenário da CTU-13 ou CAIDA), uma intervenção na base para separar estes conjuntos poderia favorecer o BotFetcher ou prejudicá-lo, de maneira arbitrária, interferindo nos resultados. Assim, preferiu-se uma base já dividida em conjuntos de treinamento e teste. Esta base de dados foi construída agrupando três outras bases de dados: base de dados ISOT (Zhao et al., 2013); base de dados ISCX 2012 IDS (Shiravi et al., 2012); e tráfego de *botnets* gerado pelo *malware* do Projeto *Capture Facility* (García et al., 2014)). Busca-se, com esta base, melhorar a generalização, realismo e representatividade dos dados, em uma única base.

A técnica de aprendizagem de máquina que foi comparada foi a Árvores de Decisão (*Decision Tree*). Esta foi escolhida pois resultados da literatura indicaram possuir melhores resultados que outras técnicas (Daya et al., 2019). Além disso, a técnica de Árvores de Decisão provê uma seleção de características natural, inerente à sua execução, realizando a seleção de características enquanto realiza a análise de performance (em um método conhecido como híbrido/*embedded* (Iglesias e Zseby, 2015)). Assim, como o BotFetcher também realiza a seleção de características, é justa a utilização da técnica Árvores de Decisão para a comparação. O método de seleção de características no BotFetcher é denominado *filter*, em contraste com o método *embedded* da técnica Árvores de Decisão, conforme detalhado no Capítulo 2.

Porém, deve-se ressaltar que a execução da técnica Árvores de Decisão não utilizou clusterização nem predição, dada a condição de utilizar os conjuntos de treinamento e teste da base de dados. O conjunto de treinamento foi utilizado para treinar o algoritmo de aprendizagem de máquina e, assim, os resultados foram avaliados utilizando o conjunto de teste. Como o

		Real	
		Bot	Benigno
Classificação	Bot	1	104
	Benigno	0	186101

Tabela 5.10: Matriz de Confusão após a detecção de *bots* no cenário 04 da CTU-13

		Real	
		Bot	Benigno
Classificação	Bot	2	17
	Benigno	1	41872

Tabela 5.11: Matriz de Confusão após a detecção de *bots* no cenário 11 da CTU-13

BotFetcher não requer uma fase de treinamento, os resultados foram obtidos aplicando a detecção determinística diretamente no conjunto de teste. Os resultados da técnica Árvores de Decisão podem ser encontrados na Tabela 5.12.

		Real	
		Bot	Benigno
Classificação	Bot	7	3
	Benigno	24	14173

Tabela 5.12: Matriz de Confusão para a técnica Árvores de Decisão para a base de dados Botnet 2014

O componente de predição é altamente dependente do tempo. Até em janelas em que não são dividas pelo tempo, mas por contagem de pacotes (janelas a cada 10 000 pacotes, por exemplo), os indicadores estatísticos são calculados utilizando as estampas de tempo dos pacotes. Isto se dá pelo fato que uma transição crítica é avaliada em um dado tempo, em um intervalo. Pela maneira que a base de dados Botnet 2014 foi construída, os pacotes são esparsamente distribuídos no tempo. Dividindo a base de dados a cada 60 segundos (ou até 3000 segundos - janelas de 50 minutos), apenas um (ou poucos) pacotes eram inseridos naquela janela. Se a contagem de pacote fosse utilizada, como os pacotes são esparsamente distribuídos, os indicadores estatísticos também não poderiam ser calculados para toda janela (dado o uso de estampas de tempo nos cálculos). Utilizando contagem de pacotes, algumas janelas tiveram seus indicadores calculados e foram capazes de detectar uma transição crítica. No entanto, dada a inconsistência, optou-se por não aplicar o componente de predição nos resultados a seguir, dado que não foi possível adequar a base e suas estampas de tempo de maneira confiável para a análise.

