UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE COMPUTAÇÃO - ICOMP
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

KAIO RAFAEL DE SOUZA BARBOSA

Detecção de Redes de Serviço de Fluxo Rápido Baseada em Otimização por Colônia de Formiga

Manaus
2018

KAIO RAFAEL DE SOUZA BARBOSA

Detecção de Redes de Serviço de Fluxo Rápido Baseada em Otimização por Colônia de Formiga

Tese de Doutorado submetida ao Programa de Pós-Graduação em Informática do Instituto de Computação da Universidade Federal do Amazonas, como requisito parcial para obtenção do título de Doutor em Informática.

Orientador: Prof. Dr. Eduardo James Pereira Souto
Coorientador: Prof. Dr. Eduardo Luzeiro Feitosa

Manaus
2018

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Barbosa, Kaio Rafael de Souza
B238d Detecção de Redes de Serviço de Fluxo Rápido Baseada em Otimização por Colônia de Formiga / Kaio Rafael de Souza Barbosa. 2018
162 f.: il.; 31 cm.

Orientador: Eduardo James Pereira Souto
Coorientador: Eduardo Luzeiro Feitosa
Tese (Doutorado em Informática) - Universidade Federal do Amazonas.

1. Botnet. 2. Colônia de Formiga. 3. Fast-Flux. 4. Segurança. I. Souto, Eduardo James Pereira II. Universidade Federal do Amazonas III. Título

PODER EXECUTIVO
MINISTÉRIO DA EDUCAÇÃO
INSTITUTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

UFAM

FOLHA DE APROVAÇÃO

"Detecção de Redes de Serviço de Fluxo Rápido Baseada em Otimização por Colônia de Formiga"

KAIO RAFAEL DE SOUZA BARBOSA

Tese de Doutorado defendida e aprovada pela banca examinadora constituída pelos Professores:

Prof. Eduardo James Pereira Souto - PRESIDENTE

Profa. Eulanda Miranda dos Santos - MEMBRO INTERNO

Prof. Horácio Antonio B. Fernandes de Oliveira - MEMBRO INTERNO

Prof. José Luiz de Souza Pio - MEMBRO EXTERNO

Prof. Arthur de Castro Callado - MEMBRO EXTERNO

Manaus, 04 de Abril de 2018

AGRADECIMENTOS

Em primeiro lugar agradeço a Deus por ter me dado forças durante esses anos de realização de doutorado. Houveram momentos em que a fé foi mais importante que tudo. Não posso deixar de agradecer à minha família, minha esposa Diana, meu amado filho Felipe, minha mãe Lourdes e minha irmã Karol, assim como todos os familiares que estiveram comigo e puderam compreender a importância desse projeto de vida. Ao meu orientador Prof. Eduardo Souto, só posso lhe dizer muito obrigado por todos esses anos de parceria, amizade e profissionalismo na condução dos nossos projetos. Da mesma maneira, quero agradecer ao Professor Eduardo Feitosa pela ajuda na realização deste trabalho. Não posso esquecer de um grande parceiro durante a construção deste trabalho, obrigado Marcos Denis. Aos meus amigos, Haline, Edma, Davi Viana, Karla, Adeilson, Maely, Eduardo, Mary, Maiara, Rayol, Thiago, Wesllen, Alex e muitos outros, obrigado. Ao meu grande amigo João Luis, só posso lhe dizer, obrigado "minu". Quero também agradecer ao meu amigo e mentor Henry Vieira pelas inúmeras conversas sobre aprendizagem de máquina, obrigado. Agradeço também o trabalho realizado pela secreatia do IComp. Agradeço aos professores do IComp que contribuíram na minha formação durantes esses anos de doutoramento. Finalmente, não posso deixar de agradecer à Fundação de Amparo à Pesquisa do Estado do Amazonas por financiar parcialmente este trabalho.

RESUMO

O controle e o acesso remoto de computadores infectados por códigos maliciosos permitem ao operador desse tipo de rede (*botnet*) realizar diferentes atividades fraudulentas como orquestrar ataques distribuídos de negação de serviço (`DDoS`) ou propagar códigos maliciosos como vírus e *worms*. Para manter o controle dessas máquinas infectadas, é necessário utilizar um mecanismo de comunicação robusto contra tentativas de interrupção dos serviços da rede e que seja capaz de evadir sistemas de detecção de intrusos. Tal mecanismo é também conhecido como canal de Comando e Controle (C&C). Para isso, algumas redes maliciosas adotam com frequência o Sistema de Nomes de Domínios (DNS) devido ao seu funcionamento global e distribuído, permitindo assim que simulem comportamentos de redes legítimas a partir de técnicas como *Round-Robin* DNS (RRDNS) e Redes de Distribuição de Conteúdo (CDN). Redes maliciosas que empregam essas estratégias são denominadas como Redes de Serviço de Fluxo Rápido, pois são capazes de modificar seu comportamento para garantir a operação contínua dos serviços, assim como do canal de Comando e Controle (C&C). Para identificar essas redes, os sistemas de detecção de intrusos atuais são construídos a partir de modelos baseados em um conjunto fixo de atributos observados em determinado instante de tempo. No entanto, os operadores dessas redes são capazes de subverter tais modelos de detecção pela modificação de características como a quantidade de endereços IP ou tempo de vida (TTL) de um nome de domínio. Por esses motivos, este trabalho apresenta um modelo bioinspirado no conceito de Otimização por Colônia de Formigas para detecção de *botnets* baseadas em Redes de Serviço de Fluxo Rápido. O principal objetivo é analisar um domínio suspeito a partir de diferentes perspectivas, pois mesmo que seja possível a manipulação de determinadas características, é improvável que o operador modifique um conjunto considerável de atributos para evadir diferentes modelos de classificação ao mesmo tempo. Os resultados experimentais usando uma base de dados real mostram que o modelo é capaz de gerar regras de classificação que priorizam menor custo a partir da combinação de diferentes métodos de detecção, obtendo uma acurácia superior a 93%.

Palavras-chave: Botnets. Detecção de Tráfego Suspeito. Análise do Tráfego DNS.

ABSTRACT

Remote control and remote access of malicious code-enabled computers allow the network operator (botnet) to perform various fraudulent activities such as orchestrating distributed denial of service (DDoS) attacks or propagating malicious code such as virus and IT worms. To maintain control of these infected machines, it is necessary to use a robust communication mechanism against attempts to disrupt network services and to be able to evade intrusion detection systems. Such a mechanism is also known as Command and Control (C&C) channel. To do this, some malicious networks often adopt the Domain Name System (DNS) because of its global and distributed operation, allowing them to simulate legitimate network behaviors from techniques such as Round-Robin DNS (RRDNS) and Content Distribution Networks (CDN). Malicious networks that employ these strategies are called Fast Flow Service Networks, because they are able to modify their behavior to ensure the continuous operation of the services, as well as the Command and Control (C&C) channel. To identify such networks, current intrusion detection systems are constructed from models based on a fixed set of attributes observed at a given time point. However, the operators of these networks are able to subvert such detection models by modifying characteristics such as the number of IP addresses or the lifetime (TTL) of a domain name. For these reasons, this work presents a bioinspired model in the concept of Optimization by Colony of Ants for detection of botnets based on Fast Flow Service Networks. The main objective is to analyze a suspicious domain from different perspectives, because even if it is possible to manipulate certain features, the operator is unlikely to modify a of attributes to evade different classification models at the same time. The experimental results using a real database show that the model is able to generate classification rules that prioritize lower cost from the combination of different detection methods, obtaining an accuracy of more than 93%.

Keywords: Botnets. DNS Traffic Analisis. Suspicious Traffic Detection.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1

Introdução

Nos últimos anos, os códigos maliciosos (ou *malware*) deixaram de ter como função principal danificar ou tornar inoperantes os sistemas e máquinas alvos de seus ataques. Em geral, os sistemas computacionais são infectados principalmente para o roubo de informações ou para o sequestro dos recursos computacionais disponíveis nestes sistemas sem afetar o comportamento usual desses dispositivos. Neste modelo de atividade maliciosa, quanto maior for a escala de infecção do *malware*, maior será o poder computacional disponível aos seus criadores.

Um *malware* pode ser definido como um software que "deliberadamente" tem como objetivo realizar uma ação prejudicial de um atacante (Ye et al., 2017). Na literatura sobre evolução de códigos maliciosos (Tiirmaa-Klaar et al., 2013; Ye et al., 2017) é reportado que os primeiros *malwares* identificados não ofereciam ao atacante algum tipo de controle remoto aos hosts infectados. Tal limitação de acesso dificultava eventuais correções no funcionamento do softwares maliciosos. No entanto, a maioria das versões de *malwares* é provida de mecanismos para maximizar ou manter o controle dos dispositivos infectados[1].

A partir do momento que um atacante é capaz de controlar um conjunto distinto de máquinas infectadas, esses dispositivos são associados à uma rede maliciosa, denominada como *botnet*. A principal definição sobre *botnet* faz referência ao conjunto de máquinas comprometidas, denominadas de *bots*, as quais permite a um controlador humano, conhecido como *botmaster*, controlar remotamente os recursos computacionais e realizar atividades fraudulentas ou ilícitas (Liu et al., 2018).

[1]As palavras dispositivo, host e máquina são intercambiáveis quando fazem menção às máquinas infectadas

É importante notar que as *botnets* têm sido utilizadas por criminosos para promover ataques a um diversificado número de alvos incluindo provedores de diferentes serviços como jogos, hospedagem,computação em nuvem, instituições governamentais e privadas que dependem fortemente da Internet. A utilização constante desse tipo de rede maliciosa por parte dos atacantes é devido ao imenso poder computacional obtido a partir das máquinas comprometidas. Um clássico exemplo de código malicioso capaz de infectar um grande número de máquinas é o *malware Storm*, que se espalhou por mais de dois milhões de computadores e forneceu ao seu proprietário um poder computacional agregado superior ao de um supercomputador (Colajanni et al., 2008).

Para reforçar a efetividade dos ataques contra alvos de maior escala, *botmasters* necessitam aumentar o número de *bots* e, para isso, empregam diferentes técnicas para espalhar códigos maliciosos. As versões iniciais de códigos implementados para tornar uma máquina infectada não eram completamente automatizadas e necessitavam de interação humana como clicar em um endereço de `URL` para conduzir um download de um código malicioso, ou abrir um e-mail com um software malicioso anexado.

Versões mais recentes de códigos maliciosos modificaram tal procedimento, permitindo que as etapas de infecção e propagação sejam cada vez mais automatizadas. Por exemplo, o código malicioso `SDBot` (Ye et al., 2017) utiliza propagação por meio de troca de arquivos locais como no caso de dispositivos USB ou através do compartilhamento de arquivos em redes ponto-a-ponto (`P2P`), do inglês *peer-to-peer*. No entanto, outros tipos de códigos maliciosos buscam explorar vulnerabilidades nos serviços de rede como no caso do *malware Conficker* (Yadav et al., 2012).

Após infecção de um host, a comunicação e todos os procedimentos de interação entre o *botmaster* e os *bots* são realizados através de mecanismo de comunicação, denominado como canal de Comando e Controle (C&C). Do ponto de vista do atacante, a operação do C&C deve ser sigilosa e resiliente contra intervenções de administradores de redes ou responsáveis pela segurança da mesma. Caso um canal de C&C seja revelado ou identificado, é possível extrair informações sobre o funcionamento da rede incluindo a identificação de hosts infectados conduzindo para a desarticulação das atividades da *botnet*.

Devido à importância do canal de C&C, os *botmasters* podem combinar diferentes estratégias para manter esse mecanismo de comunicação em operação. Entre as estratégias adotadas, é importante citar o uso de aplicações válidas e diferentes protocolos de rede incluindo `HTTP`, `IRC`, `P2P` e `DNS` (Barbosa et al., 2014). Vale ressaltar que o tráfego `DNS` representa papel essencial em algum estágio do ciclo de vida da *botnet*, seja para identificar novas vítimas ou fornecer o endereço do canal de comando e controle.

Para encontrar o endereço e acessar o canal de C&C, um host infectado pode consultar nomes de domínios criados dinamicamente seguindo algum critério definido por um algoritmo de geração de nomes *Domain Generation Algorithm* (DGA). Esse tipo de resolução dinâmica de nomes também é conhecida como domínios de fluxo rápido ou *domain-flux* (Jiang et al., 2010). Adicionalmente, um endereço do C&C pode estar associado à redes que operam como serviço proxy, onde o principal objetivo é esconder o real endereço do canal de comunicação. Tal prática rede é definida como redes de serviço de fluxo rápido (*fast-flux service networks - FFSN*) ou IP-*flux* (Lombardo et al., 2018)

A ideia principal das redes de fluxo rápido é a operação dos *bots* como *encaminhadores* (`proxy`) entre as requisições de possíveis vítimas e as respostas dos servidores de conteúdo maliciosos. Mais precisamente, os *bots proxies* são conhecidos como agentes de fluxo, pois são frequentemente substituídos por outro conjunto de hosts para garantir alta disponibilidade da *botnet* (Al-Duwairi e Al-Hammouri, 2014).

Nas redes de serviço de fluxo rápido, cada consulta ao domínio malicioso corresponde a um endereço `IP` do agente (*bot*) de fluxo (Lombardo et al., 2018). O comportamento das `FFSN` é semelhante às redes de distribuição de conteúdo (`CDN`) e aos sistemas que utilizam o `DNS` para fazer balanceamento de carga como *Round-Robin DNS* (`RRDNS`). No entanto, as redes de fluxo rápido são utilizadas para hospedar e disseminar conteúdo malicioso da *botnet* e para proteger o real endereço do conteúdo malicioso.

A combinação de estratégias como `DGAs` e `FFSNs` permite que a operação entre o *botmaster* e os hosts infectados seja mais resiliente. Por esse motivo, operadores desse tipo de rede maliciosa utilizam frequentemente técnicas de fluxo rápido na operação e gestão de campanhas de spam on-line e para garantir o acesso ao canal de comando e controle (Al-Duwairi e Al-Hammouri, 2014).

Para tentar mitigar o avanço desse tipo de *botnet*, diferentes características podem ser utilizadas. Por exemplo, a partir da análise passiva do tráfego DNS é possível observar a data de criação do domínio[2], quantidade de endereços IP que respondem por um domínio, proximidade entre as redes que hospedam o conteúdo e análise léxica do nome consultado. Tais atributos permitem entender questões como quanto tempo um domínio está operacional, se os endereços IP que respondem pelo domínio fazem parte do mesmo segmento de rede ou pertencem à mesma empresa e se o domínio possui um nome com algum tipo de significado (Huang et al., 2010; Stone-Gross et al., 2009; Yadav et al., 2012).

No caso da investigação por análise ativa, é necessário que exista algum tipo de interação com a *botnet* para extrair suas características. Por exemplo, o tempo de viagem de ida e de volta (`RTT`) de um pacote pode ser utilizado como indicador de uma possível

[2]A partir da base `Whois`.

rede de serviço de fluxo rápido, pois o RTT de uma rede legítima é menor (Castelluccia et al., 2009). Outra característica utilizada nesse tipo de análise é enumerar a quantidade de endereços IP que respondem por um nome de domínio a partir do envio repetido de consultas DNS.

É importante deixar claro que o problema na detecção de Redes de Serviço de Fluxo Rápido é que a coleta de algumas características de rede pode tornar o processo de identificação e classificação desse tipo de rede maliciosa inviável, em virtude do tempo necessário para obter esses dados. Por exemplo, devido ao número de máquinas infectadas por uma *botnet* ao redor do globo, o total de endereços IP associados a um domínio malicioso é um atributo que demanda tempo adicional para coleta, pois é preciso considerar a disponibilidade dessas máquinas assim como diferentes tipos de fusos horários.

Outro desafio nesse tipo de detecção é que as redes de serviço de fluxo rápido operam de maneira semelhante a outros serviços de rede como CDN e RRDNS. Portanto alguns trabalhos tentam identificar esse tipo de estrutura de rede observando um conjunto mais abrangente de características. Por exemplo, no trabalho em (Stevanovic et al., 2016), as redes de serviço de fluxo rápido são investigadas a partir da análise de 62 características divididas em 4 componentes: análise gráfica, análise léxica do nome de domínio, análise de características de rede e análise em listas negras. Cada componente demanda janelas de monitoramento distintas, das quais representam no total uma semana para coletar, extrair e classificar esse tipo de rede.

Para tentar identificar as FFSN, outros trabalhos investigam o comportamento de domínios suspeitos a partir de longos períodos de monitoramento do tráfego DNS (Antonakakis et al., 2010, 2011; Bilge et al., 2014). A principal premissa desses trabalhos é reduzir as chances de manipulação do comportamento das *botnets* como demonstrado em (Knysz et al., 2011). Isto é, embora seja possível alterar algumas características da rede maliciosa, ainda existem outros atributos de rede que para serem manipulados demandariam esforço adicional por parte do *botmaster*, como exemplo, a quantidade de endereços IP compartilhados entre nomes de domínio ou similaridade de consultas por nome de domínio.

Na direção oposta a longos períodos de monitoramento, também existem trabalhos que conseguem extrair características de rede em poucos segundos como em (Holz et al., 2008), (Caglayan et al., 2009b) e (Zhao e Jin, 2015). Nesses casos, a detecção de redes de serviço de fluxo rápido alcança valores de acurácia acima de 88%. No entanto, a maioria dos atributos utilizados nesses trabalhos pode ser manipulada e, consequentemente, os métodos de detecção subvertidos.

Para tentar entender o uso dos atributos de rede no processo de detecção de redes de serviço de fluxo rápido, Barbosa et al. (2014) demonstram como as características são utilizadas por esse tipo de *botnet*. Os resultados mostram que é possível identificar um conjunto de atributos que são combinados estaticamente por diferentes autores na classificação desse tipo redes. Além disso, é possível entender que as soluções atuais não são capazes de acompanhar a evolução no comportamento das *botnets*, pois os operadores de redes maliciosas estão em constante busca por novos mecanismos que facilitem burlar os sistemas de detecção de intrusos.

Assim, o problema de detecção de redes de serviço de fluxo rápido pode ser tratado em um modelo hipotético da seguinte maneira: se o tempo total do processo de extração de dados até a classificação de um domínio suspeito for desconsiderado, então os problemas de *botnets* podem ser resolvidos a partir de um número considerável de características que são obtidas dentro de uma janela de monitoramento compatível com tempo de coleta. Por outro lado, tal modelo hipotético não pode ser aplicado no mundo real pois durante ataques de rede, disseminação de código malicioso e recebimento de *spam* em massa, o objetivo é identificar e interromper tais comportamentos no menor tempo possível para que os serviços de rede não sejam paralisados ou novos hosts infectados.

Sabendo que operadores de rede maliciosa podem manipular o comportamento das *botnets*, diferentes trabalhos buscam identificar novos atributos e estratégias para mitigar o avanço dessas redes. É possível assumir que tal processo gera um ciclo de busca constante, onde é preciso identificar qual melhor conjunto de atributos ou método de detecção que permita classificar as redes maliciosas após mudança de comportamento.

Por esses motivos, este trabalho apresenta uma nova tratativa para detecção de *botnets* baseadas em redes de serviço de fluxo rápido. Isto é, enquanto outros trabalhos na literatura buscam identificar atributos para serem utilizados em diferentes métodos de detecção, este trabalho observa a detecção desse tipo de rede a partir de diferentes perspectivas, onde cada ponto de vista pode ser um método de detecção distinto.

A principal ideia que sustenta essa nova tratativa está na própria dinâmica adotada por operadores de rede maliciosas para prolongar o funcionamento da *botnet*. Pois, é razoável assumir que o *botmaster* seja capaz de manipular determinados comportamentos. No entanto, tanto do ponto de vista da natureza desse tipo de rede como na questão econômica, é impraticável que operador consiga alterar as características de rede durante longos períodos de monitoramento para burlar diferentes métodos de detecção. Portanto, para lidar com os desafios apontados na identificação de *botnets* de fluxo rápido, este trabalho propõe um *framework* capaz de correlacionar diferentes métodos de detecção para identificar as redes de serviço de fluxo rápido.

Para que a correlação dos métodos de detecção seja eficaz, é preciso estabelecer algum tipo de restrição a este processo. Isto é, o objetivo aqui é classificar redes de serviço de fluxo rápido no menor tempo possível obtendo máximo de acurácia. Desta maneira, uma vez definido o critério de avaliação, o nome de domínio é investigado até que uma resposta seja encontrada.

A correlação de diferentes métodos de detecção a partir de determinados critérios denota um problema de otimização combinatória, pois é preciso encontrar uma possível solução ótima dentro de um espaço de busca de soluções. Para resolver esse problema, este trabalho utiliza os conceitos de *Otimização por Colônia de Formigas* ou `ACO` *Ant Colony Optimization* (Dorigo e Stützle, 2004). Até onde vai o conhecimento deste autor, este é o primeiro trabalho que utiliza o sistema de formigas para resolução de *botnets* baseadas em redes de serviço de fluxo rápido.

1.1 Motivação

Embora existam alternativas para mitigar o avanço das `FFSNs` como *i*) análise léxica do nome do domínio (Antonakakis et al., 2012; Mowbray e Hagen, 2014); *ii*) análise de fluxo de rede (Haq et al., 2014; Lin et al., 2009); engenharia reversa do *malware* (Sinha et al., 2010; Stone-Gross et al., 2009); *iii*) análise de características temporais de rede (Nazario e Holz, 2008; Wu et al., 2010); e *iv*) análise de características espaciais (Hsu et al., 2010; Lin et al., 2013), tais estratégias demonstram limitações para acompanhar o processo evolutivo adotado por *botmasters*.