Assim, o BotFetcher prossegue realizando a seleção de características. Para a base de dados Botnet 2014 as seguintes características foram selecionadas: TTL médio; quantidade de portas de destino; Tamanho do *Frame*; Porcentagem ICMP; Porcentagem DNS; Porcentagem TCP; Porcentagem Outros; Portas de Origem Não Privilegiadas; *In Degree*; e *In Degree Weight*. A seleção de características reduziu o volume de dados a ser analisado em 48%. Estas características são as mais estatisticamente relevantes para esta base de dados. Dessa forma, o componente de clusterização agrupa os nós. A Tabela 5.13 apresenta os *clusters* 5×5, resultado da aplicação da técnica *Self-Organizing Map*, para a base de dados Botnet 2014.

De maneira similar a outras base de dados, o componente de detecção classifica os nós em *bots* ou não. A classificação final, após analisado cada *cluster*, pode ser vista na Tabela 5.14. Comparando o resultado da técnica de Árvores de Decisão, houve uma melhora na detecção

	0	1	2	3	4
0	19	34	27	2	14
1	99	72	60	28	18
2	369	86	66	41	142
3	299	36	32	1536	8900
4	684	114	754	364	411

Tabela 5.13: Botnet 2014: *Clusters*

de *bots*, identificado no valor de verdadeiros-positivos. Em contrapartida, houve um aumento também de nós benignos classificados como *bots*.

		Real	
		Bot	Benigno
Classificação	Bot	12	43
	Benigno	19	14133

Tabela 5.14: Matriz de Confusão para o BotFetcher na base de dados Botnet 2014

5.4 RESUMO

Neste capítulo, foram apresentados os resultados obtidos pela aplicação do BotFetcher nas bases de dados CTU-13 (cenários 05, 10 e 11) e CAIDA. Também, utilizou-se a arquitetura proposta para a comparação de técnicas, utilizando a base de dados Botnet 2014. A implementação do BotFetcher se deu principalmente na linguagem Python, com o uso das bibliotecas *NetworkX* e *MiniSOM*. R e objetive-C também foram utilizados. Utilizou-se a contagem de verdadeiro-positivos, verdadeiro-negativos, falso-positivos e falso-negativos como métrica para a avaliação. Estes resultados podem ser apresentados em uma matriz de confusão, facilitando o entendimento.

Foram discutidos os resultados de cada cenário separadamente. Na base de dados CTU-13, que é rotulada, foi possível identificar a mudança de comportamento na rede antes do início do ataque, caracterizado por uma transição crítica, indicando um possível ataque. Iniciada a análise para a identificação dos *bots*, nos três cenários avaliados foi possível identificar o(s) *bot(s)* presentes, com um baixo número de falso-positivos e falso-negativos, dado o volume de nós avaliados. Já na base de dados CAIDA, que não é rotulada, após a identificação da transição crítica da rede, 2738 nós foram identificados como *bots* de mais de 9000 nós avaliados.

Para comparar a arquitetura proposta com outras técnicas da literatura, utilizou-se a base de dados Botnet 2014. Esta base de dados foi escolhida pois sua integralidade foi dividida em conjuntos de treinamento e teste. A técnica de aprendizagem de máquina que foi comparada foi a Árvores de Decisão (*Decision Tree*). Esta foi escolhida pois resultados da literatura indicaram possuir melhores resultados que outras. Comparando o resultado da técnica de Árvores de Decisão, houve uma melhora na detecção de *bots*, identificado no valor de verdadeiros-positivos. Em contrapartida, houve um aumento também de nós benignos classificados como *bots*. Assim, ambas técnicas se mostraram competentes para a detecção, sendo uma apresentando um resultado maior de falso-positivos enquanto outra apresentando um maior número de falso-negativos. Esse *trade-off* deve ser avaliado ao optar-se por um determinado método.

6 CONCLUSÃO

Este trabalho apresentou BotFetcher, uma arquitetura para predição de ataques de negação de serviço distribuído e detecção de *botnets*, capaz de operar com um grande volume de dados de análise. Diferentemente de outros trabalhos e pesquisas, BotFetcher realiza a identificação precoce de um ataque, iniciando a análise para detecção dos *bots* a partir de uma identificação de ataque. Isto reduz o volume de dados analisados, sem a perda de qualidade no resultado. Além disso, faz o uso de características híbridas, provenientes tanto de análise de grafos, buscando inferir relações, quanto do tráfego propriamente dito, oriundas das informações dos pacotes. Embora o grafo seja construído de uma representação do tráfego, as características extraídas do grafo agregam informações relevantes para a análise. A utilização de características híbridas aumentou as características disponíveis para a realização da detecção dos *bots*.