Para tratar o problema de detecção de *botnets* baseadas em redes de serviço de fluxo rápido, é preciso uma estratégia que permita observar esse problema a partir de diferentes perspectivas fornecidas pelos diferente métodos de detecção dessas redes maliciosas. Isto é, a principal sustentação para essa premissa é que mesmo que o *botmaster* seja capaz de burlar uma estratégia de detecção como demonstrado em (Knysz et al., 2011), ainda existirão outros pontos de vistas que podem ser utilizados para analisar e identificar o mesmo problema.

Partindo desse princípio, é preciso definir uma metodologia que seja capaz de buscar possíveis soluções a partir de um conjunto de maior abrangência de soluções e que considere ainda as restrições relacionados ao menor tempo de detecção e maior acurácia. Caso contrário, um leitor desavisado pode acreditar que um conjunto de métodos de detecção selecionados aleatoriamente é capaz identificar as redes de serviço de fluxo rápido atendendo as restrições apresentadas aqui.

Vale ressaltar que a partir do problema descrito, é possível estabelecer uma analogia com problemas de otimização combinatória, isto é, dentro de um espaço de soluções (diferentes métodos de detecção e seus atributos), deseja-se encontrar aquela que seja próxima da ótima para resolver o problema em questão. Além disso, um problema de otimização combinatória apresenta três componentes básicos (função objetivo, espaço de soluções e restrições do problema) que definem como o processo de correlação dos métodos de detecção deve acontecer.

Entre os algoritmos utilizados para resolver os problemas de otimização combinatória a Otimização por Colônia de Formigas (ACO) é utilizada, pois tem sido aplicada em diversos trabalhos que envolvem problemas em redes de computadores (Dorigo e Stützle, 2004; Gajpal e Abad, 2009), detecção de anomalias de redes (Ranga e Mandhar, 2018) e detecção hosts infectados na rede (Wu et al., 2018), assim como em problemas com múltiplos objetivos (Mokhtari e Ghezavati, 2018).

Na literatura, a adoção do conceitos de colônias de formigas para detecção de *botnets* ainda é inicial e os trabalhos estão direcionados na identificação dos endereços IP durante ataques de negação de serviço (`IP traceback`) (Wang et al., 2011, 2010) e no agrupamento de comportamentos semelhantes (Shelokar et al., 2004). No entanto, até onde vai o conhecimento deste autor, nenhum trabalho utiliza a Otimização por Colônia de Formigas para detectar *botnets* baseadas em Redes de Serviço de Fluxo Rápido.

Por essa razão, a utilização de colônias de formigas é uma proposta pioneira para tratar esse tipo de problema, pois de maneira similar ao comportamento natural das formigas que buscam por novas fontes de alimento (Dorigo e Stützle, 2004), a detecção de *botnets* também deve utilizar mecanismos adicionais para mitigar o avanço desse tipo de rede maliciosa.

Vale lembrar que durantes ataques de rede, o principal objetivo é identificar a origem do evento para interrupção imediata. Por exemplo, no caso do código malicioso *Conficker*, inúmeras máquinas foram infectadas e ingressaram em uma grande rede maliciosa, da qual também simulava redes de serviço de fluxo rápido (Irwin, 2012). No estudo sobre o comportamento da *botnet Mirai* (Antonakakis et al., 2017), os pesquisadores demonstraram que durante as primeiras 20 horas após a descoberta da anomalia, mais de 65 mil dispositivos de Internet das Coisas (`IoT`) haviam sido sequestrados para gerar ataques contra diversos serviços de Internet por meio do protocolo `DNS`. Após 5 meses de monitoramento foi possível constatar que mais de 300 mil hosts haviam sido infectados por essa *botnet* (Antonakakis et al., 2017).

Diante desses cenários, a principal motivação é identificar dentro de um conjunto de métodos de detecção de redes maliciosas, uma combinação que permita identificar redes

de serviço de fluxo rápido considerando os cenários expostos aqui. Assim, para lidar com essas questões e oferecer novas contribuições nesta área de pesquisa, esta Tese apresenta *framework* para classificação de *botnet* baseadas em redes de serviço de fluxo rápido através de colônia de formigas.

1.2 Objetivos

Aprimorar o processo de detecção de redes de serviço de fluxo rápido por meio de um *framework* inspirado em Otimização por Colônia de Formiga capaz de correlacionar diferentes métodos de detecção de *botnets* considerando o menor custo e obtendo a maior acurácia.

Para alcançar o objetivo geral deste trabalho, um conjunto de etapas deve ser seguido, proporcionando fundamentos para atingir a meta final pretendida:

1. Formalizar o problema de detecção de redes de serviço de fluxo rápido considerando a correlação de métodos de detecção a partir da restrição do problema;
2. Definir um modelo representativo inspirado em Otimização por Colônia de Formiga que consiga atender as relações entre métodos de detecção;
3. Avaliar os principais métodos e suas características de rede mais relevantes;
4. Construção e adaptação de um algoritmo bioinspirado que seja capaz de utilizar os recursos de rede disponíveis para classificação de redes de fluxo de serviço rápido.

1.3 Contribuição

A partir dos objetivos definidos neste trabalho, as seguintes contribuições serão complementos adicionais obtidas:

1. Uma representação formal de *botnets* baseadas em `FFSN` a partir do conceito de Otimização por Colônia de Formigas. A principal ideia nesse ponto é demonstrar como os componentes de uma rede maliciosa podem ser modelados em uma estrutura de dados que permita que diferentes métodos de detecção e seus atributos possam ser utilizados na avaliação de um nome de domínio suspeito.
2. Construção de um *framework* baseado no conceito de colônia de formigas capaz de definir um modelo onde um nome de domínio suspeito é avaliado por diferentes

métodos de detecção. Enquanto outros trabalhos estão direcionados na detecção de redes maliciosas utilizando uma tupla fixa composta de método de detecção e atributos, a construção de um *framework* vai além dessa estratégia, pois permite que diferentes métodos e atributos sejam combinados para investigar esse tipo de rede maliciosa a partir de ponto de vistas distintos.

3. Um método distribuído para extração de características de redes. As redes de serviços de fluxo rápido possuem um grande número de hosts infectados e para lidar com esse problema é necessário obter informações dessas redes a partir de pontos distribuídos ao redor do globo. Nesse sentido, este trabalho apresenta uma possível abordagem para comunicação entre diferentes agentes de coleta de tráfego por meio do protocolo `HTTP`.

4. Desenvolvimento de uma prova de conceito que demonstre que o *framework* proposto é capaz de detectar redes de serviço de fluxo rápido. Para facilitar o entendimento de outros pesquisadores, esta Tese de Doutorado apresenta uma prova de conceito desenvolvida na linguagem de programação *Python*.

1.4 Organização da Tese

O restante desse documento está organizado da seguinte maneira: O Capítulo 2 apresenta os conceitos e definições gerais sobre as *botnets*, incluindo métodos utilizados para propagação, ciclo de vida das *botnets*, como é feita a comunicação entre o operador da rede maliciosa e os *bots* e arquiteturas das *botnets*. Nesse capítulo também são demonstrados exemplos do funcionamento das *botnets* baseadas em redes de serviço de fluxo rápido. Além disso, o capítulo aborda as definições sobre problemas de Otimização Combinatória, Colônia de Formigas e ilustra os principais componentes utilizados para resolução de problemas através desta técnica.

O Capítulo 3 discute métodos e estratégias adotadas para detecção de *botnets* baseadas em fluxo rápido incluindo técnicas utilizadas por algoritmos de geração de domínios e de redes de serviço de fluxo rápido. O capítulo conclui apresentando trabalhos baseados em colônia de formiga para resolução de problemas de classificação.

O Capítulo 4 apresenta os principais componentes do *framework* baseado em otimização por colônia de formiga para detecção de redes de serviço de fluxo rápido. O capitulo introduz uma visão geral da arquitetura proposta e como os componentes interagem entre si. É preciso destacar o principal componente, denominado como `AFFACO-M`, responsável pela correlação e definição do modelo de detecção de redes de

serviço de fluxo rápido. Além disso, o Capítulo também apresenta o componente `AFFACO-P` que representa uma prova de conceito de que o modelo gerado é capaz de detectar redes de serviço de fluxo rápido no menor tempo e obtendo maior acurácia.

O Capítulo 5 apresenta o protocolo experimental e discute resultados obtidos durante a construção do *framework* proposto. Por exemplo, neste Capitulo é demonstrado como o número de formigas pode influenciar a geração do modelo de classificação, ou ainda, como os parâmetros necessários para correlação de métodos de detecção podem auxiliar as formigas na busca pelo próximo método a ser selecionado.

Finalmente, o último capítulo apresenta as principais conclusões obtidas neste trabalho, lições aprendidas e Trabalhos Futuros.

Além dos capítulos descritos aqui, esta Tese ainda apresenta os Anexos A, B e C. O Anexo A apresenta uma nova proposta de abordagem baseada em entropia denominada `EntropyFFNet`. Esta proposta é resultado de experimentos durante a coleta de atributos que estavam ausentes durante o período de coleta e que dificultaram o desenvolvimento dos métodos de detecção. No Anexo B é realizado um estudo de um segundo método denominado `BurdenMethod` que utiliza todas as características selecionadas durante a realização dessa Tese a partir dos algoritmos `Logistic Regression` e `Random Forest`. O Anexo C apresenta um estudo comparativo entre todos os métodos identificados na literatura e o modelo proposto por essa Tese de Doutorado.

2

Fundamentação Teórica

Em 2003, McCarty (McCarty, 2003) apresentou um dos trabalhos pioneiros na detecção de computadores infectados (máquinas zumbis). O autor observou as atividades de um conjunto de máquinas comprometidas ingressando em um servidor `IRC` . McCarty define tais máquinas como *bots* e o conjunto de *bots*, controlados por um ou mais atacantes, como uma *botnet*.

No início da automação dos serviços `IRC`, a função de um *bot* era o gerenciamento de usuários e controle de canais do servidor. No entanto, devido à flexibilidade e ao crescente número de computadores conectados à Internet, tais máquinas passaram a ser utilizadas para atividades ilegais como ataques de negação de serviços (Peng et al., 2007), envio de spam em massa (John et al., 2009), *phishing* (Souza et al., 2013), fraudes em sistemas de propaganda (Daswani e Stoppelman, 2007) ou aluguel de *botnets* (Studer, 2011).

Redes maliciosas mais recentes adotam abordagens evasivas para tentar escapar de sistemas detectores de intrusos e de métodos que dificultam a interrupção na comunicação entre criminosos e computadores infectados.

Para entender a dinâmica utilizada nesse tipo de rede, este Capítulo apresenta os principais conceitos, componentes, arquiteturas e protocolos utilizados pelas *botnets*. Além disso, para que o leitor entenda os conceitos de Otimização por Colônia de Formiga, este capítulo faz uma breve revisão das características que denotam um problema de otimização combinatória, envolvendo soluções de *maximização* ou *minimização* (Dorigo e Stützle, 2004).

2.1 Botnets

2.1.1 Definições

O conceito de *botnets* está associado ao conjunto de máquinas comprometidas que permitem ao atacante o controle remoto dos recursos computacionais para realizar atividades fraudulentas ou ilícitas (Freiling et al., 2005; McCarty, 2003). Tais atividades possuem forte relação com um mercado consumidor, onde, por exemplo, um cliente tem a possibilidade de contratar serviços oferecidos pela *botnet* como ataques de negação de serviço e envio de spam em massa.

A gestão e controle das operações de uma *botnet* são realizadas por uma entidade externa, definida como *botmaster* (Stone-Gross et al., 2009). Diferentes vetores de propagação como mensagens *spam*, vírus e *worms* são usados pelos operadores para sequestrar novos hosts para ingressar na *botnet*. Uma vez infectadas, as máquinas utilizam um software chamado de *bot* (da palavra robô - *robot*), o qual liga os computadores à uma infraestrutura de Comando e Controle (C&C).

Tal infraestrutura consiste dos *bots* e uma entidade de controle que pode ser centralizada ou distribuída. Diferentes protocolos como `HTTP`, `IRC`, `P2P` e `DNS` são usados pelos *botmasters* para controlar as máquinas infectadas e coordenar suas ações.

Além disso, a utilização de diferentes protocolos por uma mesma *botnet* possibilita à continuidade das operações da *botnet*, mesmo em situações de interrupções por via judicial (Sinha et al., 2010), sequestro do canal de Comando e Controle (Stone-Gross et al., 2009) ou contra-ataques de inundação dos *bots* (Davis et al., 2008). A Figura 2.1 apresenta os principais componentes do ecossistema de uma *botnet*.

Em geral, novos *bots* podem ser obtidos através de dois métodos: autopropagação (método ativo) ou propagação por indução (método passivo) de *malware* (Shin et al., 2011). Na autopropagação, o *bot* busca na rede outros dispositivos com vulnerabilidades que possam ser exploradas e que permitam acesso remoto. Na propagação passiva, técnicas de engenharia social (por exemplo, redirecionamento de URLs) tentam ludibriar o usuário para que o mesmo execute um *malware*. Em ambos os casos, após a infecção do dispositivo, a máquina busca o canal de C&C para notificar o *botmaster* e aguardar novas instruções.

Devido à importância do *malware* para operação da *botnet*, operadores da rede podem buscar em fóruns de discussões ocultos ou no mercado negro (Li e Chen, 2014), novos métodos de infecção (`ZeroDay`) e softwares maliciosos. Isso ocorre porque os desenvolvedores não são, necessariamente, os controladores da *botnet*. Nessas comunidades, *botmasters* e terceiros (clientes que desejam usar um *malware* como produto) podem

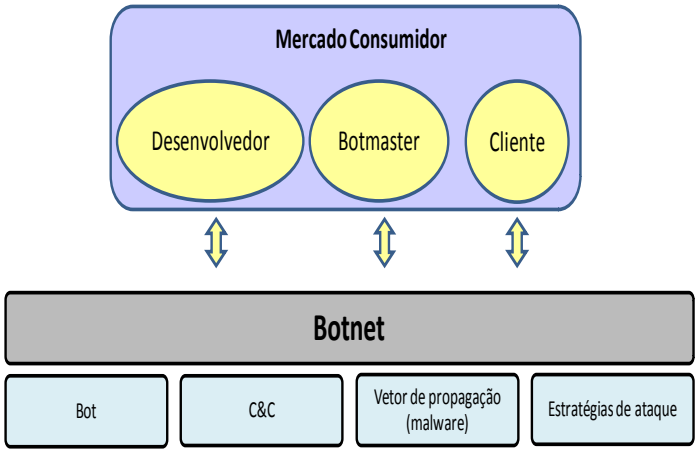


Figura 2.1: Principais componentes de um ecossistema de uma *botnet* (Fonte: Próprio Autor).

comprar aplicações maliciosas *prontas*, pacotes de software para a criação personalizada de executáveis ou módulos de extensão. A *botnet Zeus* (Binsalleeh et al., 2010) é um típico exemplo desse tipo de comércio de *malware*.

2.1.2 Ciclo de Vida

Para que uma *botnet* continue operando é importante que novos hosts sejam recrutados com frequência, pois uma vez identificados por sistemas de detecção, tais hosts são cadastrados em lista negra (Ishibashi et al., 2005), dificultando a propagação de software malicioso. Por essa razão, identificar hosts vulneráveis e comprometê-los é uma atividade fundamental para o sucesso de uma *botnet*.

O processo para encontrar hosts vulneráveis, explorá-los e torná-los membros da *botnet* é definido como ciclo de vida das *botnets*. Muitos trabalhos exploram diferentes fases desse ciclo para encontrar pontos de fraqueza e interromper as operações ilícitas da rede (Abu Rajab et al., 2006; Morales et al., 2009; Rodríguez-Gómez et al., 2013).

A Figura 2.2 ilustra o ciclo de vida de uma *botnet*. De maneira resumida, o ciclo de vida pode ser representado pelas seguintes etapas. No *Passo 1*, um membro da *botnet* identifica um host vulnerável na rede. Após a infecção desse host através de algum vetor de propagação (e.g, mensagem spam, vírus, worms, etc), consultas `DNS` são realizadas para encontrar o servidor que distribui software *bot* (*Passo 2*) No *Passo 3*, o host infectado baixa e instala o software *bot* (binário maliciosos) para, finalmente, ingressar no canal de comando e controle (*Passo 4*).

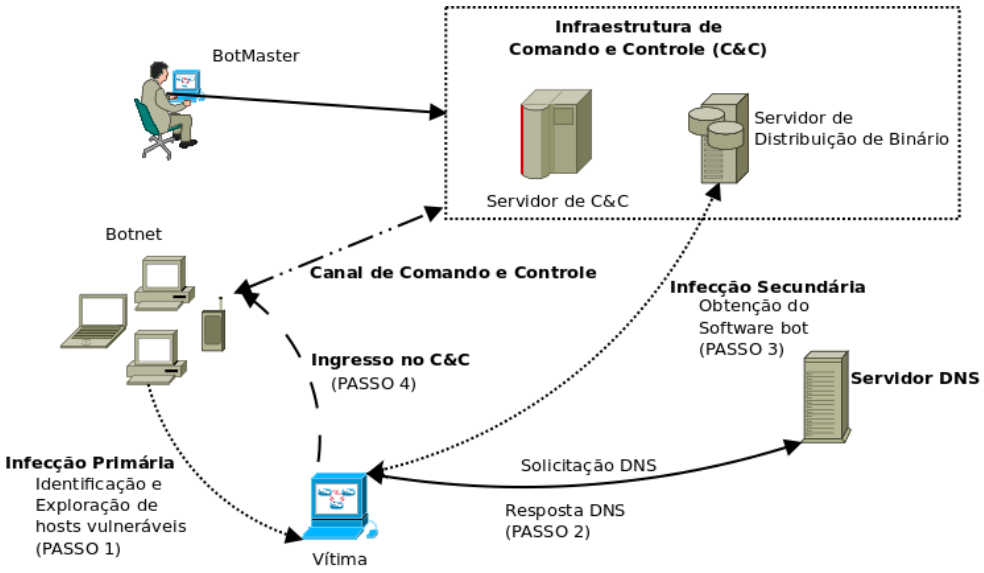


Figura 2.2: Ciclo de vida da *botnet* (Fonte: Próprio Autor).

Para alguns trabalhos (Bazrafshan et al., 2013; Li et al., 2009), o binário malicioso já faz parte do *malware* que infectou o dispositivo. Em contrapartida, outros trabalhos observaram que o host comprometido realiza o download desse binário em um segundo instante (infecção secundária), a partir de um servidor que armazena o software malicioso (Abu Rajab et al., 2006; Feily et al., 2009). Embora esses trabalhos divirjam quanto à presença do binário malicioso durante o estágio inicial de infecção, um dispositivo zumbi somente é útil para *botnet* a partir do momento em que o *botmaster* conhece sobre sua existência.

A vantagem em separar em instantes diferentes à aquisição do binário malicioso permite que a *botnet* seja capaz de oferecer versões específicas do *bot* para arquiteturas de hardware distintas, includindo dispositivos como celulares, notebooks e roteadores.

Já para encontrar o canal de Comando e Controle e unir-se aos demais membros da *botnet*, o *bot* realiza um procedimento de agrupamento, denominado como *rallying*. O termo define o momento em que um *bot* está se autenticando no servidor de comando e controle (Schiller e Binkley, 2007). Esse procedimento pode ser realizado através de uma abordagem estática ou dinâmica.

Na abordagem estática, o host comprometido utiliza o endereço IP do servidor de C&C que pode ser localizado no próprio código fonte utilizado para infectar o host (Liu et al., 2009). Embora tal estratégia seja simples, técnicas de engenharia reversa poderiam ser usadas para revelar o endereço IP do servidor de C&C, permitindo que a *botnet* fosse desativada e interrompida (Egele et al., 2008).

Na abordagem dinâmica, o *bot* consulta servidores DNS (comprometidos ou não) para encontrar o endereço IP que está associado ao nome de domínio que responde pelo C&C. Caso a conexão apresente problemas de comunicação, o *bot* envia uma consulta DNS para receber o novo nome de domínio do servidor C&C. É importante observar que essa abordagem permite ao atacante redirecionar sua rede de maneira rápida, caso servidores sejam retirados de operação.

Devido ao modelo simples e distribuído do protocolo DNS, *botmaster* usam tais características para evadir sistemas de detecção de intrusão. O controlador da *botnet* pode registrar um ou vários nomes de domínios e atribuí-los a um conjunto de máquinas infectadas que serão alcançadas através da resolução de nomes. O registro de nomes de domínios é um procedimento fácil e que pode ser realizado também através de provedores que fornecem esse serviço gratuitamente como `dyn.com`[1], `noip.com`[2] e `freedns.afraid.org`[3].

2.1.3 DNS como Mecanismo de Resiliência para Botnets

Durante um ataque distribuído de negação de serviço, os recursos computacionais das vítimas são exauridos ao ponto que o acesso aos serviços oferecidos ficam indisponíveis. Para reduzir o impacto desses ataques, muitas empresas na Internet adotam técnicas para melhorar o desempenho do DNS e a disponibilidade dos serviços legítimos. Duas dessas técnicas incluem o uso de *Round-Robin* DNS (RRDNS) e de serviços de redes de ditribuição de contéudo (*Content Delivery Network* - CDN) (Al-Duwairi e Al-Hammouri, 2014)

A técnica de *Round-Robin* DNS permite que o conteúdo seja distribuído uniformemente ao longo de um conjunto de máquinas. Para conseguir isso, cada vez que o mesmo nome de domínio é consultado, um único endereço IP é selecionado (usando um Round-Robin) a partir de um conjunto de endereços IP possíveis. Essa técnica aumenta a resiliência contra ataques DoS por meio do balanceamento do tráfego direcionado para os servidores. Já as CDNs tentam realizar o balanceamento de carga não só em máquinas diferentes no mesmo local, mas com máquinas espalhadas por diferentes localizações geográficas. A principal ideia das CDNs é distribuir as solicitações entre os servidores mais próximos ao cliente.

Assim como as técnicas apresentadas, os *botmasters* exploram o uso do DNS para gerenciar sua infraestrutura de ataques, migrando os seus servidores C&C com facilidade. Mais ainda, os *botmasters* podem fazer uso do Round-Robin DNS para associar os endereços IP a um nome de domínio ou usar técnicas baseadas em CDNs em suas próprias *botnets*, de modo que as requisições podem ser processadas por servidores diferentes mais

[1] http://dyn.com/
[2] http://www.noip.com/
[3] https://freedns.afraid.org/

próximos dos clientes (vítimas). Tais técnicas permitem às *botnets* alcançar um elevado nível de agilidade e aumentar a sua disponibilidade. Portanto, para tratar esse problema é imprescindível entender e diferenciar o comportamento legítimo das técnicas `RRDNS` e `CDNs` das redes *botnets*.

Em uma CDN, a distribuição de conteúdo pode ser parcial ou total (Su et al., 2009). No primeiro caso, apenas uma porção do conteúdo desejado está hospedado na `CDN`. No segundo, tanto o conteúdo quanto outros serviços podem estar hospedados inteiramente na rede de distribuição.

A Figura 2.3 apresenta o processo de consulta ao conteúdo parcial hospedado em uma `CDN`. No passo um, o cliente faz uma solicitação para o site que hospeda o conteúdo original. Em seguida, o servidor original retorna a pergunta fornecendo a referência para o provedor `CDN` que hospeda o conteúdo. No passo dois, o provedor `CDN` com autoridade recebe uma consulta `DNS` e, com base no endereço IP de origem, o cliente recebe como resposta um endereço IP mais próximo de sua localidade. Finalmente, o cliente acessa o conteúdo diretamente do servidor que hospeda o conteúdo, isto é, aquele servidor que possui menor distância topológica com o endereço de origem. Diversos provedores, includindo `CloudFlare`[4], `Google App Engine`[5] e `jsDelivr`[6], oferecem gratuitamente o serviço distribuição de conteúdo para algumas aplicações web como *WordPress* e *JQuery*.

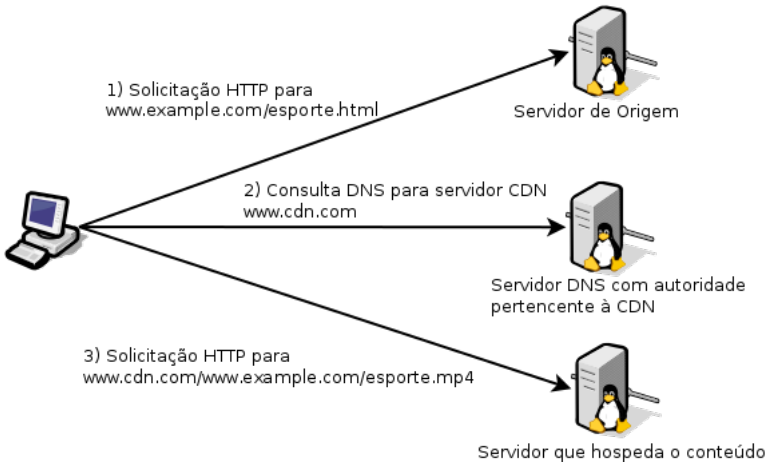


Figura 2.3: Exemplo de distribuição de conteúdo parcial para arquivos de vídeo (Fonte: Próprio Autor).

[4]https://www.cloudflare.com
[5]https://appengine.google.com/start
[6]http://www.jsdelivr.com

Para entender como a hospedagem de conteúdo total funciona, considere o exemplo de um site de noticias (*Estadão*[7]) exibido Figura 2.4.

Na seção de respostas (*answer section*), é possível observar que o subdomínio (`www`) é uma referência (`CNAME`) para outra referência de um endereço de serviço `CDN`. O registro de recurso `CNAME` é frequentemente utilizado por redes de distribuição de conteúdo, pois permite que um cliente mantenha o endereço original do domínio da sua marca, enquanto o conteúdo é hospedado em redes de terceiros.

Quando o domínio `www.estadao.com.br` é consultado por máquinas localizadas em alguma cidade (e.g. Manaus), a resposta obtida encontra endereços IPs roteados dentro do Brasil `.40` e `.41` da rede `200.143.247`. No entanto, a mesma consulta realizada em um servidor nos EUA retorna os endereços IPs `.75` e `.91` da classe de rede `184.51.102`. A vantagem da `CDN` para um cliente localizado no Brasil é que em apenas 4 saltos o conteúdo é acessado, enquanto o mesmo conteúdo precisa de 13 saltos caso o conteúdo esteja nos EUA.

```
; <<>> DiG 9.9.5-9-Debian <<>> www.estadao.com.br
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36884
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 8, ADDITIONAL: 9

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.estadao.com.br.            IN      A

;; ANSWER SECTION:
www.estadao.com.br.     226     IN      CNAME   estadao.edgesuite.net.
estadao.edgesuite.net.  14035   IN      CNAME   a1638.g.akamai.net.
a1638.g.akamai.net.     20      IN      A       200.143.247.41
a1638.g.akamai.net.     20      IN      A       200.143.247.40

;; AUTHORITY SECTION:
g.akamai.net.           433     IN      NS      n2g.akamai.net.
g.akamai.net.           433     IN      NS      n3g.akamai.net.
g.akamai.net.           433     IN      NS      n1g.akamai.net.
g.akamai.net.           433     IN      NS      n0g.akamai.net.
g.akamai.net.           433     IN      NS      n4g.akamai.net.
g.akamai.net.           433     IN      NS      n5g.akamai.net.
g.akamai.net.           433     IN      NS      n7g.akamai.net.
g.akamai.net.           433     IN      NS      n6g.akamai.net.

;; ADDITIONAL SECTION:
n0g.akamai.net.         433     IN      A       200.182.35.151
n1g.akamai.net.         2433    IN      A       88.221.81.194
n2g.akamai.net.         4433    IN      A       200.160.97.134
n3g.akamai.net.         433     IN      A       200.143.247.45
n4g.akamai.net.         4444    IN      A       161.22.123.54
n5g.akamai.net.         430     IN      A       200.182.35.158
n6g.akamai.net.         433     IN      A       200.160.97.135
n7g.akamai.net.         2433    IN      A       200.143.247.46

;; Query time: 106 msec
;; SERVER: 10.208.0.1#53(10.208.0.1)
;; WHEN: Fri May 22 11:17:30 AMT 2015
;; MSG SIZE  rcvd: 415
```

Figura 2.4: Consulta DNS para o domínio www.estadao.com.br que está hospedado em um provedor CDN (Fonte: Próprio Autor).

[7]http://www.estadao.com.br

Além de distribuir endereços próximos da origem, as CDNs utilizam o *Tempo de Vida* (TTL) do registro de recurso relativamente baixo. Tal estratégia permite que o resolvedor do cliente realize novas consultas DNS a partir do momento que o dado for expirado (Stamos et al., 2010). Diversos autores descrevem que o número de IPs obtidos a partir de novas consultas tende a convergir para um número estável (Knysz et al., 2011; Lin et al., 2013; Su et al., 2009), isto é, após um determinado tempo, a quantidade total de endereços IPs é conhecida.

Em comparação com as CDNs, as RRDNS, através do tráfego DNS, permitem distribuir endereçamentos IPs que estejam com maior disponibilidade para processamento de carga (Borkar et al., 2011). As RRDNS podem distribuir endereços IPs de redes distintas, visando melhorar o processamento da carga, ao invés do menor custo de acesso à informação. Além disso, consultas DNS a partir de localidades distintas não alteram o total de endereços ou a distribuição de IPs obtidos em cada resposta.

A Figura 2.5 ilustra um exemplo de um domínio (www.debian.org) hospedado em uma RRDNS. É possível observar que as classes de redes são distintas e pertencem a países como Austrália, Reino Unido, Nova Zelândia, Brasil e EUA. Vale ressaltar que o mesmo conjunto de endereços IPs também pode ser obtido a partir de consultas em outros países como EUA e Reino Unido.

```
;; ANSWER SECTION:
debian.org.             98      IN      A       150.203.164.38
debian.org.             98      IN      A       128.31.0.62
debian.org.             98      IN      A       200.17.202.197
debian.org.             98      IN      A       5.153.231.4
debian.org.             98      IN      A       130.89.148.14
debian.org.             98      IN      A       140.211.15.34
```

Figura 2.5: Total de endereços IPs que respondem pelo domínio do site do sistema operacional Debian GNU/Linux (Fonte: Próprio Autor).

Como já observado, as CDNs e RRDNS são técnicas utilizadas por sites legais para garantir resiliência e acesso aos serviços oferecidos. Por esses motivos, operadores de *botnets* adotam as mesmas estratégias para que seu comércio ilegal continue operando. Como observado em (Holz et al., 2008), *botnets* que oferecem serviços de venda de produtos farmacêuticos precisam disponibilizar seu site para que possíveis clientes realizem a compra desse material. De maneira semelhante, como indicado em (Chen et al., 2013a; Hands et al., 2015; Wu et al., 2010), ataques de *phishing*, para terem êxito, necessitam que a página maliciosamente forjada esteja em pleno funcionamento.

Por esses motivos, fica claro que as características das CDNs e RRDNS são fundamentais para a operação contínua das redes maliciosas. *Botnets* que adotam tais estratégias são

conhecidas como *botnets* baseadas em redes de serviço de fluxo rápido e serão discutidas a seguir.

2.2 Botnets baseadas em Serviços de Fluxo Rápido

Para evadir os sistemas de detecção, as *Botnets* baseadas em Serviços de Fluxo Rápido trocam rapidamente tanto os endereços IPs que respondem pelo domínio quanto o nome de domínio malicioso. Essa rápida troca dos endereços IPs é conhecida como *IP-Flux*, enquanto *Domain-Flux* reflete as mudanças no nome de domínio.

Ambos, *IP-Flux* e *Domain-Flux,* fornecem níveis avançados de redundância e resiliência para a infraestrutura de C&C de uma *botnet*. Para detectar *botnets* baseadas em fluxo rápido, é preciso fazer uma análise minuciosa das duas técnicas.

2.2.1 Troca Rápida de IP: IP-Flux

A partir das redes *fast-flux*, o *botmaster* pode controlar o comportamento do domínio malicioso usando as técnicas de *single-flux* ou *double-flux*. É importante notar que em ambas as técnicas, o principal objetivo é utilizar diversos *bots* para responder pelo domínio solicitado (Bazrafshan et al., 2013; Caglayan et al., 2009a; Campbell et al., 2011; Lin et al., 2013). Enquanto num domínio legítimo o acesso é realizado através de computadores destinados para esse tipo de serviço, em *botntes* baseadas em fluxo rápido, a vítima envia informações às máquinas infectadas que são responsáveis pelo recebimento e encaminhamento para o verdadeiro servidor destino, também conhecido como proxy de redirecionamento (*blind proxy rediction*)(Zhou et al., 2008).

As trocas rápidas de endereços IPs ocorrem devido ao comportamento do protocolo DNS. Em uma típica configuração de zona de domínio (e.g.: `example.com`), a tupla (`Nome`, `TTL`, `Classe`, `RR`, `IP`) define quanto tempo a informação de um endereço IP que está associado a um nome permanecerá armazenado no *cache* do resolvedor DNS. Por exemplo, a entrada da tupla (`evil`, `900`, `IN`, `A`, `192.168.0.8`) corresponde ao nome de domínio totalmente qualificado (FQDN) `evil.example.com` que pode ser acessado através do endereço IP `192.168.0.8`. Essa informação será modificada após o período de 900 segundos (15 minutos).

Para entender como cada técnica de *IP-Flux* funciona, a Figura 2.6 ilustra as características do *single-flux* durante o processo de comunicação inicial entre a vítima e as máquinas que respondem pelo domínio solicitado.

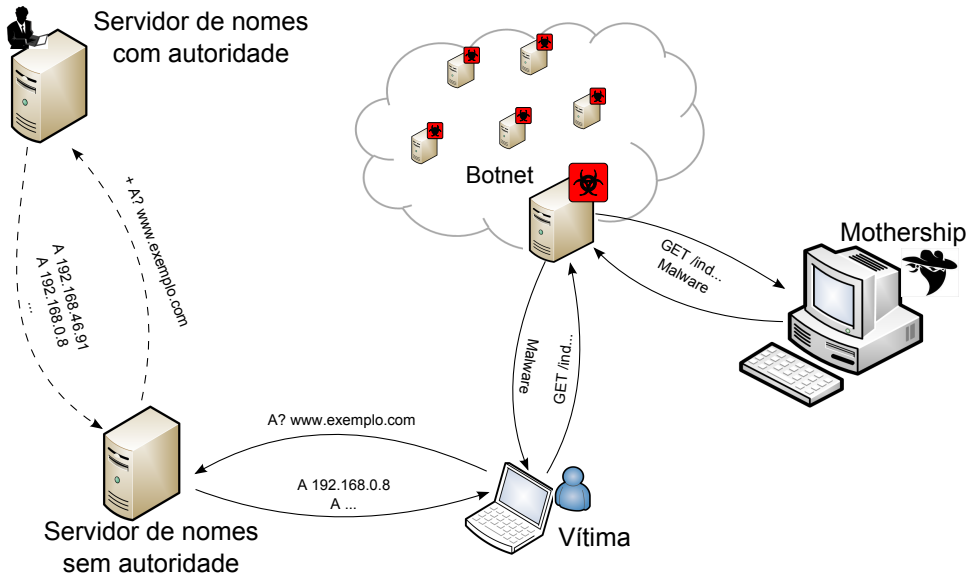


Figura 2.6: Exemplo de Rede de Serviço de Fluxo Rápido (Fonte: Próprio Autor).

Antes de acessar à página maliciosamente forjada pelo *botmaster*, a vítima faz uma consulta DNS para o endereço do servidor que hospeda tal conteúdo. Por exemplo, a vítima consulta o domínio `www.exemplo.com` e obtém como resposta um endereço IP `192.168.0.8` que está sob controle do operador da rede maliciosa. Devido ao tempo de vida consideravelmente baixo, os endereços IPs são trocados com frequência, forçando o resolvedor do cliente à realizar novas consultas DNS. Caso algum endereço IP que responde pelo domínio malicioso não esteja disponível, uma coleção de máquinas infectadas ainda podem atender as solicitações de possíveis vítimas.

Através do recurso de redirecionamento (*proxy*) de solicitações e respostas, a vítima passa a se comunicar com máquinas infectadas que operam como intermediadores do servidor central (*mothership*). Do ponto de vista da vítima, a comunicação foi estabelecida diretamente com servidor que hospeda os dados, no entanto, todos os dados são repassados para os *bots*. Essa arquitetura garante disponibilidade e protege a verdadeira localização do servidor que hospeda o conteúdo malicioso.

A interrupção ou mitigação das redes *fast-flux* são desafiadoras porque operadores das *botnets* tentam imitar o comportamento das redes que oferecem o serviço de `CDNs` e `RRDNS`. Por exemplo, as Figuras 2.7 e 2.8 ilustram a distribuição geográfica do domínio legítimo `debian.org` e malicioso `637login.com`[8].

Em ambos os casos, existem similaridades quanto a distribuição geográfica de endereços IPs, total de endereços únicos, total de sistemas autônomos (`ASN`) e o total

[8]O domínio 637login.com foi extraído da base de dados disponível em `http://bit.ly/1jRzdnO`

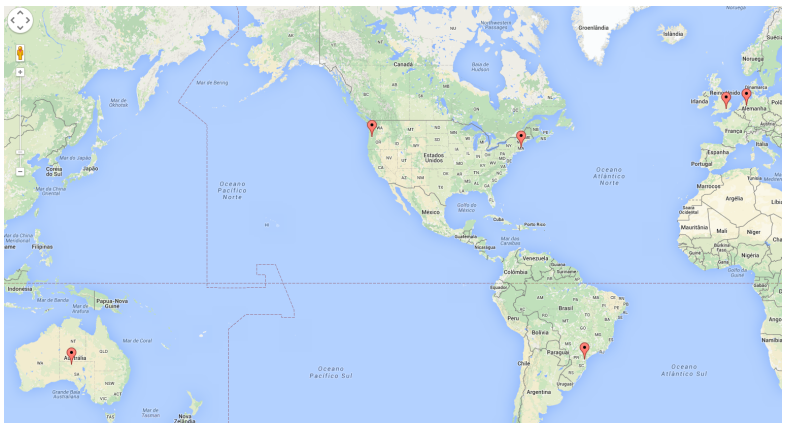


Figura 2.7: Distribuição geográfica do site legítimo `debian.org` (Fonte: Próprio Autor).

de países. Para efeitos comparativos, a Tabela 2.1 apresenta as diferenças entre as características dos domínios em questão. Os resultados indicados na característica `TTL` mostram que um domínio legítimo também pode assumir valores baixos. Diversos trabalhos têm demonstrado que as *botnets* que fazem uso da troca rápida de IP apresentam faixas de valores abaixo de 1800 segundos (Chen et al., 2013a; Hsu et al., 2010; Hu et al., 2009).

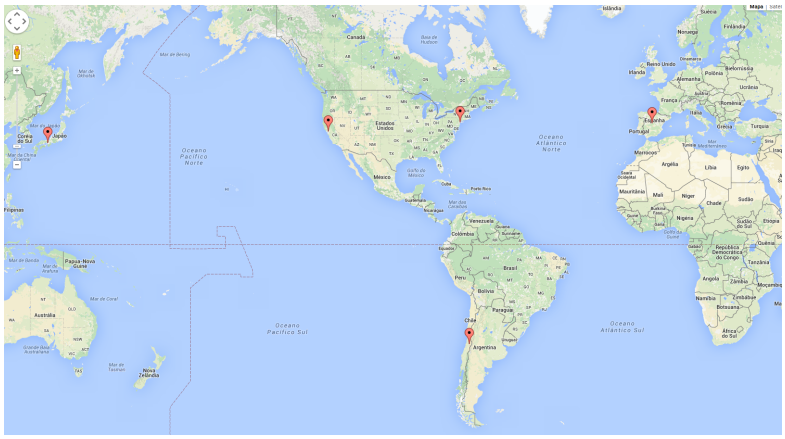


Figura 2.8: Distribuição geográfica do site malicioso `637login.com` (Fonte: Próprio Autor).

Tabela 2.1: Comparação de caracaterísticas entre o domínio legítmo `debian.org` e malicioso `637login.com`

Domínio	TTL	Total IP	Total País	Total ASN
debian.org	58	6	5	6
637login.com	1800	5	4	5

Além de determinar diferentes valores para o TTL, *botmasters* podem manipular outras características do comportamento das *botnets*, os quais podem ser adotados como mecanismos evasivos. Por exemplo, algumas técnicas utilizadas para burlar mecanismos de detecção envolvem manter um número fixo de endereços IPs que são retornados nas consultas DNS (Knysz et al., 2011), evitar sobreposição de endereços IPs durante intervalos de consultas (Hu et al., 2011) e informar um conjunto de endereços IPs que possuam maior disponibilidade (Hu et al., 2009).

Botmasters também conseguem mudar as informações dos servidores DNS que possuem autoridade sob o domínio solicitado. Através da modificação do registro de recurso do tipo NS, é possível definir diferentes endereços IPs que podem responder pela zona de domínio. Tal estratégia é denominada como *double flux*, pois troca tanto os endereços do tipo A (*single flux*) quanto os endereços IPs do tipo NS (Hu et al., 2011). A Figura 2.9 ilustra uma comparação entre as estratégias de *single flux* e *double flux*. No primeiro passo, a vítima é direcionada à consulta do nome de domínio malicioso ff.example.com. Em seguida, uma instância de domínio de primeiro nível (TLD) responde o endereço do servidor que possui autoridade sob a zona de domínio example.com.

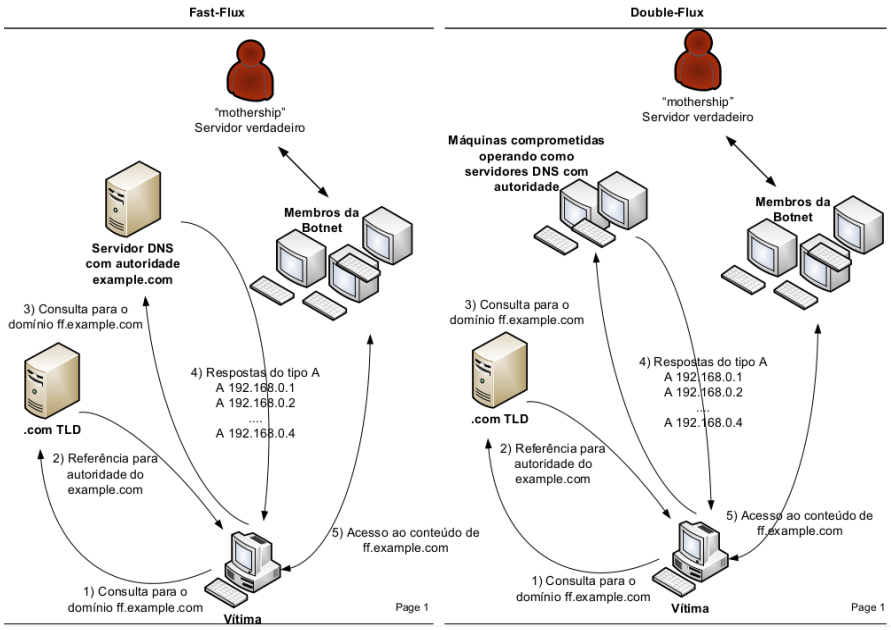


Figura 2.9: Exemplo de Rede de Serviço de Fluxo Rápido (Fonte: Próprio Autor).