Com a seleção de características, tem-se uma redução no volume de dados a serem analisados, sem perder qualidade na análise. Esta seleção identifica, dentre as características extraídas, aquelas que não possuem relação de dependência com as outras. Essa independência das características é esperada para técnicas de aprendizado de máquina, como a clusterização que é realizada na arquitetura. A clusterização propicia uma análise mais eficaz pois define um conjunto de *clusters* com comportamento similar. Por terem diferentes tamanhos, a análise parte dos menores *clusters* até chegar ao maior, de modo a permitir identificar um *bot* mais rapidamente e mitigar seus efeitos prontamente. Além disso, a dimensionalidade da análise fica reduzida pois avalia-se os dados por um número reduzido de *clusters* ao invés de analisar toda a base de dados com um número grande de nós. Por fim, a análise de cada *cluster* pode ser paralelizada em diversos dispositivos, acelerando os resultados. Testes foram realizados utilizando a base de dados CTU-13 e CAIDA. Pelos resultados, mostrou-se que a arquitetura é capaz de identificar os *bots* presentes nestas bases de dados. Também, comparou-se a presente arquitetura com uma técnica de aprendizado de máquina supervisionada, mostrando que os resultados são condizentes com a literatura, havendo, no BotFetcher, uma redução no número de falso-positivos.

A arquitetura, as técnicas e resultados foram objeto de submissão a conferências e periódicos. No Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) foi aceito o artigo "BotFetcher: Um Método Híbrido de Detecção de Botnets" (Rahal et al., 2020), (co-autores: Michele Nogueira e Alissar Moussa), no qual foram apresentadas as técnicas de detecção, com a extração e seleção de características, clusterização e detecção de *bots*. A ideia foi expandida para o periódico IEEE Access, que aceitou o artigo "*A Distributed Architecture for DDoS Prediction and Bot Detection*" (Rahal et al., 2020), (co-autores: Michele Nogueira e Aldri Santos), onde o componente de predição foi incorporado, de maneira distribuída na rede apresentando um maior número de testes para validar a proposta.

6.1 TRABALHOS FUTUROS E PONTOS EM ABERTO

A avaliação dos resultados utilizou-se de base de dados conhecidas na literatura. Os resultados foram satisfatórios e indicam que a proposta apresentada é promissora. Entretanto, vale ressaltar, que a análise da arquitetura é feita de maneira *offline*, sem dados serem trafegados de verdade. Como a arquitetura possui os componentes de predição distribuídos na rede, uma coordenação destes nós é necessária, identificando quais nós capturarão o tráfego e indicarão ao nó central quando iniciar e interromper a análise de detecção de *bots*. Além disso, pode haver a necessidade de encaminhamento do tráfego entre nós, de modo que as janelas avaliadas sejam

todas no mesmo intervalo. Assim, embora os resultados apresentados sejam pertinentes, quando realizados de forma *offline*, abstraem diversas comunicações que se fariam necessárias em um sistema atuando na rede de maneira ativa.

De maneira similar, o uso da clusterização propicia a possibilidade de paralelização da análise. Entretanto, embora tenha sido levantada essa possibilidade, a computação paralela também não foi abordada neste trabalho. Os resultados, porém, continuam válidos, haja visto que a paralelização não interfere no resultado da análise mas sim em como a análise pode ser feita. Assim, o entendimento é que a coordenação e comunicação distribuída e, também, o tópico de computação paralela seriam objetos de um mestrado por si só ou, então, o aprofundamento em uma tese, juntamente com outras propostas. O componente de predição identifica uma transição crítica no estado da rede e esta transição crítica é considerada como um prenúncio de um ataque de negação de serviço distribuído. Esta transição foi identificada em todos os cenários avaliados, em diferentes intervalos de tempo antes do ataque ser iniciado. Como ponto em aberto, porém, há de se avaliar até quanto tempo depois de uma transição crítica ser detectada o resultado ainda deve ser considerado e os dados coletados para detecção serem analisados.

Também, conforme dito acima, a análise se deu em bases de dados, de maneira *offline*. A aplicação da proposta em um ambiente real é um trabalho de longo prazo que colheria bons frutos. Os custos computacionais das etapas de predição, seleção de características e clusterização podem ser otimizados, porém, são marginais quando comparados com a extração de características e a detecção utilizando *Causal Graph Process*. Otimizações na utilização do *Causal Graph Process* são alvo de pesquisa do grupo CCSC (*Center for Computational Security sCience* da Universidade Federal do Paraná) e melhorias já foram alcançadas no quesito tempo de execução, quando comparadas com a técnica apresentada neste trabalho. Vale ressaltar que o resultado, porém, não é afetado. Dessa forma, sanadas as questões de implementação acima discutidas, seria possível aplicar de maneira *online* a detecção, propiciando resultados capazes de detectar as *botnets* causadoras de negação de serviço distribuído.

REFERÊNCIAS

AbuKhousa, E., Mohamed, N. e Al-Jaroodi, J. (2012). e-Health Cloud: Opportunities and Challenges. *Future Internet*, 4(3):621–645.

al Qerem, A. (2020). Network-Based Detection of Mirai Botnet Using Machine Learning and Feature Selection Methods. Em *Handbook of Research on Multimedia Cyber Security, edited by Brij B. Gupta and Deepak Gupta*, capítulo: 16, páginas 308–318. IGI Global.

Atkin, D. J., Jeffres, L. W. e Neuendorf, K. A. (1998). Understanding Internet adoption as telecommunications behavior. *Journal of Broadcasting & Electronic Media*, 42(4):475–490.

Bellman, R. (1966). Dynamic Programming. *Science*, 153(3731):34–37.

Biglar Beigi, E., Hadian Jazi, H., Stakhanova, N. e Ghorbani, A. A. (2014). Towards effective feature selection in machine learning-based botnet detection approaches. Em *Conference on Communications and Network Security*, páginas 247–255. IEEE.

Binkley, J. R. e Singh, S. (2006). An Algorithm for Anomaly-based Botnet Detection. Em *Conference on Steps to Reducing Unwanted Traffic on the Internet - Volume 2*, Berkeley, CA, USA. USENIX Association.

Bissell, K., LaSalle, R. M. e Cin, P. D. (2019). The Cost Of Cybercrime. Accenture Ninth Annual Cost of Cybercrime Study – Research Report.

Brugger, T. (2007). KDD Cup ‘99 dataset (Network Intrusion) considered harmful). Acessado em 18/08/2020.

Burghouwt, P. (2015). Detection of Botnet Command and Control Traffic in Enterprise Networks. Ph.d. thesis, The Hague University of Applied Science, Holanda.

Business, I. (2010). Pushdo botnet pummels more than 300 Web sites. Disponível em: `https://www.itbusiness.ca/news/pushdo-botnet-pummels-more-than-300-web-sites/12341`. Acessado em 18/04/2020.

Bykova, M. e Ostermann, S. (2002). Statistical Analysis of Malformed Packets and Their Origins in the Modern Internet. Em *Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurment*, IMW ’02, páginas 83–88, New York, NY, USA. ACM.

Cai, Y., Morales, J. e Sun, G. (2020). Cyber Attribution from Topological Patterns. Em *International Conference on Computational Science - ICCS*, páginas 58–71.

Chowdhury, S., Khanzadeh, M., Akula, R., Zhang, F., Zhang, S., Medal, H., Marufuzzaman, M. e Bian, L. (2017). Botnet detection using graph-based feature clustering. *Journal of Big Data*, 4(1):1–14.

Cisco (2018). Visual Networking Index Complete Forecast Highlights. Disponível em: `https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_Device_Growth_Traffic_Profiles.pdf`. Acessado em 22/12/2019.

Cisco (2020). Cisco Annual Internet Report (2018–2023). Disponível em: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf. Acessado em 13/04/2020.

Collins, M., Shimeall, T., Faber, S., Janies, J., Weaver, R., Shon, M. e Kadane, J. (2007). Using uncleanliness to predict future botnet addresses. Em *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, páginas 93–104.