No terceiro passo, a vítima pergunta do servidor de nomes qual endereço IP que responde pelo domínio malicioso e obtém como resposta uma lista de endereços IPs. Vale ressaltar que no caso de *double-flux*, a vítima recebe uma lista de endereços de servidores

DNS forjados, que na verdade, são membros da *botnet*. Finalmente, a vítima pode realizar solicitações de conteúdo junto ao conjunto de máquinas infectadas.

Conforme descrito acima, tanto as técnicas de *single-flux* quanto de *double-flux* tratam do comportamento dos hosts que são responsáveis pelo processamento de dados entre a vítima e o servidor que hospeda o conteúdo. Para tentar mitigar esse tipo de abordagem, um operador de rede pode cadastrar esses domínios em uma lista negra e interromper as consultas aos domínios maliciosos (Sinha et al., 2010). No entanto, como contra-ataque, os operadores de *botnets* adotam técnicas que são capazes de gerar nomes de domínios aleatórios para subverter o problema dos nomes de domínios estáticos, como será descrito a seguir.

2.2.2 Troca Rápida de Domínio: Domain-Flux

Domain-flux é efetivamente a operação inversa realizada pelo *IP-flux*, ou seja, refere-se à constante mudança e alocação de múltiplos nomes de domínios a um único endereço IP ou infra-estrutura de C&C. Técnicas aplicáveis à `Domain-flux` incluem o domínio *wildcarding* (Suwa et al., 2012) e, mais recentemente, os algoritmos de geração de domínio, também conhecidos como `DGA` (Antonakakis et al., 2012; Yadav et al., 2012).

Os `DGAs` conseguem produzir nomes de domínios distintos a partir de procedimentos aleatórios que podem tomar como entrada uma semente (Yadav et al., 2012). Para ilustrar como nomes de domínios são gerados, o Algorítimo 1 apresenta um algoritmo que gera nomes a partir das sementes de entrada: ano, mês e dia. Por exemplo, os valores fornecidos como entrada (`2014, 9, 25`) resultam na sequência de caracteres `odnslofgimonbruy`, os quais podem ser utilizados como prefixo em domínios como `odnslofgimonbruy.com.br` e `odnslofgimonbruy.net`. Embora o nome de domínio não seja legível, existem *botnets* capazes de produzir resultados próximos da língua inglesa (Royal, 2008). A partir do uso de `DGAs`, operadores de *botnets* podem migrar as máquinas infectadas para diferentes canais de comando e controle em datas predeterminadas.

Algoritmo 1 Exemplo de algoritmo para geração de nomes de domínios

```
1: genDomain( year, month, day):
2: for a in range(16) do
3:     year ← ((year ⊕ 8 ∗ year) >> 11) ⊕ ((year&0xFFFFFFF0) << 17)
4:     month ← ((month ⊕ 4 ∗ month) >> 25) ⊕ 16 ∗ (month&0xFFFFFFF8)
5:     day ← ((day ⊕ (day << 13)) >> 19) ⊕ ((day&0xFFFFFFFE) << 12)
6:     domain + = chr(((year ⊕ month ⊕ day) % 25) + 97)
7: end for
```

O conceito de *wildcarding* adota um tipo de registro de recurso que irá corresponder a um ou mais subdomínios, se estes subdomínios não possuem os registros de recursos definidos. Esse registro (*) é especificado como o rótulo mais à esquerda do nome de domínio, usando um asterisco, como por exemplo (`*.evil.com`). A interrupção dessa abordagem pode acontecer através do bloqueio do domínio que atende as solicitações (Suwa et al., 2012).

Diferentes estratégias têm sido propostas para detectar domínios de fluxo rápido como engenharia reversa do código malicioso (Stone-Gross et al., 2009), monitoramento de consultas `DNS` que apresentam erro de resposta como *domínio inexistente* (`NXDomain`) ou *falha no servidor* (`SRVFail`) (Haq et al., 2014; Shin e Gu, 2010a; Yadav et al., 2012), análise de distribuição de caracteres e tamanho do nome gerado (Mowbray e Hagen, 2014) e reputação de domínio por histórico (Antonakakis et al., 2012).

A detecção de redes maliciosas de domínio de fluxo rápido fica ainda mais complexa quando os operadores de tais redes combinam estratégias como *fast-flux* e *domain-flux* na mesma arquitetura da *botnet*. Por exemplo, o nome de domínio a ser gerado pelo algoritmo é atendido pelo conjunto de máquinas infectadas que operam como proxy (Yadav et al., 2012). Desta forma, o *botmaster* consegue evadir listas negras de bloqueio de domínios assim como manter o verdadeiro servidor malicioso escondido atrás do proxy de máquinas infectadas. As redes maliciosas que combinam as características de *fast-flux* e *domain-flux* são descritas neste trabalho como *botnets baseadas em fluxo rápido*.

2.3 Botnet: Um Problema de Otimização Combinatória

A detecção de *botnets* baseadas em fluxo rápido a partir do monitoramento de um conjunto específico de características de rede já se mostrou ineficiente, pois o operador da rede maliciosa é capaz de mudar o comportamento da *botnet* em situações evasivas ou caso a rede esteja sendo atacada (Knysz et al., 2011).

Para identificar mudanças de comportamento nas *botnets* de fluxo rápido, alguns trabalhos tentam utilizar Algoritmos Genéticos (Lin et al., 2013) para identificar os melhores pesos que (*weight*) que possam ser aplicados na função linear proposta inicialmente por Holz et al. (Holz et al., 2008).

Outras abordagens, no entanto, buscam acrescentar novas características de rede que possam representar o comportamento dessas redes maliciosas. Esses atributos podem ser coletados a partir de diferentes camadas de rede e metodologias. Por exemplo, na camada de aplicação do protocolo DNS são coletados o `TTL`, a quantidade de endereços IPs que respondem por um domínio (Wu et al., 2010) e os registros de recursos utilizados (Barbosa

et al., 2015); na camada IP, são observados a quantidade de redes distintas correspondentes aos endereços IPs do domínio e o número de sistemas autônomos (Caglayan et al., 2009a). A combinação de diferentes camadas permite estabelecer novas características de comportamento como o total de endereços IPs que um nome de domínio pode assumir dentro de uma janela de monitoramento (Caglayan et al., 2009a) ou se um endereço IP é observado respondendo por múltiplos domínios (Bilge et al., 2014).

Outros trabalhos utilizam estratégias que dependem da interação entre o sistema coletor de características e o domínio suspeito (Caglayan et al., 2009a; Knysz et al., 2011), que podem ser coletados a partir de base de dados de terceiros e que possuem restrições de uso (Stalmans et al., 2012), ou através do monitoramento contínuo do protocolo DNS e reputação do nome de domínio (Bilge et al., 2014; Yadav et al., 2012).

Vale notar que na proposta inicial de detecção de *botnets* baseadas em IP de fluxo rápido (Holz et al., 2008), apenas 3 atributos de redes eram necessários para identificar características de *single-flux* ou *double-flux*. Em trabalhos mais recentes (Bilge et al., 2014), essa quantidade aumentou 5 vezes, totalizando 15 novos atributos, ou 17 caso os atributos da proposta inicial também sejam incluídos. Na detecção de domínios de fluxo rápido, o total de atributos pode ser maior, pois as abordagens incluem tanto as características de rede, quanto as de análise léxica do nome, como citado anteriormente.

O problema em adotar inúmeros atributos para detecção de *botnets* está relacionado à quantidade de tempo necessário para obtenção dessas características, ao tratamento de dados e à duração total para que uma rede maliciosa seja identificada. Uma forma de resolver esse problema é encontrar dentro de um conjunto de atributos, àqueles que conseguem representar melhor a instância de um problema.

Na literatura, técnicas de seleção de atributos estão associadas aos problemas de otimização combinatória, pois dentro de um espaço de soluções possíveis, deseja-se encontrar um subconjunto capaz de minimizar ou maximizar a solução de um problema sabendo que restrições que envolvem esse tipo de cenário devem ser respeitadas (Cormen et al., 2009). Existem diferentes métodos de seleção de atributos que podem ser utilizados para reduzir atributos redundantes e que trazem ruídos durante a classificação, como algoritmos genéticos (Ge e Hu, 2014) e técnicas de ranqueamento (Geng et al., 2007),

Nos algoritmos genéticos, o espaço de todos os subconjuntos possíveis de um grupo de atributos é representado de maneira binária, onde cada elemento no conjunto de atributos é considerado como um gene binário e é formado por uma sequência binária de tamanho fixo, a qual representa um subconjunto de possíveis combinações (Ge e Hu, 2014). Cada combinação (mutação) pode ser avaliada através de um método de classificação que avalia a qualidade da combinação de atributos.

Na técnica de ranqueamento, todos os atributos recebem uma pontuação inicial referente à sua importância e outra com base na similaridade entre dois atributos (Geng et al., 2007). Em seguida, uma função multiobjetivo maximiza a pontuação total de importância e minimiza o total de similaridades entre os atributos, resultando nas melhores características para classificação.

Entretanto, mesmo identificando as características mais apropriadas para uma instância do problema, ainda é preciso observar algumas questões como a quantidade mínima de atributos em relação ao nível de acerto (acurácia) e o tempo necessário para coleta desses atributos. As restrições de seleção de atributos ficam mais evidentes quando existem outros fatores que devem ser considerados para classificação desse tipo de rede maliciosa. Por exemplo, diversos autores apresentam altas taxas de detecção quando um atributo é observado por mais de 24 horas (Caglayan et al., 2009a; Futai et al., 2013; Stevanovic et al., 2016; Yadav et al., 2012). Nesses casos, os resultados obtidos refletem que a detecção acurada está relacionada com o tempo de monitoramento do tráfego `DNS`. Por outro lado, o objetivo desde trabalho é identificar essas redes no menor tempo possível sem afetar o nível de acerto de detecção.

Em um modelo hipotético, se o tempo total do processo de extração de dados até a classificação de um domínio suspeito for desconsiderado, então os problemas de *botnets* podem ser resolvidos a partir de um número considerável de características que são obtidas dentro de uma janela de monitoramento compatível com tempo de coleta. Por outro lado, tal modelo hipotético não pode ser aplicado no mundo real pois durante ataques de rede, disseminação de código malicioso e recebimento de *spam* em massa, o objetivo é identificar e interromper tais comportamentos no menor tempo possível para que os serviços de rede não sejam paralisados ou novos hosts infectados.

Vale notar que mesmo que seja possível identificar um conjunto relevante de características que contribuam com os métodos de detecção, os operadores de rede maliciosa estão buscando novas alternativas para burlar esses mecanismos assim como seu conjunto de características para manter as *botnets* em operação. Do outro lado, tanto na academia como na indústria, novas abordagens de deteção e características são incluídas no processo de classificação para reveter esses ataques. Neste trabalho, essa caça ao rato é definida como ciclo infinito de busca de métodos de detecção, como ilustra a Figura 2.10.

Neste trabalho, a Otimização por Colônia de Formigas é escolhida por ter sido aplicada em diversos trabalhos que envolvem problemas em redes de computadores (Dorigo e Stützle, 2004; Gajpal e Abad, 2009), detecção de anomalias de redes (Ranga e Mandhar, 2018) e detecção hosts infectados na rede (Wu et al., 2018), assim como em problemas com múlti-objetivo (Mokhtari e Ghezavati, 2018). Na literatura, a adoção do conceitos

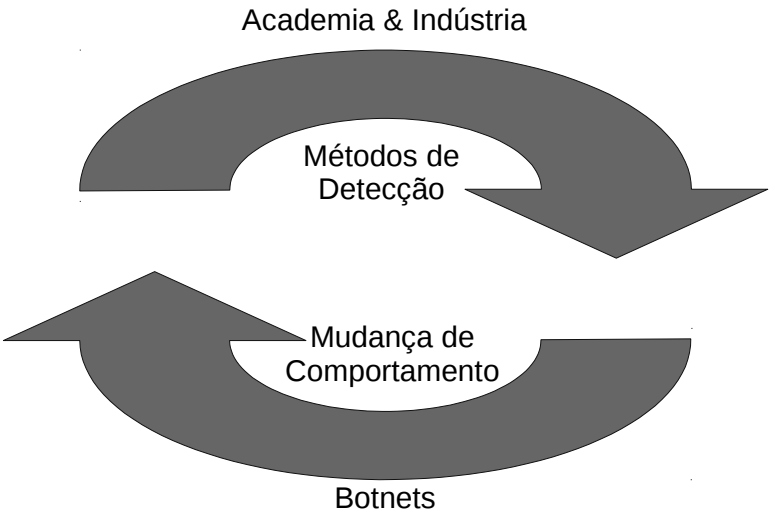


Figura 2.10: Ciclo infinito de busca entre academia e indústria vs operadores de rede maliciosas (Fonte: Próprio Autor).

de colônias de formigas para detecção de *botnets* ainda é inicial e, os trabalhos estão direcionados na identificação dos endereços IPs durante ataques de negação de serviço (`IP traceback`) (Wang et al., 2011, 2010) e no agrupamento de comportamentos semelhantes (Shelokar et al., 2004). No entanto, até onde vai o conhecimento deste autor, nenhum trabalho utiliza a Otimização por Colônia de Formigas para detectar *botnets* baseadas em Redes de Serviço de Fluxo Rápido.

2.4 Otimização por Colônia de Formigas

O comportamento de formigas vem sendo estudado por cientistas da área de computação para ajudar na criação de mecanismos de controle de sistemas multiagentes. Isso acontece devido às propriedades associadas ao comportamento das formigas como operação em grupo, auto-organização, robustez e flexibilidade. Essas características são fundamentais para os sistemas artificiais, otimização e controle ou execução (Bonabeau et al., 1999).

Além disso, para que as formigas consigam se organizar, duas características são utilizadas nesse processo: *estigmergia* (do inglês *stigmergy*) e *feromônio* (Dorigo et al., 2000). Estigmergia é um termo biológico que descreve um modelo de comunicação através do meio ambiente (Dipple, 2011), enquanto o feromônio é o composto químico que permite que tal modelo seja concretizado.

Antes de aplicar os conceitos de formigas num problema de otimização combinatória, é preciso entender que tipo de resultado deseja-se encontrar; soluções que envolvem *maximização* ou *minimização* (Dorigo e Stützle, 2004). Os problemas de minimização têm como objetivo encontrar uma *solução ótima* com um mínimo de custo (Cormen et al., 2009), enquanto problemas de maximação buscam soluções com máximo de ganho (Talbi, 2009). Por exemplo, qual o menor custo em um circuito Hamiltoniano em um grafo ponderado, ou ainda, qual seria a quantidade máxima de tarefas que um trabalhador poderia executar considerando suas limitações como total de horas trabalhadas, esforço e custo?

Vale notar que para cada problema a ser tratado existe um conjunto de *restrições* que devem ser respeitadas durante a busca da solução. Por exemplo, no Problema do Caixeiro Viajante, apenas uma cidade deve ser visitada por vez, enquanto que na produção de um material, é preciso levar em consideração o segmento específico de mercado que pode comprar tal produto.

Um problema de otimização, formalmente, apresenta características: *i*) que consideram um conjunto de soluções candidatas $\mathcal{S}$; *ii*) uma função objetivo f; *iii*) e um conjunto de restrições Ω. As soluções candidatas são as variáveis de decisão do problema que são avaliadas através da função objetivo f, a qual pode maximizar ou minimizar os valores observando as restrições do problema (Luenberger e Ye, 2008).

A partir da combinação das características das formigas e de problemas de otimização, foi proposto um conceito denominado como `ACO` - *Metaheurística de Otimização por Colônia de Formigas*, do inglês (*Ant Colony Optimization Metaheuristic*) (Dorigo et al., 1996).

Metaheurística é um conjunto de conceitos que definem métodos aplicáveis a diferentes problemas e pode permitir que subsoluções (*heurísticas*) sejam combinadas em um *espaço de busca* para encontrar uma solução próxima da ótima (Blum e Roli, 2003; Dorigo e Stützle, 2004). Tal estratégia pode ser aplicada na resolução de problemas de otimização combinatória como *Caixeiro Viajante* e *Problema de Agendamento* (Dorigo et al., 2000).

O algoritmo *ANT System* (`AS`) (Dorigo et al., 1996) foi a primeira abordagem baseada nos comportamentos das formigas para encontrar o caminho mais curto no Problema do Caixeiro Viajante. Esse algoritmo utiliza o retorno positivo (*positive feedback*) para encontrar boas soluções, computação distribuída para evitar convergência prematura, e heurística gulosa para encontrar soluções aceitáveis no início do processo.

A ideia principal é utilizar movimentos probabilísticos (*aleatórios*) das formigas para que múltiplas rotas sejam percorridas e, àquelas que apresentarem o menor caminho sejam marcadas com feromônio para que outras formigas também comecem a utilizar essa

nova alternativa. O processo de escolha de um novo caminho e a marcação de feromônio acabam quando as formigas convergirem para uma única rota (Dorigo et al., 2000; Dorigo e Stützle, 2004).

A metaheurística proposta por Dorigo e Stützle é um método de busca de propósito geral que permite usar busca local ou construtiva para encontrar uma solução próxima à ótima de uma determinada instância. Na busca construtiva, novas soluções são criadas a partir da adição de componentes até que a solução esteja pronta, porém a solução pode apresentar resultado inferior em comparação aos dados obtidos na busca local.

A busca local parte de algum ponto inicial (*solução atual*) e tenta melhorar tal solução a partir da busca na vizinhança por um resultado melhor, o qual é utilizado como solução atual na iteração seguinte (García-Martínez e Lozano, 2008). A busca local é encerrada quando não há outra solução na vizinhança que possa substituir ou melhorar a solução atual.

Para entender o funcionamento do conceito de formigas na aplicação de problemas de otimização combinatória, o Algoritmo 2 ilustra os passos que são realizados pelo *ANT System*: *Construção das Soluções*, *Busca Local* e *Atualização de Feforomônio*. No `AS`, o método de busca local é opcional e pode ser usado para melhorar as soluções encontradas pelas formigas.

Algoritmo 2 ACO para problemas de otimização combinatórias

```
1: Inicialização de parâmetros
2: while Condições não forem verdadeiras do
3:    ConstruçãoDasSoluções()
4:    BuscaLocal()
5:    AtualizaçãoFeromônio()
6: end while
```

Na *Construção das Soluções*, um conjunto de m formigas constroem soluções para a instância do problema a partir de um estado inicial de soluções parciais $s_p = \emptyset$. As soluções parciais são definidas a partir valores que fazem parte do conjunto de todas as soluções componentes C. Cada solução componente c_i^j, tal que $i = 1, \ldots, n$, pode assumir um possível valor j para esse componente e, para que a solução parcial s_p assuma c_i^j, o feromônio τ_{ij} deve indicar a atratividade de tal componente. A solução parcial atual s_p é substituída quando outra solução viável $c_i^j \in \mathcal{N}(s_p) \subseteq C$ existir. Vale notar que o conjunto $\mathcal{N}(s_p)$ representa as possíveis soluções componentes que podem ser adicionadas às soluções parciais respeitando as restrições do problema (Dorigo et al., 1996; Dorigo e Stützle, 2010).

Na proposta original do algoritmo *Ant System*, a atualização da solução parcial é definida através da Equação 2.1, onde $\eta(c_i^j)$ denota uma função que calcula viabilidade de cada solução componente c_i^j, essa função também é conhecida em outros trabalhos como heurística ou informação heurística (Dorigo et al., 2000; Dorigo e Gambardella, 1997). Os parâmetros α e β determinam a influência do feromônio ou da heurística na decisão probabilística, respectivamente.

$$P(c_i^j|s_p) = \frac{[\tau_{ij}]^\alpha [\eta(c_i^j)]^\beta}{\sum_{c_i^j \in \mathcal{N}(s_p)} [\tau_{ij}]^\alpha [\eta(c_i^j)]^\beta}, \quad \forall c_i^j \in \mathcal{N}(s_p) \tag{2.1}$$

O procedimento de *Busca Local* é opcional e pode ser utilizado para melhorar uma solução parcial encontrada. Por exemplo, no caso do caixeiro viajante, uma estratégia é aplicar o algoritmo de duas trocas (*2 exchange*) para que um caminho seja substituído e avaliado por até duas rotas alternativas (Martens et al., 2011).

Na *Atualização de Feromônio*, para que as formigas continuem escolhendo melhores caminhos, é preciso diminuir (evaporação) a quantidade de feromônio daqueles que tiverem sidos marcados eventualmente e que possuem pouca atratividade. A evaporação de feromônio é necessária para evitar convergência rápida do algoritmo para uma região ótima local, do inglês *local optima* (Sundaram, 1996). Local *optima* são pontos ótimos em algumas partes da vizinhança, mas não necessariamente para todos componentes do conjunto de soluções (*global optima*).