Cordeiro, A. M., Oliveira, G. M. d., Rentería, J. M. e Guimarães, C. A. (2007). Systematic review: a narrative review. *Revista do colégio Brasileiro de Cirurgiões*, 34(6):428–431.

Dakos, V., Carpenter, S. R., Brock, W. A., Ellison, A. M., Guttal, V., Ives, A. R., Kefi, S., Livina, V., Seekell, D. A., van Nes, E. H. et al. (2012). Methods for detecting early warnings of critical transitions in time series illustrated using simulated ecological data. *PloS one*, 7(7):e41010.

Dakos, V. e Lahti, L. (2013). R Early Warning Signals Toolbox. *The R Project for Statistical Computing*. http://www.early-warning-signals.org/.

Daya, A. A., Salahuddin, M. A., Limam, N. e Boutaba, R. (2019). A Graph-Based Machine Learning Approach for Bot Detection. *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, páginas 144–152.

Dong, C., Chen, Z., Cheng, Y. e Chen, B. (2020). BotDetector: An extreme learning machine-based Internet of Things botnet detection model. *Transactions on Emerging Telecommunications Technologies*.

El-Shaarawi, A. e Niculescu, S. (1992). On Kendall's Tau as a test for trend in time series data. *Environmetrics*, 3:385 – 411.

Ferreira, A. E. G. e Nogueira, M. (2018). Identificando Botnets Geradoras de Ataques DDoS Volumétricos por Processamento de Sinais em Grafos. Em *Anais do Workshop de Gerência e Operação de Redes e Serviços*, Porto Alegre, RS, Brasil. SBC.

García, S., Grill, M., Stiborek, J. e Zunino, A. (2014). An empirical comparison of botnet detection methods. *Computers & Security*, 45:100–123.

Group, T. S. (2009). Tracking GhostNet: Investigating a Cyber Espionage Network. Disponível em: https://www.scribd.com/doc/13731776/Tracking-GhostNet-Investigating-a-Cyber-Espionage-Network. Acessado em 18/04/2020.

Guillén, E., Rodriguez Parra, J. e Páez, R. V. (2014). *Improving Network Intrusion Detection with Extended KDD Features*, volume 247, páginas 431–445. Springer.

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

Hossain, I., Eshrak, S., Auvik, M. J., Nasim, S. F., Rab, R. e Rahman, A. (2020). Efficient Feature Selection for Detecting Botnets based on Network Traffic and Behavior Analysis. Em *International Conference on Networking, Systems and Security*.

Huifang Feng e Yantai Shu (2005). Study on network traffic prediction techniques. Em *International Conference on Wireless Communications, Networking and Mobile Computing, 2005.*, volume 2, páginas 1041–1044. IEEE.

Husák, M. (2020). Graph-based models in prediction and projection of cyber attacks. Em *International Workshop on Graph-based network Security*.

Iglesias, F. e Zseby, T. (2015). Analysis of network traffic features for anomaly detection. Em *Machine Learning 101*, páginas 59–84.

Ismail, Z., Jantan, A., Yusoff, M. e Kiru, M. (2020). The effects of feature selection on the classification of encrypted botnet. *Journal of Computer Virology and Hacking Techniques*.

Joshi, C., Bharti, V. e Ranjan, R. K. (2020). Analysis of Feature Selection Methods for P2P Botnet Detection. Em *Advances in Computing and Data Sciences*, páginas 272–282. Springer Singapore.

Journals, B. (2003). EarthLink wins $25 million lawsuit against junk e-mailer. Disponível em: `https://www.bizjournals.com/atlanta/stories/2002/07/22/story4.html`. Acessado em 18/04/2020.

Karasaridis, A., Rexroad, B. e Hoeflin, D. (2007). Wide-scale Botnet Detection and Characterization. Em *Conference on First Workshop on Hot Topics in Understanding Botnets*, páginas 1–7, Berkeley, CA, USA. USENIX Association.

Karim, A., Salleh, R., Shiraz, M., Shah, S., AWAN, I. e Anuar, N. (2014). Botnet detection techniques: review, future trends and issues. *Journal of Zhejiang University SCIENCE C (Computers & Electronics)*, 15.