A Equação 2.2 descreve a taxa de atualização de feromônio, onde $\rho \in (0, 1]$ reflete o comportamento da evaporação, S_{upd} é o conjunto de todos os caminhos (soluções) e $g(s)$ é uma função objetiva que avalia a qualidade de $s \in S_{upd}$.

$$\tau_{ij} = (1 - \rho).\tau_{ij} + \sum_{s \in S_{upd} | c_i^j \in s} g(s) \tag{2.2}$$

Em termos práticos, a adoção do conceito por colônia de formigas deve considerar que o algoritmo *i*) utiliza cada caminho escolhido pelas formigas como uma possível solução para o problema. Uma heurística η avalia tal solução antes que seja adicionada ao conjunto final de soluções; *ii*) A quantidade de feromônio depositada (intensidade) é equivalente à qualidade do candidato a solução. O controle do montante de feromônio é dado pela função τ que modifica informações sobre o caminho na proporção que outras formigas decidem por tal rota; Finalmente, *iii*) a decisão para escolha entre dois ou mais caminhos é baseada na probabilidade do total de feromônio depositado, isto é, a partir dos valores em η e τ.

Para entender o conceito descrito acima, considere o Problema do Caixeiro Viajante, onde a formiga localizada na cidade$_i$ deve tomar uma decisão de qual cidade vizinha (k, j, g) será visitada, como ilustra a Figura 2.11. Vale ressaltar que para tal deslocamento, é necessário *considerar a restrição* que a próxima cidade a ser visitada ainda não faz parte do conjunto de cidades conhecidas.

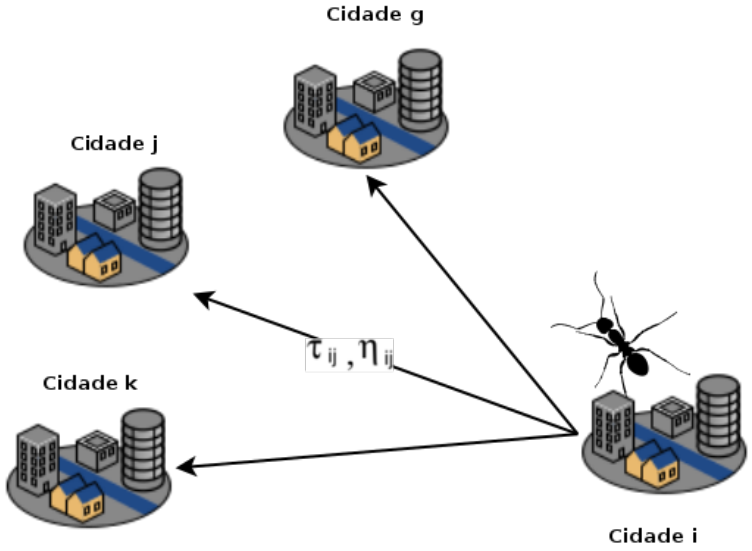


Figura 2.11: Tomada de decisão do menor caminho observando os valores em η e τ (Dorigo e Stützle, 2004).

Além disso, é preciso ainda observar outras restrições durante a resolução Caixeiro Viajante, incluindo a atualização da quantidade de feromônio tanto na ida quanto na volta de uma caminhada, a quantidade de formigas em relação ao número de cidades e questões de desempenho para instâncias muito grandes e assim como a distância total percorrida pelo caixeiro viajante

Por exemplo, seja o grafo ponderado $G = (V, E)$, onde V é o conjunto de cidades a serem visitadas e E o conjunto de arestas que denotam as distâncias entre cidades. A distância a ser percorrida entre as cidades c_i e c_j pode ser denotada por d_{ij}, tal que $(i, j) \in E$. O objetivo do algoritmo é escolher o menor caminho entre as cidades, as formigas podem avaliar o custo de tal decisão através da heurística $\eta_{ij} = \frac{1}{d_{ij}}$, além da quantidade de feromônio depositado τ_{ij}.

Considerando os passos descritos no Algoritmo 2, os parâmetros iniciais definem como o algoritmo deve se comportar, incluindo o número total de formigas, quantidade de feromônio e total de interações (*tours*) que as formigas devem executar até que a solução seja encontrada.

Na *Construção das Soluções*, as formigas são distribuídas aleatoriamente nas cidades, representadas por vértices do grafo. Seja k, tal que $k \in \{\texttt{conjunto inicial de formigas}\}$ deve decidir uma direção a ser seguida, sair da cidade i para j. A Equação 2.3 descreve como as formigas devem caminhar nas cidades e atualiza as definições iniciais propostas pelo algoritmo do *ANT System* (Dorigo e Stützle, 2004). Seja $\mathcal{N}_i^k$ o conjunto de vizinhos (i) da formiga k que ainda não foram visitados, isto é, cidades que podem ser acessadas a partir de i. Os parâmetros α e β são utilizados para controlar a influência do feromônio (τ) e heurística (η); adicionar as cidades mais próximas ou evitar a estagnação da solução.

$$P_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{ij}]^\alpha [\eta_{ij}]^\beta} \tag{2.3}$$

Para efeitos demonstrativos, considere na Figura 2.12 um caminho hipotético a ser escolhido pelo algoritmo de formiga. Considere os parâmetros iniciais de $\alpha = 1$, $\beta = 1$, $\tau = 0.1, m = 5$ e $\eta = \frac{1}{d_{ij}}$. Vale notar que a quantidade de formigas deve ser igual ou maior ao número de cidades (Dorigo et al., 1996), permitindo que todos os caminhos sejam explorados pelo algoritmo. Os valores de α e β são diferentes de zero para que as formigas não sigam exclusivamente o feromônio ou a heurística, respectivamente.

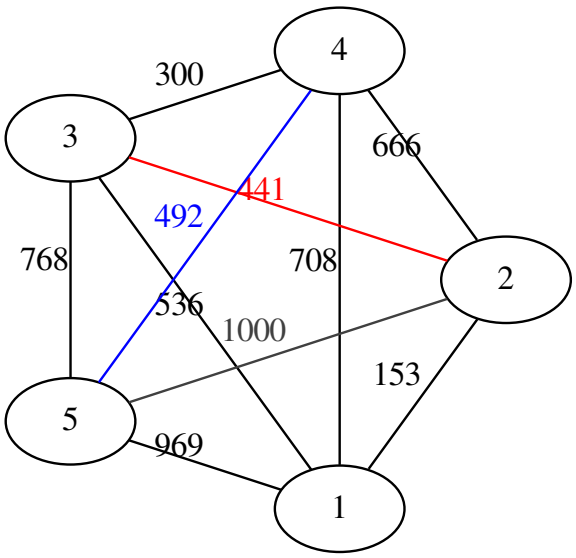


Figura 2.12: Exemplo do Caixeiro Viajante contendo cinco cidades que devem ser visitadas pelo algoritmo de formigas no menor custo (Fonte: Próprio Autor).

No processo inicial, são adicionadas 5 formigas, cada uma é distribuída aleatoriamente nas cidades a serem visitadas. Na *primeira* iteração para a formiga **1**, assumindo que o

ponto de partida seja a cidade número 1, a formiga tem a probabilidade de 15.07%, 4.30%, 3.26% ou 2.38% para ir para as cidades 2, 3, 4 e 5, respectivamente. Na *primeira* iteração para a formiga **2**, localizada na cidade 2, a probabilidade de escolha é 14.45%, 5.01%, 3.32% ou 2.21% para as cidades 1, 3, 4 e 5, respectivamente. Na *primeira* iteração para a formiga **3**, as probabilidades são 5.32%, 6.46%, 9.50% e 3.71% para as cidades 1, 2, 4 e 5, respectivamente. Para a formiga de número **4**, as probabilidades são 4.26%, 4.53%, 10.06%, 6.14% para as cidades 1, 2, 3 e 5, respectivamente. Finalmente, para a formiga **5**, as probabilidades são 4.81%, 4.66%, 6.07%, 9.47% para as cidades 1, 2, 3 e 4, respectivamente.

Ao final da primeira iteração, os caminhos escolhidos por cada formiga são $1 \rightarrow 2$; $2 \rightarrow 1$; $3 \rightarrow 4$; $4 \rightarrow 3$ e $5 \rightarrow 4$. Na *segunda* iteração para a formiga **1**, o ponto de partida é a cidade 2 e as probabilidades são 15.85%, 10.49% e 6.99% para as cidades 3, 4 e 5, respectivamente. Vale ressaltar que a formiga **1** já visitou a cidade 1, portanto cabe à formiga escolher outras rotas. Quando cada formiga terminar seu processo de escolha durante a *segunda* iteração, os melhores caminhos escolhidos serão: $1 \rightarrow 2 \rightarrow 3$; $2 \rightarrow 1 \rightarrow 3$; $3 \rightarrow 4 \rightarrow 5$; $4 \rightarrow 3 \rightarrow 2$ e $5 \rightarrow 4 \rightarrow 3$.

O processo de iteração termina quando as formigas realizarem N iterações, isto é, visitarem todas as cidades do grafo. O menor caminho escolhido foi encontrado pela formiga **2** denotando o caminho $2 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 2$ com 1634 KM. A formiga **1** fez o caminho de 2304KM, a formiga **3** 2354KM e as formigas **4** e **5** fizeram 2355 KM, respectivamente.

As Equações 2.4 e 2.5 descrevem, de uma maneira mais clara, como a definição inicial da Equação 2.2 pode ser desmembrada e aplicada na fase de `AtualizaçãoFeromônio` (Dorigo e Stützle, 2004). Primeiro, é realizado o procedimento de dissipação da quantidade de feromônio depositada pelas formigas com uma taxa de evaporação ρ, tal que $0 < \rho \leq 1$, conforme ilustra a Equação 2.4.

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij}, \quad \forall (i, j) \in E \tag{2.4}$$

Em seguida, é feito o depósito de feromônio em todas as arestas que foram cruzadas pelas formigas, a partir da Equação 2.5.

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \quad \forall (i, j) \in E \tag{2.5}$$

Onde $\Delta\tau_{ij}^{k}$ é o total de feromônio que a formiga k deposita na aresta visitada e pode ser definida pela seguinte função:

$$\Delta\tau_{ij}^{k} = \begin{cases} 1/C^{k}, \text{ se aresta } (i,j) \text{ pertence a } T^{k}; \\ 0, \text{ caso contrário;} \end{cases} \tag{2.6}$$

Seja C^k o tamanho total do caminho T^k realizado pela $k-$ézima formiga. A atualização de feromônio é dada pela soma dos tamanhos das arestas que pertencem a T^k, resultando na melhor rota aquela que possui mais feromônio.

Após a evaporação do feromônio, conforme Equação 2.5, cada formiga deve atualizar seu caminho. Por exemplo, a formiga **1** deixar um registro de $\frac{1}{2304} = 0,00043403$ se a aresta (i,j) pertencente ao caminho cruzado. No caso da formiga **2**, a quantidade de feromônio a ser deixada é de $0,00067522$, enquanto as formigas **3**, **4** e **5** depositarão $0,00042481$, $0,00042463$ e $0,00042463$, respectivamente. A partir desses resultados é possível dizer que a distância é inversamente proporcional a atratividade de um caminho, como pode ser observado no caso da formiga **2**, onde a distância é menor, porém a quantidade de feromônio depositada é maior em relação aos T^k das outras formigas.

Novas iterações podem ocorrer caso as formigas não estejam *convergindo* para o mesmo caminho, isto é, outros caminhos são desconsiderados e a maioria das formigas segue apenas uma rota para encontrar o alimento.

O procedimento de `BuscaLocal` é utilizado para evitar que as formigas fiquem estagnadas e busquem pelo melhor caminho entre seus vizinhos (Dorigo et al., 2000). Os trabalhos de Dorigo et al. (Dorigo et al., 2000; Dorigo e Stützle, 2004) descrevem com mais detalhes como os algoritmos de formigas são utilizados para resolver o problema do caixeiro viajante.

Para utilizar o algoritmo por colônia de formigas na resolução de *botnets* baseadas em fluxo rápido, é preciso identificar os atributos que possam descrever os comportamentos de tais redes. A ideia principal é modelar as características das redes de fluxo rápido para que os agentes identifiquem os melhores caminhos (atributos de rede) que levem a fonte de alimento, isto é, um domínio malicioso. O Capítulo 3 a seguir apresenta os principais atributos de rede que são utilizados em diversos trabalhos para detecção de `FFSN`.

3

Trabalhos Relacionados

O sistema de nomes de domínio (*Domain Name System* - DNS) (Mockapetris, 1987) é utilizado por muitas aplicações como navegadores Internet, aplicações de correio eletrônico e softwares de mensagens instantâneas. Este serviço associa nomes simbólicos aos respectivos endereços IP numéricos, permitindo aos usuários utilizar, com maior comodidade, recursos compartilhados em um ambiente de rede conectado.

Devido à importância do tráfego DNS para aplicações de rede, atacantes fazem uso das consultas como um dos principais pontos de partida para possíveis ataques de rede. As vulnerabilidades do DNS podem ser divididas em duas categorias: ataques que exploram falhas de segurança no serviço de tradução de nomes e ataques de rede que utilizam o tráfego DNS como ponto de partida para outros ataques de rede.

Na primeira situação, atacantes subvertem o funcionamento do protocolo para envenenar o resolvedor de um cliente ou servidor de nomes a fim de direcioná-lo a um site comprometido (Musashi et al., 2011). Na segunda categoria, atacantes utilizam o tráfego DNS para identificar possíveis alvos em redes remotas, encontrar servidores de correio eletrônico abertos, encontrar o endereço do servidor do canal de comando e controle durante a fase de *rallying* ou para manter o funcionamento das *botnets* a partir de técnicas de troca rápida de informação, também conhecido como *Fast-Flux*.

A seguir será apresentada uma revisão da literatura de trabalhos que utilizam o tráfego DNS para identificar *botnets* baseadas em redes de serviço de fluxo rápido e trabalhos que utilizam técnicas de Otimização Combinatória para resolver problemas de detecção de *botnets*.

Para melhor entendimento, a descrição a seguir apresenta os trabalhos na sua ordem cronológica de publicação. O objetivo é demonstrar o processo evolutivo das *botnets* e

ilustrar como os diferentes métodos de detecção utilizam as características de rede para identificar essas anomalias. Até onde vai o conhecimento desse autor, este é o primeiro trabalho a utilizar Colônias de Formigas para detectar *botnets* baseadas em redes de serviço de fluxo rápido.

3.1 Redes de serviço de fluxo rápido - FFSN

A detecção de redes de serviço de fluxo rápido pode ser realizada utilizando métodos ativos, passivos ou híbrido. Na detecção ativa, as abordagens enviam solicitações utilizando diferentes protocolos de rede para obterem informações sobre nome de domínio investigado. No caso da detecção passiva, o tráfego de rede é monitorado através de sensores locais ou distribuídos globalmente para identificar as características das máquinas que consultam um domínio suspeito. Finalmente, o método híbrido combina ambas estratégias na resolução do problema de rede de serviço de fluxo rápido.

Vale ressaltar que cada método possui características que podem favorecer ou prejudicar o processo de identificação desse tipo de rede. Por exemplo, na análise ativa, o operador da rede maliciosa pode identificar padrões de consultas forjadas com objetivo de extrair características do domínio. Por outro lado, no modo passivo, é possível identificar, de maneira furtiva, características relevantes que permitem detectar redes de serviço de fluxo rápido, considerando que nesse método de análise, o volume de tráfego deve conter informações que facilitem esse processo.

As seções a seguir descrevem diferentes abordagens de métodos passivo e ativo que foram utilizados na detecção de FFSN.

3.1.1 Detecção Ativa

No ano de 2008, os primeiros trabalhos estavam direcionados a entender com mais profundidade o funcionamento das redes de serviço de fluxo rápido. Por exemplo, o primeiro trabalho dedicado à identificação e caracterização do tráfego de redes de serviço de fluxo rápido, incluindo as principais diferenças entre as redes *single-flux* e *double-flux*, foi apresentado por Salusky e Danford (Salusky e Danford, 2008). Os autores extraem e avaliam características de nomes de domínios para determinar se um domínio está hospedado numa rede FFSN. As características observadas consideram o tempo de vida (`TTL`) baixo, número de endereços IP encontrados no registro de recurso do tipo `A`, número de endereços IP no registro de recurso do tipo `NS`, diversidade de redes não relacionadas e se existem endereços IP associados à redes domésticas. Os resultados mostram que domínios

legítimos possuem número estável de endereços IP associados ao tipo de registro do tipo `A`, enquanto um domínio de fluxo rápido possui um número maior de endereços IP distintos de máquinas infectadas.

O trabalho de Holz et. al. (Holz et al., 2008) mostraram que a utilização do `TTL`, apesar de ser um indicador relevante em redes de fluxo rápido, não é um fator definitivo na detecção de redes dessa natureza. Para tentar mitigar essa anomalia, os autores apresentaram uma heurística que utiliza um conjunto de recursos de rede como o número de endereços IP dos registros de recurso do `A` e `NS`, a quantidade de números de sistemas autônomos (`ASN`) por domínio, os valores do `TTL` e a localização geográfica do IP para distinguir entre tráfego FFSN, `CDN` e `RRDNS`.

Para distinguir entre redes `CDN` e `FFSN`, os autores definiram uma razão, denominada como *fluxiness*, que leva em conta a quantidade total de endereços IP encontrados no registro `A`, pela quantidade de endereços retornados em uma única consulta do tipo `A`. O objetivo de tal métrica é encontrar nomes de domínios que possuem um número elevado de endereços IP associados após a primeira observação. Entre os possíveis resultados, um domínio é marcado como legítimo se o número de endereços IP não mudar ao longo do tempo, razão igual a 1. Caso a quantidade de endereços IP seja modificada, o domínio é identificado como redes de serviço de fluxo rápido.

Para classificar corretamente o nome de domínio, são utilizadas características de base de dados de domínios benignos[1] e de domínios maliciosos, a qual foi criada a partir de `URLs` presentes em emails categorizados como *spam*. Os resultados mostram que máquinas controladas por operadores de rede possuem baixa disponibilidade de acesso, pois podem ser desligadas a qualquer momento pelo verdadeiro proprietário do computador. Adicionalmente, foi possível identificar que 30% dos domínios encontrados em spam são hospedados em redes de fluxo rápido.

Nazario e Holz (Nazario e Holz, 2008) também utilizaram os domínios de `URLs` para encontrar redes de fluxo rápido. Para apoiar esse processo, os autores empregaram engenharia reversa de *malware*, listas negras e análise manual dos domínios. Além das características observadas por Holz et al., ainda é necessário extrair informações da zona autorizativa do domínio (`SOA`) e a distribuição dos endereços IP encontrados, resultado em 9 características extraídas.

Os resultados obtidos por Nazario e Holz denotam que caso um domínio seja confirmado como suspeito, os endereços IP associados a este domínio são adicionados em uma base de dados que, posteriormente será utilizada como referência para reputações de IP. Os resultados mostraram que 76%, do total de 928 endereços de domínios `.com`, são

[1] Alexa.com e Projeto Dmoz

domínios associados às redes de fluxo rápido e que 83% dos endereços IP respondem por múltiplos nomes de domínios, demonstrando que existe uma sobreposição de endereços IP por nomes de domínios. É importante ressaltar que a sobreposição de endereços IP por nome de domínio também acontece em cenários legítimos, onde um único host hospeda diferentes clientes, o que torna a detecção de `FFSN` desafiadora.

Passerini et. al. (Passerini et al., 2008) utilizam o classificador Naïve Bayes para determinar se um nome de domínio é malicioso ou legítimo. Os autores partem do princípio que um domínio malicioso possui um tempo de vida curto, já que o domínio é retirado do ar quando o mesmo é detectado. A detecção utiliza os atributos observados em (Salusky e Danford, 2008), (Holz et al., 2008), (Nazario e Holz, 2008) e apresenta novas características como data de registro, nome do operador de registro domínio (`Registra`), número de redes distintas e número de domínios completamente qualificados (`FQDN`). Vale notar que as características de registro do nome de domínio ajudam identificar grupos de domínios que foram registrados no mesmo provedor no entanto, tais atributos só podem ser obtidos quando não há restrições de acesso, seja por questões de privacidade ou bloqueio geográfico. Os resultados mostram que 7.8% (387) hosts analisados são membros de *botnets* de fluxo rápido e que dois domínios compartilharam 81% dos hosts entre si, confirmando também os resultados obtidos em (Nazario e Holz, 2008).

Diferente de outros trabalhos de detecção, (Feamster et al., 2008) realizam um estudo empírico das principais características associadas às redes de serviço de fluxo rápido como taxa de mudança de endereços IP, distribuição geográfica, taxa de crescimento da rede e sobreposição de endereços IP (compartilhamento) por nome de domínio. Para identificar essas redes, os autores monitoraram por 3 meses diferentes listas negras para controle de campanhas de *spam*. Os resultados mostram que campanhas de *spam* que hospedam conteúdo de produtos fármacos permanecem em operação por mais tempo que sites que disponibilizam download malicioso ou *phishing*, ou seja, sites fármacos operam durante duas semanas e sites de conteúdo malicioso três dias. Além disso, Feamster et al. demonstraram que um endereço IP pode operar em duas funções nesse tipo de rede; um IP pode hospedar conteúdo e ter autoridade pela zona de domínio.