Kate, R., Perez, R., Mazumdar, D., Pasupathy, K. e Nilakantan, V. (2016). Prediction and detection models for acute kidney injury in hospitalized older adults. Em *Medical Informatics and Decision Making*, volume 16. BMC.

Kayacık, G. e Zincir-Heywood, A. (2006). Using self-organizing maps to build an attack map for forensic analysis. Em *International Conference on Privacy*, páginas 33.

Kohonen, T. (2013). Essentials of the self-organizing map. *Neural Networks*, 37:52–65.

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

Kurose, J. e Ross, K. (2017). *Computer Networking: A Top-down Approach*. Pearson.

Lagraa, S., Francois, J., Lahmadi, A., Miner, M., Hammerschmidt, C. e State, R. (2017). BotGM: Unsupervised graph mining to detect botnets in traffic flows. Em *Cyber Security in Networking Conference (CSNet)*, páginas 1–8. IEEE.

Lan, K.-C., Hussain, A. e Dutta, D. (2009). The Effect of Malicious Traffic on the Network. *Proc. Passive and Active Measurement Wksp. (PAM)*.

Mei, J. e Moura, J. M. F. (2017). Signal Processing on Graphs: Causal Modeling of Unstructured Data. *IEEE Transactions on Signal Processing*, 65(8):2077–2092.

Mirkovic, J., Prier, G. e Reiher, P. (2002). Attacking DDoS at the source. Em *IEEE International Conference on Network Protocols, 2002.*, páginas 312–321.

Moussa, A. A., Nogueira, M. e Guedes, A. L. (2019). Seleção Online de Features em Streaming Baseada em Alpha-investing para Ataques DDoS. Em *Workshop de Gerência e Operação de Redes e Serviços- WGRS*, páginas 1–14, Porto Alegre. SBC.

Nadler, A., Aminov, A. e Shabtai, A. (2019). Detection of malicious and low throughput data exfiltration over the DNS protocol. *Computers & Security*, 80:36–53.

Neira, A., Medeiro, A. e Nogueira, M. (2020). Identificação Antecipada de Botnets por Aprendizagem de Máquina. Em *Anais do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, páginas 896–909, Porto Alegre, RS, Brasil. SBC.

Ogden, N., Abdelmalik, P. e Pulliam, J. (2017). Emerging infectious diseases: prediction and detection. Em *Canada Communicable Disease Report*, volume 43, páginas 206–211.

Pelloso, M., Vergutz, A., Santos, A. e Nogueira, M. (2018). A Self-Adaptable System for DDoS Attack Prediction Based on the Metastability Theory. Em *IEEE Global Communications Conference (GLOBECOM)*, páginas 1–6.

Post, T. (2010). Remember Aurora–and Other Botnets. Disponível em: `https://threatpost.com/remember-aurora-and-other-botnets-040110/73769/`. Acessado em 18/04/2020.

Rahal, B., Moussa, A. e Lima, M. (2020). BotFetcher: Um Método Híbrido de Detecção de Botnets. Em *Anais do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, páginas 309–322, Porto Alegre, RS, Brasil. SBC.

Rahal, B. M., Santos, A. e Nogueira, M. (2020). A Distributed Architecture for DDoS Prediction and Bot Detection. *IEEE Access*, 8:159756–159772.

Rajahalme, J., Conta, A., Carpenter, B. e Deering, S. (2004). IPv6 Flow Label Specification. Em *RFC 3697*.

Rapid7 (2017). WannaCry Update: Vulnerable SMB Shares Are Widely Deployed And People Are Scanning For Them (Port 445 Exploit). Acessado em 24/12/2019.

Reinsel, D., Gantz, J. e Rydnin, J. (2018). Seagate: The Digitization of the World - From Edge to Core. Disponível em: `https://www.seagate.com/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf`. Acessado em 18/04/2020.

Salusky, W. e Danford, R. (2007). Know Your Enemy: Fast-Flux Service Networks.

Samonas, S. e Coss, D. (2014). The CIA strikes back: Redefining Confidentiality, Integrity and Availability in Security. Em *Journal of Information System Security*, volume 10, páginas 21–45. Information Institute Publishing, Washington DC, USA.