Em 2009 o funcionamento desse tipo de rede maliciosa já era de conhecimento da literatura, então os trabalhos a partir daquele ano buscavam identificar e classificar redes de serviço de fluxo rápido no menor tempo possível. Por exemplo, Caglayan et al. (2009a) classificam redes de fluxo rápido observando as características citadas nos trabalhos anteriores, descritos acima. No entanto, a diferença é que os domínios suspeitos de fluxo rápido são monitorados dentro de uma janela de 10 minutos, a qual é definida como referência para identificar mudanças na rede (inclusões ou exclusões de endereços

IP). Os resultados mostram que diversas *botnets* estão distribuídas geograficamente em diferentes países e redes, incluindo provedores de Internet e Universidades.

Para resolver as limitações em seu trabalho anterior, Caglayan et al. (2009b) apresentam uma arquitetura para identificação em tempo real de FFSN que utiliza diferentes fontes de dados incluindo listas negras e domínios encontrados em *honeypots* distribuídos. Os resultados mostram que 96.6% dos domínios são identificados corretamente no período inferior a dois minutos e que esse percentual aumenta para 100% se o classificador analisar um domínio por mais tempo. As observações apresentadas por Caglayan et. al. demonstram que o tráfego `DNS` como fonte exclusiva para detecção de `FFSN` possui certas limitações, principalmente nos casos onde as *botnets* conseguem simular o tráfego de redes benignas para evadir sensores de rede (Knysz et al., 2011).

Além do tráfego `DNS` como fonte de análise, o tráfego `HTTP` também pode ser investigado para detectar esse tipo de rede maliciosa. O trabalho em Castelluccia et al. (2009) utilizam o tempo de ida e de volta das requisições `HTTP` para determinar se existem servidores de conteúdos escondidos por de trás de endereços IP infectados. Para determinar o comportamento malicioso, os autores enviam solicitações `HTTP` e, a partir dos resultados, é realizado um cálculo para determinar se o tempo de resposta está de acordo com os padrões armazenados em bases de dados conhecidas. É importante observar que o protocolo `HTTP` pode ser combinado em múltiplas camadas com outros protocolos como `P2P` e `DNS` (Sinclair et al., 2009).

Para contornar as limitações quanto ao uso exclusivo do tráfego DNS para classificar redes de serviço de fluxo rápido, diversos trabalhos têm correlacionado múltiplas fontes de dados para determinar se um domínio é malicioso ou legítimo (Antonakakis et al., 2010; Hu et al., 2009; Huang et al., 2010; Wu et al., 2010).

Hu et al. demonstram um *framework*, denominado `RBSeeker`, que emprega fontes de informações externas includindo e-mail categorizados como spam, consultas DNS a servidores recursivos e listas negras, além da análise do fluxo de rede (`NetFlow`). A partir desses dados, os autores monitoram o total de endereços IP associados ao nome de domínio, número de sistemas autônomos (`ASN`) e o nome reverso de um endereço IP (`PTR`), o qual pode ser utilizado para identificar se um endereço IP pertence às redes de provedores de Internet banda larga. Os resultados mostram que a solução proposta identificou 91 mil domínios suspeitos e apontam baixo falso positivo de 0.008%. Quando um domínio benigno é identificado, esse domínio e seus endereços IP são adicionados numa lista branca para que tais endereços não sejam avaliados novamente. Essa abordagem no primeiro momento é valida, no entanto, caso esse domínio seja sequestrado ou as máquinas

associadas sejam comprometidas, então a solução proposta não é capaz de identificar mudanças de comportamento.

Campbell et al. (2011) investigam as principais diferenças entre as redes de distribuição de conteúdo e redes de fluxo rápido. Por exemplo, durante um intervalo de 48 horas, a quantidade de novos endereços IP encontrados em `CDNs` representa 0.1%, enquanto nas `FFSNs`, esse número representa 78.9%. Outra característica observada é que a quantidade de prefixos da classe B dos endereçamentos IP nas sub-redes de `FFSN` é maior que nas `CDNs`, 233 e 109, respectivamente. Tal resultado mostra que a distribuição de IP no primeiro tipo de rede é esparsa, pelo fato que vários hosts ao redor do globo fazem parte da *botnet*, enquanto na segunda o endereçamento é concentrado em uma única região.

Knysz et al. (2011) implementaram um sistema denominado `DIGGER` de monitoramento global do tráfego DNS. O sistema proposto coletou durante 4 meses tráfego DNS para entender os padrões e modelos evolutivos das *botnets* de fluxo rápido. Os resultados mostram que métodos propostos em trabalhos anteriores possuem baixa precisão na detecção de *botnets* mais recentes.

Por exemplo, os resultados de Holz et al. e Hu et al. tiveram uma queda na taxa de acerto de 2 e 24 pontos percentuais, em comparação com os valores obtidos inicialmente 88% e 97%, respectivamente. Além disso, os resultados mostraram que a partir do controle do número de endereços IP pertencentes a diferentes `ASNs` é possível subverter as soluções atuais controlando o total de endereços IP que são retornados em cada consulta DNS. Outro aspecto que permite burlar os sistemas detectores de redes de fluxo rápido são as informações das consultas reversas (`PTR`). Por exemplo, mesmo que palavras como `modem`, `adsl`, `dial-up` sejam fortes indicadores de máquinas comprometidas respondendo pelo domínio malicioso, 50% dos hosts identificados não possuíam um nome reverso.

Apesar dos problemas citados, as soluções de Holz et al. e Hu et al. tendem a melhorar a precisão de detecção quando a janela de monitoramento aumenta. No entanto, algumas *botnets* foram observadas mudando seu comportamento, como a mudança gradual do total de endereços IP a cada consulta DNS, acrescentando desta forma, novos endereços IP de maneira estável.

Lin et al. (2013) apresentam um algoritmo genético que determina a melhor estratégia de detecção em tempo real de redes de fluxo rápido mesmo que o padrão de rede seja modificado. Diferente de outros trabalhos baseados em múltiplas consultas DNS, essa proposta obtém as características do nome de domínio a partir de uma única consulta, reduzindo assim, o tempo necessário para classificação de FFSN. Para classificação são utilizados o total de endereços IP associados ao nome de domínio; número único de ASN; entropia da distribuição de nomes dos roteadores de borda (nome reverso); e o desvio

padrão do tempo de ida e de volta dos pacotes (RTT) que interagem com a *botnet*. Segundo Lin et al., o RTT entre um cliente e uma CDN legítima deve ser menor em comparação com uma rede de fluxo rápido, visto que, hosts de uma CDN estão geograficamente mais próximos da origem da consulta.

Para detectar uma rede de fluxo rápido, os autores apresentam uma versão modificada do método proposto por Holz et. al. (2008). Enquanto os pesos do índice de fluxo em Holz et. al. são fixos, Lin et al. utilizam algoritmos genéticos para encontrar os melhores valores durante a detecção.

Para treinar o algoritmo genético, Lin et al. utilizam uma base de dados contendo domínios legítimos e domínios de FFSN obtidos no alexa.com e listas negras, respectivamente. Os resultados mostram que a entropia obtida em redes FFSN é maior devido à diversidade de distribuição de IP, assim como desvio padrão do RTT para esse tipo de rede é superior em comparação as FFSN. Além disso, em comparação com outros trabalhos, Lin et al. acertam 98.47%, enquanto Holz et al. acertam 93.61% e Huang et al. 63.43% para as bases utilizadas. Por outro lado, apesar de reduzir consideravelmente o tempo de detecção, o trabalho de Lin et al. não consegue identificar os ataques que imitam redes válidas CDN ou que possuem um conjunto de máquina do mesmo sistema autônomo (Knysz et al., 2011).

Al-Duwairi e Al-Hammouri (2014) utilizam o comportamento inerente dos agentes de fluxo (máquinas infectadas) para identificar nomes de domínios que utilizam redes de serviço de fluxo rápido. A proposta, denominada como FF-Watch, opera como um sensor no roteador de borda da rede e é capaz de correlacionar as solicitações de conexão TCP de entrada e saída dos agentes de fluxo. As conexões TCP são armazenadas numa estrutura de dados do tipo *bloom filter*, onde cada posição do vetor de bits representa um estado de comunicação de rede. Os resultados mostram que é possível identificar agentes de proxy na rede, no entanto, os autores não deixam claro como detectar aplicações válidas que utilizam proxy reverso para oferecer seus serviços. Além disso, a proposta utiliza base de dados de 2004 para identificar FFSN, enquanto que o primeiro trabalho retrata esse problema no ano de 2007.

Para Zhao e Jin (2015) a detecção de redes de serviço de fluxo rápido observa 3 características associadas ao domínio assim como a interação entre hosts e nome de domínio suspeito. Os autores utilizam o número de endereços IP associados ao nome de domínio, entropia da rede classe B, e a volatilidade de tempo de acesso ao domínio. A volatilidade é obtida a partir de visitas recorrentes ao domínio suspeito, em ambiente seguro, para coletar o tempo de acesso. O modelo é construído a partir de bases maliciosas como malware.com e Atlas, assim como domínios legítimos do Alexa.com,

esse modelo é baseado do algoritmo `CART`. Os resultados mostram uma acurácia de 89,1%, os quais podem ser considerados promissores, uma vez que essa proposta utiliza apenas 3 características. Por outro lado, as visitas recorrentes ao nome de domínio podem ser identificadas e bloqueadas pelos operadores da *botnet*.

Martinez-Bea et al. (2013) apresentam uma abordagem de identificação de redes de serviço de fluxo rápido através do classificador `SVM`. O principal objetivo é classificar uma rede de fluxo rápido observando um novo conjunto de características formado pela união dos atributos utilizados em (Hsu et al., 2010) e (McGrath et al., 2009). Os atributos são definidos em dois grupos: características baseadas no tráfego DNS e características intrínsecas ao comportamento das máquinas infectadas.

Essa abordagem parte do princípio que um operador de rede maliciosa é capaz de manipular o comportamento da *botnet* como o número de endereços IP que são retornados em cada consulta DNS (Knysz et al., 2011), mas não consegue modificar o comportamento intrínseco dessas rede como o tempo de ida e de volta (`RTT`) dos pacotes. Os resultados demonstram que é possível identificar redes de fluxo rápido com 99% de acurácia, no entanto, os autores utilizam métodos de manipulação de pacotes que podem ser identificados por *botmasters*.

Sabendo que as *botnets* possuem comportamento evolutivo, o trabalho em (Jiang e Li, 2017) apresenta um estudo comparativo entre métodos supervisionados de aprendizagem de máquina, `SVM`, `Naïve Bayes` e `KNN`. Os autores utilizam 5 atributos frequentemente utilizados na literatura para classificação. Os resultados mostram que para a base de dados investigada, o classificador `SVM` apresentou melhores resultados 97,6% de acurácia.

Al-Nawasrah et al. (2018) apresentam uma nova abordagem para identificar as redes de serviço de fluxo rápido considerando que o *botmaster* é capaz de manipular o comportamento desse tipo de rede. Os autores utilizam um conceito evolutivo baseado em Redes Neurais Dinâmicas de Picos - *Dynamic Spiking Neural Network* (`DeSSN`). Os SNNs operam usando picos, que são eventos discretos que ocorrem em pontos no tempo, em vez de valores contínuos. Essencialmente, uma vez que um neurônio atinge um certo potencial, esse neurônio é redefinido. Isto é, se houverem mudanças no comportamento dos atributos fornecidos, então a rede é atualizada. No entanto, embora o modelo consiga identificar mudanças de comportamento em tempo real, o trabalho em (Knysz et al., 2011) já demonstrou que atributos escolhidos podem ser manipulados pelo operador da *botnet*.

Resumo dos atributos utilizados no método ativo

As Tabelas 3.1 e 3.2 apresentam um resumo sobre os métodos, protocolos e atributos utilizados por cada abordagem direcionada para detecção de redes de serviço de fluxo rápido.

Tabela 3.1: Abordagens e Métodos utilizados.

ID	Trabalho	Métodos	Protocolos
T_1	(Salusky e Danford, 2008)	-	DNS
T_2	(Holz et al., 2008)	Função Linear	DNS
T_3	(Nazario e Holz, 2008)	Cluster	DNS
T_4	(Passerini et al., 2008)	NaiveBayes	DNS
T_5	(Caglayan et al., 2009a,b)	NaiveBayes	DNS
T_6	(Hu et al., 2009)	Hipótese Sequencial	DNS,HTTP,Flow
T_7	(Huang et al., 2010)	NaiveBayes	DNS,TimeDB
T_8	(Campbell et al., 2011)	Hipótese/Cluster	DNS
T_9	(Knysz et al., 2011)	Holz/RBSeek	DNS
T_{10}	(Lin et al., 2013)	AG-Holz	DNS
T_{11}	(Martinez-Bea et al., 2013)	SVM	DNS/HTTP
T_{12}	(Al-Duwairi e Al-Hammouri, 2014)	BloomFilter	HTTP
T_{13}	(Zhao e Jin, 2015)	CART	DNS
T_{14}	(Jiang e Li, 2017)	SVM	DNS
T_{15}	(Al-Nawasrah et al., 2018)	Rede Neural	DNS

Atributos do Método Ativo:

1. Número de IP encontrados no registro do tipo A
2. Número de IP encontrados no registro do tipo NS
3. Tempo de vida (TTL) baixo $<$ 1800 segundos
4. Diversidade dos endereços IP do A ou NS (provavelmente entropia)
5. Número único de ASN para o tipo A ou NS (ambos)
6. Distância entre endereços IP do tipo A
7. Tempo `retry` do tipo SOA
8. Idade do domínio (DomainAge)
9. Nome do Domain Provider (Registrar)

Tabela 3.2: Resumo dos atributos utilizados para detecção de FFSN (Ativo).

		Trabalhos														
		T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}
Atributos	A1	X	X	X	X	X	X	X	X	X	X	X	-	X	X	X
	A2	X	X	X	X	X	X	X	X	X	-	X	-	-	X	-
	A3	X	X	X	X	X	X	-	X	X	-	-	-	-	X	-
	A4	X	-	-	-	X	-	-	-	X	-	-	-	-	-	X
	A5	-	X	X	X	-	-	-	-	X	X	-	-	-	-	X
	A6	-	-	X	X	-	-	-	-	-	-	-	-	-	-	X
	A7	-	-	X	X	-	-	-	-	-	-	-	-	-	-	X
	A8	-	-	-	X	-	-	-	-	-	-	-	-	-	-	-
	A9	-	-	-	X	X	-	-	-	-	-	-	-	-	-	-
	A10	-	-	-	X	-	-	-	-	-	X	-	-	-	-	-
	A11	-	-	-	-	-	X	-	-	X	-	-	-	-	-	-
	A12	-	-	-	-	-	-	X	-	-	-	-	-	-	-	X
	A13	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-
	A14	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-
	A15	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-
	A16	-	-	-	-	-	-	-	X	X	-	-	-	-	-	-
	A17	-	-	-	-	-	-	-	-	-	X	X	-	X	X	X
	A18	-	-	-	-	-	-	-	-	-	-	X	X	X	X	X
	A19	-	-	-	-	-	-	-	-	-	-	X	-	-	-	-
	A20	-	-	-	-	-	-	-	-	-	-	X	-	-	-	-

10. Número distinto de organizações por roteador (traceroute)
11. Nome suspeito de endereço IP de cliente banda larga (dialup, customer, ppoe...)
12. Entropia da fuso horário dos IP associados ao tipo A[‡]
13. Entropia do fuso horário dos IP associados ao tipo NS[‡]
14. Média das distâncias dos serviços mínimos
15. Desvio padrão das distâncias dos serviços mínimos
16. Prefixo único /8, /16 das redes
17. Desvio padrão RTT[⋆]
18. Número de Países
19. Tempo de obtenção de documento[⋆]
20. Tempo de processamento errado[⋆]

Explicações de métricas

Índice de fluxiness: o índice de *fluxiness* utiliza uma única consulta do tipo A no instante t_0 e depois verifica em outros t_i se a quantidade de endereços IP obtidos em t_0 mudou, isto é, $\frac{N_{At0}}{N_{Ati}}$. Esse índice é utilizado apenas para saber se houve mudança nos endereços, pois a classificação é feita a partir do *flux-score* (Holz et al., 2008).

Métricas do tipo A e NS: o trabalho de (Nazario e Holz, 2008) além de extrair o total de endereços IP do tipo A e NS, realiza diferentes cálculos para saber a relação entre os endereços IP. Por exemplo, o domínio é suspeito se o total de IP > 8 e a diversidade for maior que uma rede /16.

Atributos difíceis: alguns atributos demandam tempo extra para coleta, pois estão dependem de inúmeros fatores incluindo restrição por acesso de país, acurácia do protocolo como atributos 8, 9 e 10.

Dependência de bases externas‡: são aqueles atributos que possuam a característica ‡ e são obtidos a partir de bases de dados externas públicas ou proprietárias.

Manipulação/Envio de pacotes*: no método ativo, alguns autores manipulam ou enviam consultas DNS/HTTP forjadas para avaliar o tempo necessário que as máquinas infectadas demoram para processar tais solicitações no entanto, essa abordagem pode ser detectada e interrompida pelo *botmaster*.

3.1.2 Detecção Passiva

Perdisci et al. apresentam um modelo de detecção de redes de serviço de fluxo rápido a partir da análise passiva do histórico do tráfego DNS (Perdisci et al., 2009). Nessa abordagem, o tráfego é monitorado a partir de sensores que operam em colaboração com servidores DNS recursivos, permitindo extrair características de nomes de domínios de maneira silenciosa. As características observadas denotam o tempo de vida (`TTL`); quantidade de endereços IP que atendem um domínio; a proporção de `ASN` por domínio; o total de consultas novas destinadas a um domínio dentro de um espaço de tempo; e total de endereços IP que buscaram por um domínio. Essas características são utilizadas para criar grupos (*clusters*) de nomes de domínio que apresentam comportamento semelhante, e o processo de classificação desses grupos é realizado por meio do algoritmo `C4.5`. Apesar dos resultados mostrarem que o modelo proposto possui acurácia de 99.7% e erra 0.002%, é necessário monitorar o tráfego DNS dentro de uma janela de 24 horas devido à dependência do monitoramento extensivo do tráfego DNS

Stalmans et al. (2012) utilizam uma abordagem baseada na análise de dispersão geográfica de endereços IP que é capaz de detectar nomes de domínios maliciosos hospe-

dados em FFSN. A partir da extração dos endereços IP associados a um nome de domínio, diferentes abordagens de autocorrelação espacial são aplicadas para avaliar se os endereços IP associados ao domínio são próximos ou dispersos. Para identificar a proximidade entre hosts numa determinada região geográfica, os autores utilizam o fuso horário e os sistemas de coordenadas *Universal Transversa de Mercator* (UTM) e o Sistema de Referência Militar de Grade (*Military Grid Reference System* - MGRS). Utilizando uma base proprietária de geolocalização de endereços IP, as coordenadas (latitude/longitude) de cada endereço IP pertencente ao nome de domínio são extraídas e transformadas em coordenadas dos sistemas UTM e MGRS..

A base de dados utilizada possui domínios legítimos obtidos no site alexa.com e domínios maliciosos coletados em *honeypots*. Os resultados demonstram que o sistema de coordenada UTM permite identificar redes de fluxo rápido com 98,54% de precisão. No entanto, o trabalho apresentado não ilustra como as redes que utilizam o tráfego DNS para balanceamento de carga seriam identificadas nesse sistema de coordenadas. Além disso, é necessário uma base proprietária e acurada de geolocalização para que as informações de UTM e MGRS sejam criadas.

Bilge et al. (Bilge et al., 2014) avaliaram diferentes atributos de redes para identificar redes de serviço de fluxo rápido a partir do monitoramento passivo do tráfego DNS. A proposta, denominada Exposure, investiga quais atributos podem ser selecionados para detecção dessas redes maliciosas. Os atributos são extraídos a partir de bases de dados que coletam recursivamente consultas DNS e listas negras de domínios maliciosos. Vale notar que os autores assumem que o tráfego analisado possui algum tipo de comportamento malicioso, pois essas bases de dados representam consultas reais realizadas por máquinas de usuários e, portanto, podem estar infectadas.

Para detectar *botnets* que utilizam redes de serviço de fluxo rápido, o tráfego nome de domínio é observado a partir de diferentes perspectivas incluindo características baseadas em tempo, características extraídas da seção de resposta, métricas com base no tempo de vida e características do nome de domínio. Os nomes de domínios são observados no escopo global (sensores ao redor do globo) e local (sensores locais), o que permite identificar domínios maliciosos recém criados.

Os resultados mostram que o tempo de vida de um domínio malicioso é menor que 5 dias, esse comportamento é observado em 90% dos casos. Além disso, quando comparado com outros métodos de detecção, EXPOSURE é capaz de identificar com antecedência nomes de domínio maliciosos.

Em (Yu et al., 2014), a detecção passiva de redes de serviço de fluxo rápido é realizada a partir da análise de bases de dados de larga escala. Os autores utilizam uma base da

ISC/SIE que é atualiza com milhões de consultas DNS diariamente. Para detecção de FFSN, o tráfego DNS foi coletado no período de Novembro de 2012 até Junho de 2013, representando 10 milhões de consultas válidas para análise e investigação. Essas consultas foram analisadas em estruturas de dados que permitem processamento em paralelo em diversos computadores incluindo `Storm`, `Kafka` e `Hbase`, permitindo assim, 2.8 milhões de consultas DNS por segundo. Os resultados mostram que, em 200 dias de dados DNS coletados, 906 domínios únicos foram classificados como de domínio fluxo rápido. Além disso, foi possível estabelecer uma relação de latência de detecção entre tempo (dias) e o número de consultas enviadas para o domínio malicioso.

Em geral, os domínios maliciosos são detectáveis em até 1 semana (7 dias) ou podem ser identificados quando o número de solicitação a um domínio encontrar determinado limitar (100 consultas) no entanto, ainda existe uma pequena parcela de nome de domínios (4%) que dependem de mais dias para detecção.

Paul et al. (2014) realizam detecção de redes de serviço de fluxo rápido a partir do agrupamento de características do tráfego HTTP. A proposta é dividida em três etapas: extração, agrupamento e classificação dos grupos. Na primeira é feita a extração das características, observando o tamanho do pacote e informação da URI. Os autores partem do princípio que uma *botnet* baseada no tráfego HTTP compartilha comportamentos semelhantes, como o caminho utilizado para enviar informações da máquina infectada. Na segunda etapa, pacotes pertencentes à mesma atividade de rede são agrupados em grupos para investigação, mesmo se tais pacotes sejam coletados a partir de diferentes endereços. Desta forma, a partir da comunicação de vários hosts com o canal de C&C é possível fazer uma análise de comportamento por grupo de máquinas infectadas. Para análise do grupos, os autores utilizam *Power Spectral Density of packet timings* (PSD) que descreve um sinal ou uma série de eventos em uma determinada frequência.

No caso dos hosts infectados, os pacotes agrupados são investigados em intervalos de 1 minuto, conforme definição dos autores. A partir das frequências, foi observado que máquinas infectadas possuem picos proeminentes de comunicação com pontos centrais (C&C), enquanto que hosts legítimos apresentam valores de PSD relativamente baixos. Na última etapa, o classificador examina os dados PSD para picos significativos, ou seja, qualquer amostra de dados que excede 100 vezes o valor médio dos dados totais. Os resultados mostram que utilizando o cálculo PSD, o método proposto conseguiu uma taxa de acerto de 95,8% e de falso positivo 1,16% do total pacotes.

Soltanaghaei e Kharrazi (2015) apresentam um cálculo da taxa de troca de IP de um nome de domínio a partir do monitoramento histórico do tráfego DNS. Os autores utilizam Teste Sequencial Probabilístico de Razão (*sequential probability ration testing sprt*) que

permite definir janelas de monitoramento que acompanham a probabilidade de troca de informação de um domínio. A taxa de troca define a quantidade de novos endereços que foram mudados ou que estão associados ao nome de domínio. Os nomes de domínios são monitorados em uma janela de 24 horas e aqueles que são conhecidos são removidos do processo de monitoramento, permitindo apenas que os nomes de domínios desconhecidos sejam analisados. Os resultados mostram que o método proposto acerta 94% de redes de serviço de fluxo rápido no entanto um atacante pode manipular o comportamento da *botnet* durante as janelas de monitoramento e burlar o método proposto.

Almomani (2016) apresenta uma abordagem, denominada como *fast-flux hunter*, para classificação de redes de serviço de fluxo rápido através do algoritmo *Evolving Fuzzy Neural Networks* (`EFuNN`), que faz parte da área de Inteligência Artificial e representa um conceito dos sistemas de conexão evolucionista (do inglês *Evolving Connectionist Systems* - ECOS). Para classificação de um domínio como `FFSN`, o autor pré-seleciona 14 atributos que são repassados para o algoritmo `EFuNN` que é dividido em 5 camadas: Entrada, Entrada *Fuzzy*, Nós de Regras, Saída *Fuzzy*, Saída. Entrada representa os atributos escolhidos sem qualquer modificação. A Entrada Fuzzy permite combinar alguns atributos para criar *valores fuzzy* que denota essa união. A terceira camada, Nós de Regras, avalia os valores gerados em grupos (*clusters*) observando os erros e a similaridade entre esses valores. Além disso, nessa camada o algoritmo é capaz de aprender (criar regras) as relações entre os grupos gerados. A Saída *Fuzzy* é o processo de validação das regras resultantes, ou seja, se cada valor criado na segunda camada pertence ao seu respectivo grupo. A última fase corresponde à saída dos atributos encontrados.

A abordagem proposta obteve uma taxa de acerto de 98% em menos de 10 segundos. No entanto, o autor não deixa claro a quantidade de tempo que um nome de domínio deve ser monitorado para que as características sejam extraídas.

Lombardo et al. (2018) realizaram uma análise do tráfego `DNS` durante 30 dias a partir de um ambiente controlado para extração de comportamentos maliciosos. Os autores utilizam filtros, listas negras e diferentes métricas para identificar o comportamento de um nome de nome de domínio suspeito. Para tentar identificar os nomes de domínios suspeitos, os autores estabeleceram uma janela de monitoramento mínima de 1 hora. Nomes de domínios que não foram identificados na fase de filtragem são analisados por métricas divididas em dois grupos: métricas estáticas e do comportamento histórico do tráfego de rede.

As métricas estáticas destacam as características da resposta `DNS` includindo tamanho da resposta, quantidade de endereços IP associados ao domínio, quantidade acumulada

de redes distintas, quantidade acumulada de sistemas autônomos, razão da quantidade de sistemas autônomos pelo número de IP associados e a dispersão de endereços IP.

As métricas do comportamento histórico do tráfego DNS demostram como um nome de domínio se comporta durante uma janela de monitoramento. Os autores investigaram a taxa de mudança de endereços IP, do número de redes distintas, da quantidade de sistemas autônomos e do tamanho da resposta DNS.

Os resultados apresentados confirmam outros trabalhos citados na literatura quanto ao comportamento das redes de serviço de fluxo rápido.

Relação dos atributos utilizados no método passivo

As Tabelas 3.3 e 3.4 apresentam os métodos, protocolos e atributos utilizados por cada abordagem direcionada para detecção de redes de serviço de fluxo rápido.

Lista dos atributos utilizados pelo método detecção em modo passivo:

1. Número de IP encontrados no registro do tipo A
2. Número de domínios por cluster
3. Média TTL por domínios num cluster / TTL baixo
4. Número de domínios por rede que responderam a um IP dentro de um cluster
5. Razão de crescimento de IP
6. Diversidade do ASN
7. Diversidade do Prefixo BGP
8. Diversidade da organização (nome do ASN)

Tabela 3.3: Abordagens, Métodos e Protocolos de Rede utilizados.

ID	Trabalho	Métodos	Protocolos
T_1	(Perdisci et al., 2009)	C4.5/Cluster	DNS
T_2	(Stalmans et al., 2012)	Similaridade	DNS
T_3	(Bilge et al., 2014)	C4.5	DNS
T_4	(Yu et al., 2014)	SemiSupervisionado	DNS
T_5	(Paul et al., 2014)	PSD	HTTP
T_6	(Soltanaghaei e Kharrazi, 2015)	SPRT	DNS
T_7	(Almomani, 2016)	Fuzzy	DNS/HTTP
T_8	(Jiang e Li, 2017)	KNN/Bayes	DNS/HTTP

Tabela 3.4: Resumo dos atributos utilizados para detecção de FFSN (Passivo).

		Trabalhos							
		T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
Atributos	A1	X	X	X	X	X	X	X	X
	A2	X	-	-	-	X	-	-	-
	A3	X	-	X	-	-	X	X	X
	A4	X	-	-	-	-	-	-	-
	A5	X	-	-	-	-	-	-	X
	A6	X	-	-	-	-	-	X	X
	A7	X	-	-	-	-	-	-	-
	A8	X	-	-	-	-	-	-	-
	A9	X	-	X	-	-	X	-	-
	A10	X	-	X	-	-	X	-	-
	A11	X	-	-	-	-	-	-	-
	A12	-	X	-	-	-	-	-	-
	A13	-	-	X	X	-	X	-	-
	A14	-	-	X	X	-	X	-	-
	A15	-	-	X	-	-	-	-	X
	A16	-	-	X	-	-	-	-	-
	A17	-	-	X	-	-	-	-	-
	A18	-	-	X	-	-	X	-	-
	A19	-	-	X	-	-	-	-	-
	A20	-	-	X	-	-	-	-	-
	A21	-	-	X	-	-	-	-	-
	A22	-	-	X	-	-	-	-	-
	A22	-	-	X	-	-	-	-	-
	A24	-	-	-	-	-	X	-	-
	A25	-	-	-	-	-	X	-	X
	A26	-	-	-	X	-	-	-	X
	A27	-	-	-	-	X	-	-	-

9. Diversidade de Países
10. Diversidade de nomes de clientes de banda larga PTR (dialup, modem)
11. Média de uptime (conexão http*/dns) ativos
12. Sistemas de coordenadas (Lat./Long/UTM/MGRS)
13. Número de domínios que compartilham IP
14. Domínios de vida curta
15. Similaridade diária

16. Padrões semelhantes/repetitivos
17. Razão de acesso
18. Desvio padrão e/ou média TTL
19. Número distinto de valores de TTL
20. Número de trocas de TTL
21. Percentual de uso de determinada faixa de TTL
22. Percentual de caracteres numéricos
23. Percentual do LMS
24. Média do tamanho da solicitação
25. Média do tamanho da solicitação/resposta
26. Entropia /8 /12 /16
27. Métricas HTTP (tamanho pacote, endereço URI)

Explicações de métricas

Filtragem no modelo passivo: algumas abordagens de detecção de FFSN no modo passivo utilizam filtragem do tráfego DNS para garantir que os nomes de domínios a serem monitorados possuem características mínimas das redes maliciosas. Por exemplo, Perdisci et al. podem filtrar domínios que possuem o TTL $\leq$ 10800 segundos, total de endereços IP observados associados ao nome de domínio ou o prefixo de endereços encontrados.

3.2 Domínios de fluxo rápido

Antonakakis et al. (2010) apresentaram um sistema de reputação dinâmica para nomes de domínios novos ou desconhecidos no ccTLD do Canadá (`.ca`). O sistema, denominado `Notos`, analisa o comportamento do tráfego DNS e atribui uma pontuação de acordo com as atividades relacionadas com o domínio investigado. O comportamento histórico do DNS é obtido a partir de bases legítimas e maliciosas. O comportamento legítimo é coletado em servidores recursivos DNS, enquanto o tráfego malicioso é obtido através de sensores de rede como *honeypots*, *spam-traps* e *sandboxes*. Para facilitar a análise do tráfego, os

autores agruparam domínios de comportamentos semelhantes observando características léxicas do nome de domínio (e.g.: frequência de caracteres e tamanho do nome) e de rede (e.g.: número AS e total de endereços IP associados ao domínio). A partir dessas características, nomes de domínios que apresentem os comportamentos conhecidos podem ser classificados como benignos ou suspeitos.

Shin e Gu (2010a) apresentaram uma visão global do número de máquinas infectadas pelo *malware Conficker* (25 milhões de vítimas). Os resultados mostraram que 99% de vítimas de um domínio específico são clientes de banda larga e responsáveis pela maioria dos emails de *spam* enviados. Esse resultado é semelhante ao trabalho de Barbosa e Souto (2009), o qual aponta o uso de clientes de banda larga do Brasil (.br) varrendo por endereços de servidores de email. Para Shin e Gu, os domínios `.br`, `.net` e `.cn` representam 24% do total de vítimas do *Conficker*, onde o continente Asiático e da América Sul são os principais focos de infecção. Além disso, os autores mostraram que apenas 17,18% do total de 24.912.492 vítimas foram corretamente identificadas por listas negras, o que indica que outros tipos de identificação são necessários.

Stone-Gross et al. (2009) apresentaram uma visão geral de como funciona um canal de comando e controle de *botnets*. Durante o período de 10 dias, o canal de C&C da *botnet Torpig* foi sequestrado permitindo que pesquisadores pudessem entender seu funcionamento, aplicando técnicas de engenharia reversa ao algoritmo de DGA do bot. Stone-Gross et al. observaram que durante a comunicação com o C&C, cada *bot* possuía um identificador único, o qual foi utilizado para estimar com precisão o tamanho da *botnet* (aproximadamente 180 mil máquinas). Em 10 dias, mais de 49 mil novas máquinas foram infectadas e buscaram o C&C monitorado pelos pesquisadores.

Yadav et al. (2012) partem do princípio que domínios gerados por algoritmos são diferentes em termos de padrões de nomes quando comparados aos domínios registrados por seres humanos. Para entender a relação entre domínios, Yadav et al. observaram a distribuição alfanumérica e o bigrama dos caracteres utilizados no nome de domínio. Além disso, os autores assumem que os nomes gerados por algoritmos tendem a ser impronunciáveis, apesar da *botnet* `Kraken` (Royal, 2008) ser capaz de produzir nomes mais próximos da língua inglesa.

Para validar a proposta, os autores utilizaram domínios legítimos, obtidos através da varredura do espaço de endereçamento do IPv4, enquanto os domínios maliciosos foram coletados por domínios gerados pelas *botnets* `Conficker` e `Kraken`. O modelo proposto foi validado em um tráfego DNS real de grandes servidores de Internet da América do Sul e Ásia. Os domínios foram agrupados através de um conjunto de características, tais como nível do nome de domínio, relação entre endereços IP e a relação entre domínio e

endereço IP. Durante os experimentos foi possível observar que a partir da distribuição de frequência dos caracteres, as letras 'm', 'o' e 's' em domínios legítimos não eram uniformes, enquanto as vogais 'a', 'e', e 'i' possuíam distribuição uniforme para domínios maliciosos.

A distribuição e probabilidade de caracteres também é avaliada em (Truong e Cheng, 2016). Os autores verificam o tamanho do nome gerado, a distribuição de frequência e se um determinado conjunto de caracteres foi utilizado. Os resultados mostram que o método proposto atinge a maior eficiência de detecção utilizando algoritmo de árvore de decisão (J48), com uma precisão de até 92,3% e uma taxa de falsos positivos de 4,8%. Para encontrar esses resultados, o método proposto utiliza bases de dados que foram geradas por duas versões de código malicioso: `Confiker` e `Zeus` no entanto, os autores não comparam com versões avançadas de DGA como a *botnet* MatsuBotnet que é capaz de utilizar verbos e substantivos na geração de nome de domínio.

A Tabela 3.5 sumariza as principais características utilizadas pelos trabalhos citados nesta Seção para detecção de *botnets* baseadas em troca rápida de informação. Algumas características podem ser comuns aos dois tipos de anomalia (*fast-flux* e *domain-flux*). Entretanto, as redes de fluxo rápido são geralmente identificadas por meio da correlação dos recursos de rede, enquanto os domínios de fluxo rápido são investigados pelas características do nome do domínio.

Tabela 3.5: Características utilizadas para classificação de botnets baseadas no tráfego DNS.

Fast-Flux	Recursos de Rede	Tempo de vida do pacote (TTL)
		Endereço IP obtido através dos registros A, NS e SOA
		Número do sistema autônomo (ASN)
		Prefixo do endereço do roteador de borda (BGP)
		Retorno da consulta reversa de um domínio
		Data do registro do domínio (whois)
		Nome da operadora que controla os registros de domínio
	Estatística de Rede	Total de endereços IP que respondem por um registro do tipo A durante um intervalo de tempo (t)
		Distância entre os endereços que respondem por um registro do tipo A, NS e SOA
		Total de endereços IP únicos obtidos em uma consulta do tipo A, NS e SOA
		Distância entre os endereços IP de nameservers
		Nomes dos roteadores obtidos através do traceroute
		Desvio padrão do RTT entre o cliente e todos os endereços IP obtidos através dos registros A, NS e SOA
		Total de endereços IP que se sobrepõem
		Número único de endereços IP encontrados nos registros A e NS
		Porcentagem de distribuição geográfica dos endereços IP
		Média do TTL durante um intervalo de tempo (t)
Domain-Flux	Recursos de Rede	Consultas de domínios com erro (NX-Domain, SRVFail)
	Estatísticas do Nome de Domínio	Distribuição de frequência de caracteres de [a-z] e [0-9]
		Total de caracteres
		Total de níveis de sub-domínio
		Frequência de caracteres por nível de domínio
		Número de domínios que retornam o mesmo IP
		N-grama do nome
		Entropia dos sub-domínios
		Média, mediana, desvio padrão e variância para o nome e sub-domínios especiais em uma sentença e no tópico do canal
		Total de domínios de primeiro nível
		Distribuição do total de níveis de sub-domínios
		Similaridade entre domínios e n-gramas

Vale ressaltar que além das características ilustradas na Tabela 3.5, alguns trabalhos também utilizam fontes externas como `URL` em *spam*, listas negras, listas brancas, *honeypots*, engenharia reversa, base de dados geográficas e o fuso horário.

3.3 Otimização por Colônia de Formigas

Algoritmos de Colônia de Formigas têm sido aplicados em diferentes áreas de pesquisa como ilustrado no Capítulo 2. Esta Seção apresenta abordagens que utilizam algoritmos de Colônia de Formiga para detecção de ataques ou comportamentos maliciosos de *botnets*.

3.3.1 Detecção de Ataques Baseada em Colônia de Formiga

Redes descentralizadas e distribuídas são vulneráveis à ataques onde um nó malicioso pode alegar que representa múltiplas identidades, ou seja, um nó pode fingir ser uma identidade que não existe na rede, inundando a comunicação entre hosts com informações forjadas, esse ataque é denominado como *Sybil Attack* (John et al., 2015).

Para identificar nós forjados em redes P2P, (Zeng e Chen, 2010) apresentam um algoritmo denominado como `SybilACO`, baseado no conceito de Colônia de Formigas. O objetivo desse algoritmo é permitir que determinado host consiga avaliar se seus nós vizinhos são válidos ou forjados para troca de informações como download de arquivo. Quando o sistema é iniciado, cada formiga posicionada em determinado nó, anda no grafo aleatoriamente a partir da probabilidade definida em (Dorigo e Stützle, 2010). Caso um nó de destino (vizinho) seja cruzado pela mesma formiga pelos menos duas vezes, a partir de rotas diferentes, então esse nó vizinho é válido para troca de informações.

Os conceitos de Colônia de Formiga também podem ser utilizados para identificar sequências de eventos que denotam o comprometimento de um sistema ou usuário. Por exemplo, em (Abadi e Jalili, 2006), os autores modelaram vulnerabilidades de sistemas e *exploits* em um grafo denominado como *grafo de privilégios*, onde cada nó representa um conjunto de privilégios ou sistemas e as arestas são os possíveis mecanismos de exploração. A modelagem dos sistemas inclui serviços de rede (e.g: ftp, http), hosts na rede, relações de confiança (e.g: conexão válida entre hosts) e *exploits*.

Para minimizar a análise do grafo de privilégios representando ataques em larga escala, o conceito de Colônia de Formiga é utilizado. No algoritmo original de formigas, três passos são realizados: construção das soluções, procedimento de poda e atualização de feromônio (Dorigo e Stützle, 2010). Na abordagem proposta, a construção das soluções representa a definição de um conjunto de *exploits* que são acrescentados incrementalmente até que um ataque seja concluído com sucesso, esses *exploits* são adicionados a cada caminho tomado pela formiga. No entanto, é possível que *exploits* redundantes tenham sido adicionados, e por isso, devem ser removidos com processo de poda. O último passo é a realização da atualização de feromônio, onde os caminhos que tiveram o melhor ganho para um conjunto de *exploits* são atualizados.

A partir de um ambiente simulado formado por diferentes sistemas operacionais, serviços com problemas de vulnerabilidade e *exploits*, os resultados mostram que a proposta consegue identificar conjuntos críticos de *exploits* que poderiam ser utilizados por atacantes para ganhar acesso privilegiado aos sistemas.

Em ataques distribuídos de negação de serviço, parte dos endereços IP que realizam o ataque podem ter sua origem forjada para dificultar a detecção. Desta forma, para reduzir o volume do ataque e bloquear os verdadeiros hosts, a técnica de `IP traceback (IPTBK)` pode ser aplicada nesse processo de identificação. O problema nesses ataques é que uma parte dos endereços são de origem forjada (*spoofed IP)*) portanto, o objetivo é utilizar os sistemas de Colônia de Formiga (ACS - *ant colony system*) para minimizar o número de rotas de pacotes necessárias para reconstruir o caminho do ataque. Os autores modelam as conexões entre os nós da rede usando o esquema denominado como Waxman, onde os nós são distribuídos uniformemente em uma área retangular de coordenadas. Após a definição do modelo de conexão entre os nós, as formigas caminham no grafo a partir de duas técnicas: de exploração e viciada (*biased*). Na técnica de exploração, as formigas observam as arestas que possuem maior quantidade de *feromônio* e *visibilidade*. Por outro lado, a técnica viciada utiliza a probabilidade definida por Dorigo.

Li et al. (2015) tentam resolver o problema de IP `traceback` a partir da combinação de PSO (*Particle Swarm Optimization*) e algoritmos de agrupamento como o `K-Means`. O uso de partículas pode ser direcionado para combinar diferentes soluções candidatas, por meio de uma função objetivo que identifica a qualidade da solução escolhida, enquanto o algoritmo *K-means* utiliza as partículas geradas pelo PSO como centroides e, desta forma, classificar corretamente o problema.

Chen e Huang (2016) apresentam um novo método baseado em otimização por colônia de formigas para detectar ataques distribuídos de negação de serviço de baixa frequência. Esse método é dividido em três fases: filtragem, algoritmo multiagente e busca *backward* e *forward*. Na filtragem, é realizado o processo de identificação dos nós (roteadores) que estão recebendo um volume de pacotes acima do limiar previamente estabelecido. O algoritmo multiagente utiliza o comportamento das formigas para percorrer todos os roteadores da rede. Para as formigas visitarem outros nós na rede, é realizado o cálculo do valor máximo da multiplicação entre o *feromônio* e *heurística* caso esse valor for menor que q_0 (Dorigo e Stützle, 2010) caso contrário, a função probabilística do ACO é aplicada. Finalmente, é aplicada a busca *backward* e *forward* com objetivo de identificar outros roteadores que possam estar sofrendo ataques. Os resultados mostram que a taxa de acerto é de 89% e acurácia do método reflete 83%.