Sandryhaila, A. e Moura, J. M. F. (2013). Discrete signal processing on graphs: Graph fourier transform. Em *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, páginas 6167–6170. IEEE.

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.

Saporito, G. (2019). A Deeper Dive into the NSL-KDD Data Set). Acessado em 18/08/2020.

Scheffer, M., Bascompte, J., Brock, W., Brovkin, V., Carpenter, S., Dakos, V., Held, H., Nes, E., Rietkerk, M. e Sugihara, G. (2009). Early-Warning Signals for Critical Transitions. *Nature*, 461(7260):53–59.

Scheffer, M., Carpenter, S., Foley, J., Folke, C. e Walker, B. (2001). Catastrophic shifts in ecosystems. *nat*, 413:591–.

SecureWorks (2010). ZeuS Banking Trojan Report. Disponível em: `https://www.secureworks.com/research/zeus`. Acessado em 18/04/2020.

Security, H. (2006). "Pay Per Click" fraud botnet discovered. Disponível em: `https://www.helpnetsecurity.com/2006/05/19/pay-per-click-fraud-botnet-discovered/`. Acessado em 18/04/2020.

Security, M. (2010). The Top 10 'Most Wanted' Spam-Spewing Botnets. Disponível em: `https://www.cio.com/article/2416837/the-top-10--most-wanted--spam-spewing-botnets.html`. Acessado em 18/04/2020.

Shiravi, A., Shiravi, H., Tavallaee, M. e Ghorbani, A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Em *Computers & Security*, volume 31, páginas 357–374. ACM.

Tanenbaum, A. e Wetherall, D. (2011). *Computer Networks*. Pearson Prentice Hall.

Tavallaee, M., Bagheri, E., Lu, W. e Ghorbani, A. (2009). A detailed analysis of the KDD CUP 99 data set. *IEEE Symposium. Computational Intelligence for Security and Defense Applications, CISDA*, 2.

Tian, J., Azarian, M. H. e Pecht, M. (2014). Anomaly Detection Using Self-Organizing Maps-Based K-Nearest Neighbor Algorithm. Em *Proceedings of the European Conference of the Prognostics and Health Management Society*, páginas 1–9. Citeseer.

Times, N. Y. (2019). Big Companies Thought Insurance Covered a Cyberattack. They May Be Wrong. Disponível em: `https://www.nytimes.com/2019/04/15/technology/cyberinsurance-notpetya-attack.html`. Acessado em 18/04/2020.

Wainwright, P. e Kettani, H. (2019). An Analysis of Botnet Models. Em *Proceedings of the 2019 3rd International Conference on Compute and Data Analysis*, ICCDA 2019, páginas 116–121, New York, NY, USA. Association for Computing Machinery.

Wander, G., Bansal, M. e Kasliwal, R. (2020). Prediction and early detection of cardiovascular disease in South Asians with diabetes mellitus. Em *Diabetes & Metabolic Syndrome: Clinical Research & Reviews*, volume 14. Elsevier.

Wang, W., Shang, Y., He, Y., Li, Y. e Liu, J. (2019). BotMark: Automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors. *Information Sciences*, 511.

Wang, X. e Ramsbrock, D. (2013). The Botnet Problem. Em *Overview of System and Network Security: A Comprehensive Introduction*, capítulo: 8. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Which (2020). More than one billion Android devices at risk of malware threats. Disponível em: https://www.which.co.uk/news/2020/03/more-than-one-billion-android-devices-at-risk-of-malware-threats/. Acessado em 18/04/2020.

Wichers, C. R. (1975). The Detection of Multicollinearity: A Comment. *The Review of Economics and Statistics*, 57(3):366–368.

Wired (2014). An Unprecedented Look at Stuxnet, the World's First Digital Weapon. Disponível em: https://www.wired.com/2014/11/countdown-to-zero-day-stuxnet/. Acessado em 18/04/2020.

Zhao, D., Traore, I., Sayed, B., Lu, W., Saad, S., Ghorbani, A. e Garant, D. (2013). Botnet detection based on traffic behavior analysis and flow intervals. Em *Computers & Security*, volume 39, páginas 2–16. ACM.