3.3.2 Discussão sobre Redes Maliciosas e Colônias de Formigas

A detecção de *botnets* baseadas em redes de serviço de fluxo rápido é um problema complexo de ser resolvido devido às estratégias que *botmasters* podem adotar para escapar

de sistema detectores de intrusos. As abordagens de detecção estão concentradas em análise passiva, ativa ou análise híbrida do tráfego DNS, no entanto, nenhuma é capaz de resolver o problema quando um domínio é analisado em tempo real ou utilizando baixa interação com a rede maliciosa.

Para tentar mitigar o avanço das redes de serviço de fluxo rápido, é preciso analisar o problema a partir de diferentes perspectivas e assumir que o *botmaster* é capaz de mudar ou bular métodos de identificação como citado em (Knysz et al., 2011). Entretanto, é razoável supor que o operador não possui métodos que consigam alterar diferentes abordagens e comportamentos ao longo do tempo. Por exemplo, nos trabalhos que tratam do tempo de ida e de volta do pacote (Lin et al., 2013; Martinez-Bea et al., 2013), fica claro que o *botmaster* não possui controle de quanto tempo uma consulta DNS ou HTTP para sua rede maliciosa demora.

A resolução do tempo de ida e de volta pode ser resolvido parcialmente caso o operador da rede utilize em sua estrutura consultas DNS com a extensão *EDNS-client-subnet* (Calder et al., 2013), a qual fornece uma porção do endereço IP de origem ao servidor DNS para que esse último seja capaz de informar hosts mais próximos à origem da consulta. No entanto, vale ressaltar que para que essa *botnet* consiga fornecer endereços IP mais próximos da origem da consulta, a rede maliciosa deve possuir um número considerado de máquinas infectadas ao redor do globo.

Supondo que o *botmaster* seja capaz de fornecer respostas DNS mais próximas da origem, o operador ainda teria que ter um controle restrito quanto à disponibilidade dos hosts infectados para que o domínio malicioso sempre esteja disponível. Uma possível solução para esse problema é enviar informações sobre o estado da máquina infectada para o canal de comando e controle usando mecanismos como `ping` ou solicitação HTTP para um site controlado pelo atacante. Embora essa abordagem forneça algum tipo de controle sob hosts ativos (não desligados), o *botmaster* não teria controle quanto ao nome reverso (tipo *PTR*) associado ao endereço IP de uma máquina de rede banda larga, permitindo que algumas abordagens ainda identifiquem a *botnet* (Holz et al., 2008; Hu et al., 2009).

De maneira geral, mesmo que a manipulação dos atributos ou comportamento da rede maliciosa consiga ludibriar algumas soluções apresentadas, não é factível que o operador da rede maliciosa consiga burlar diferentes atributos para métodos de detecção distintos durante um tempo considerável de monitoramento.

Diante desse contexto, o conceito de Colônia de Formiga pode ser aplicado para auxiliar no processo de detecção de *botnets* baseadas em redes de serviço de fluxo rápido. No caso do problema do Caixeiro Viajante, as formigas são utilizadas para visitar diferentes caminhos e informar a menor rota encontrada. Por outro lado, para detecção de *botnets*,

uma alternativa é modelar os atributos de rede (características) como cidades e permitir que as formigas selecionem os atributos mais relevantes para resolver esse problema. As formigas seriam responsáveis pela seleção dos atributos (Jensen, 2006) e um classificador como `SVM` ou `J48` poderia ser utilizado para classificação. Essa abordagem, por outro lado, consegue identificar aqueles atributos mais relevantes para determinada instância do problema, o que resultaria num processo contínuo de avaliação e identificação de atributos para serem utilizados na classificação.

Outra alternativa é utilizar soluções onde o próprio classificador aplica o conceito de colônia de formigas, como o algoritmo supervisionado `Ant-Miner` (Parpinelli et al., 2002a). Durante a fase de construção das soluções, as formigas avaliam através da função heurística, qual par formado por atributos e valores são mais relevantes para o processo de classificação. Ao final desse algoritmo, é possível obter um modelo de classificação definido por conjunto representativo de atributos e valores que denotam uma instância do problema. Desta maneira, fica claro que os resultados do `Ant-Miner` dependem de como a função heurística é aplicada nesse processo, por isso, diversos trabalhos apresentaram melhorias dessa função (Baig e Shahzad, 2012; Hodnefjell e Costa Junior, 2012).

Para tentar resolver o problema de classificação de *botnets* baseadas em redes de serviço de fluxo rápido considerando os problemas citados acima, nesta tese de doutorado, o conceito de colônia de formigas é utilizado para avaliar diferentes métodos de detecção e seus respectivos conjunto de atributos, para tanto, é apresentado um *framework* denominado `AFFACO`. Os capítulos a seguir descrevem com maiores detalhes o funcionamento desse sistema.

4

AFFACO - Anti Fast-Flux based on Ant Colony Optimization

A detecção de redes de serviço de fluxo rápido é um problema em constante evolução, pois essas redes são capazes de utilizarem técnicas que permitem evadir de sistemas detectores de intrusos. Vale lembrar, por exemplo, que o trabalho de Knysz et al. demonstrou que os métodos apresentados por Holz et al. (Holz et al., 2008) e Xin et al. (Hu et al., 2009) podem ser subvertidos por operadores a partir da manipulação dos atributos de rede associados à um nome de domínio como a quantidade de endereços IP observados nos registros de recurso do tipo `A` e `NS`, ou o número de redes que respondem pelo domínio.

Para tentar mitigar o avanço dessa redes, diferentes estratégias podem ser adotadas, como demonstrado no Capítulo 3. Em todos os casos citados anteriormente existe um problema em comum: o tempo de classificação. O tempo de classificação é processo que se inicia no monitoramento de um domínio suspeito, incluindo extração de características e tratamento de dados, finalizando com a definição de um modelo capaz de determinar o comportamento do nome de domínio. Isto é, para classificar um comportamento suspeito é necessário, primeiro, coletar atributos que representem o problema.

Embora existam características que são obtidas em questões de segundos, existem outros conjuntos de atributos que demandam maior tempo de monitoramento. Em um modelo hipotético, se o tempo total do processo de extração de dados até a classificação de um domínio suspeito for desconsiderado, então os problemas de *botnets* podem ser resolvidos a partir de um número considerável de características que são obtidas dentro de uma janela de monitoramento compatível com tempo de coleta. Por outro lado, tal

modelo hipotético não pode ser aplicado no mundo real, pois durante ataques de rede, disseminação de código malicioso e recebimento de *spam* em massa, o objetivo é identificar e interromper tais comportamentos no menor tempo possível para que os serviços de rede não sejam paralisados ou novos hosts infectados.

Sabendo que operadores de rede são capazes de manipular o comportamento das *botnets*, diferentes trabalhos buscam identificar novos atributos e estratégias para mitigar o avanço dessas redes. É possível assumir que tal processo gera um ciclo de busca constante, onde é preciso identificar qual melhor conjunto de atributos ou método de detecção que permita classificar as redes maliciosas após mudança de comportamento.

É preciso notar que neste trabalho, a detecção de redes de serviço de fluxo rápido não está sustentada em um *conjunto fixo* de características associadas à um *único método* de classificação. Por isso, este trabalho apresenta um *framework* para analisar e detectar redes de serviço de fluxo rápido a partir de diferentes perspectivas sob o mesmo problema. Para que seja possível analisar as redes de fluxo rápido a partir de diferentes pontos de vistas é necessário definir critérios de investigação. Por exemplo, um operador de rede pode tentar identificar um domínio suspeito ordenando um conjunto de métodos de classificação pelo menor número de atributos; pelo total de consultas enviadas para uma base de dados externa; utilizando os métodos com maior acurácia; ou selecionado atributos baseados em determinado tipo de protocolo de rede.

Neste trabalho o critério de identificação de um domínio suspeito é baseado em métodos que utilizam o *menor* número de atributos e que forneçam *maior* acurácia. Isto é, o objetivo é minimizar o custo de extração de características e maximizar o acerto de classificação.

Sabendo que a detecção de redes de serviço de fluxo rápido pode ser tratada como um problema de otimização combinatória, é necessário identificar as principais restrições associadas ao cenário descrito para que uma solução possa ser aplicada na resolução do problema. Partindo do princípio que todos os métodos possuem comportamento determinístico, isto é, a mesma entrada sempre fornece a mesma saída, cada domínio suspeito ser deve analisado uma única vez por cada método de detecção. Além disso, a ordem em que o nome de domínio é analisado deve ser levada em consideração, pois o objetivo é minimizar o custo de classificação. Outra característica relevante é que deve existir um equilíbrio entre custo e acurácia, caso contrário, pode-se acreditar que ordenar os métodos de detecção do menor para o maior custo é suficiente para resolver o problema ou ainda, selecionar as abordagens com maior acurácia primeiro não aumentaria o custo de classificação do domínio suspeito. O Capítulo de Resultados apresenta mais detalhes sobre esse estudo.

É importante notar que as restrições descritas nesse trabalho tentam resolver a classificação no menor tempo obtendo ganho de acurácia. No entanto, esse problema ainda pode incluir outros requisitos como a redução de sobreposição de atributos entre cada método de detecção e o limite de protocolo de rede ou algoritmo/técnica por cada intervalo de método selecionado. Por exemplo, assumindo que os dois primeiros métodos selecionados utilizam o mesmo algoritmo como (`SVM`), então o próximo método deve ser distinto dos anteriores. A ideia é anular qualquer espaço que o atacante possui para manipular as características de rede e, assim, evadir os sistemas de detecção. Vale dizer que as últimas duas restrições estão fora do escopo desse trabalho.

Dado as restrições supracitadas, o conceito do Caixeiro Viajante pode utilizado como inspiração para resolver esse problema, pois cada cidade pode ser observada como um método de detecção de redes de serviço de fluxo rápido. O Problema do Caixeiro Viajante tem sido abordado na literatura por diferentes trabalhos (Hoffman et al., 2013; Hussain et al., 2018). De maneira geral, existem diferentes heurísticas que tentam encontrar uma solução próxima ótima para esse tipo problemas de otimização combinatória, como a Busca Tabu, Algoritmos Genéticos, Algoritmos por Busca Local e Otimização por Colônia de Formiga.

Algoritmos Genéticos e Otimização por Colônia de Formigas (`ACO`) são frequentemente aplicados na resolução do Problema do Caixeiro Viajante devido suas semelhanças, possibilidade no ajustes de parâmetros e desempenho na solução de algumas instâncias conhecidas na literatura (Haroun et al., 2015). Em trabalhos mais recentes, ambos algoritmos estão sendo combinados de tal maneira para criar uma versão híbrida que contemple tanto as características dos Algoritmos Genéticos como de Colônia de Formigas (Ciornei e Kyriakides, 2012; Donis-Díaz et al., 2015; Thangamani e Chidambaram, 2018).

A Otimização por Colônia de Formigas é escolhida neste trabalho por ter sido aplicada em problemas em redes de computadores (Dorigo e Stützle, 2004), detecção de anomalias de redes (Ranga e Mandhar, 2018) e detecção hosts infectados na rede (Wu et al., 2018), assim como em problemas com multiobjetivo (Mokhtari e Ghezavati, 2018). No entanto, até onde vai o conhecimento deste autor, a aplicação dos conceitos de Otimização por Colônia de Formigas para detectar *botnets* baseadas em Redes de Serviço de Fluxo Rápido é inovador.

4.1 Visão Geral do Framework AFFACO

Para detectar Redes de Serviço de Fluxo Rápido a partir de diferentes métodos de detecção (perspectivas), este trabalho apresenta um *framework* inspirado no conceito de

Otimização por Colônia de Formigas, denominado `AFFACO` *Anti Fast-Flux Based on Ant Colony Optimization*.

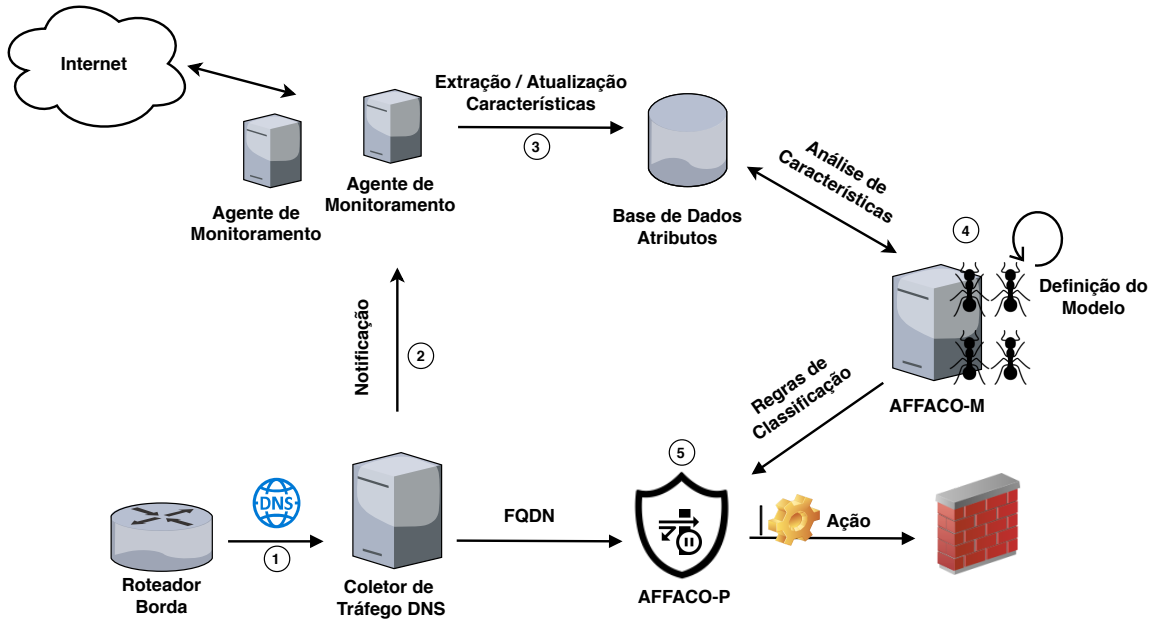


Figura 4.1: Visão geral da arquitetura para detecção de redes maliciosas baseadas em serviços de fluxo rápido (Fonte: Próprio Autor).

A Figura 4.1 apresenta os principais componentes utilizados no *framework* proposto, desde o processo de extração de tráfego de rede até o processo de identificação do comportamento do nome de domínio. Os componentes podem ser organizados em três grupos: *Extração e Análise de Tráfego de Rede*, *Definição de Modelos* e, finalmente a *Classificação* de nomes de domínios. No processo de Extração e Análise do Tráfego de Rede (passo 1) o *coletor de tráfego* é um sistema capaz de extrair do tráfego de rede consultas `DNS` ou coletar nomes de domínios cadastrados em bases de dados como `PhishTank`[1] ou `MalwareDomains`[2]. A partir dos nomes de domínios, os *agentes de monitoramento* são notificados (passo 2) para que as características previamente definidas pelos métodos de detecção sejam extraídas, manipuladas e armazenadas em uma base de dados (passo 3).

Em seguida, os componentes da Definição de Modelos (passo 4) têm como objetivo correlacionar os métodos de detecção para que *modelos de classificação* de redes de serviço de fluxo rápido sejam definidos com base na restrição do problema. O processo de

[1] `https://www.phishtank.com/`
[2] `http://www.malwaredomains.com/`

correlação desses métodos é realizado pelo algoritmo AFFACO-M utilizando os conceitos de Otimização por Colônia de Formigas.

Neste trabalho, um modelo de classificação é uma sequência métodos de detecção que juntos formam uma *regra* que permitem que um nome de domínio seja observado por diferentes perspectivas. Além disso, uma regra pode utilizada em sistemas de detecção de intrusos como Snort[3] ou Bro[4].

Para demonstrar o funcionamento do *framework* proposto, é utilizado como prova de conceito o componente AFFACO-P (passo 5). Conforme o processo de avaliação do nome de domínio acontece a partir da regra gerada, o componente AFFACO-P faz a fusão dos resultados e a partir de uma política de votação, tal componente pode tomar uma decisão em forma de ação.

As seções a seguir apresentam com mais detalhes o funcionamento de cada componente do *framework* utilizado para classificar redes de serviço de fluxo rápido. Primeiro, é demonstrado o processo de modelagem de uma rede de serviço de fluxo rápido a partir dos conceitos de Otimização por Colônia de Formiga. Tal modelagem permite entender as diferentes etapas do algoritmo AFFACO-M. Segundo, os conceitos e o sistema de políticas do fusor de resultados AFFACO-P são apresentados como prova de conceito. Finalmente, os componentes para Extração e Análise de Tráfego de Rede são descritos incluindo o funcionamento do algoritmo utilizado para coletar características e atualizar de bases de dados.

4.2 Modelando FFSN em Colônias de Formiga

A detecção das redes FFSN é desafiadora devido à capacidade do operador da *botnet* manipular algumas características para evadir os sistemas de detecção ou simular comportamento de serviços legítimos, como ilustrado no Capítulo 3. Portanto, a detecção desse tipo de rede deve partir da analise por meio diferentes métodos de detecção e seu conjunto de características.

A principal ideia que sustenta essa nova tratativa está na própria dinâmica adotada por operadores de rede maliciosas para que a *botnet* funcione por mais tempo. Vale notar que embora seja possível assumir que o *botmaster* seja capaz de manipular determinados comportamentos da rede maliciosa, como citado no Capítulo 3. É impraticável tanto do ponto de vista da natureza desse tipo de rede como na questão econômica que operador

[3] https://www.snort.org/
[4] https://www.bro.org/

consiga *i*) alterar as características de rede durante longos períodos de monitoramento e, ao mesmo tempo, *ii*) subverter diferentes abordagens.

Para ilustrar o raciocínio descrito acima, considere que o *botmaster* tenha lançado uma campanha de venda de medicamentos fármacos em determinada região do mundo (na Ásia, por exemplo) e, com objetivo de manter esse esquema fraudulento funcionando por mais tempo, é necessário manter alguns requisitos. O operador dessa rede deve ter um número considerável de máquinas infectadas naquela região para simular o comportamento de uma `CDN` legítima e, com isso, burlar métodos de detecção que dependem do número distinto de `ASN` por nome de domínio (Perdisci et al., 2009). Além disso, o operador da rede maliciosa precisa ter certo controle da disponibilidade das máquinas controladas para que apenas as máquinas acessíveis sejam incluídas nas repostas `DNS` (Chen et al., 2013b).

Não obstante as táticas para evadir algumas abordagens, o *botmaster* também deve se preocupar com um conjunto de máquinas que possuem em seu nome DNS reverso (`PTR`) palavras chaves que fazem referência ao serviço de internet de banda larga, como *dialup*, *dyn* ou *velox* (Knysz et al., 2011). Portanto, ao excluir essas máquinas de potenciais respostas `DNS`, o operador deve contratar ou sequestrar hosts de provedores legítimos para que sua campanha esteja sempre disponível para acesso.

A combinação entre hosts legítimos e sequestrados pode resultar em novos problemas para o *botmaster*. Isso acontece devido aos métodos de detecção que observam também o número de redes distintas associadas a um nome de domínio (Passerini et al., 2008; Yu et al., 2014). Para tentar subverter esses métodos de detecção, o *botmaster* deve contratar ou sequestrar mais hosts legítimos de uma mesma empresa, o que aumentaria os custos financeiros de operação da *botnet*.

Diante desse cenário, fica claro que o *botmaster* faz é combinar estratégias para evadir os sistemas de detecção. Portanto, o contra-ataque proposto no *framework* `AFFACO` tem como objetivo combinar os diferentes métodos de detecção para identificar se um domínio é legítimo ou malicioso, isto é, a partir de diferentes perspectivas.

Para tratar o problema das redes de serviço de fluxo rápido, é necessário definir alguns termos inicialmente. O termo Métodos de Detecção é utilizado inicialmente para ajudar a compreender os conceitos empregados aqui, conforme ilustra a Definição 1.

Definição 1 *Métodos de Detecção são quaisquer procedimentos utilizados para identificar o real comportamento de um nome de domínio suspeito.*

No entanto, esse termo é muito abrangente e não reflete com detalhes seu funcionamento. Por exemplo, existem métodos de detecção baseados no protocolo `DNS`, `HTTP` e `P2P`, métodos que utilizam classificação por hiperplano ou agrupamento, ou ainda, aqueles

usam detecção em tempo real ou *offline*. Por esses motivos, na visão desde autor, o termo *abordagem* é mais apropriado para especificar todos os componentes que um método de detecção possui includindo método (e.g.: algoritmo, heurística, probabilidade), protocolos e conjunto de atributos. Desta maneira, a Definição 2 formaliza conceito de abordagem.

Definição 2 *Abordagem é a representação que descreve com detalhes o funcionamento de um método de detecção.*

Ao longo do texto a seguir, o termo abordagem é utilizado com frequência para fazer alusão a um trabalho citado na literatura. Cada abordagem representa o mesmo conceito que uma cidade no problema do Caixeiro Viajante, então para formiga visitar uma próxima abordagem, é necessário realizar um cálculo probabilístico que considera a heurística (η_{ij}) e a quantidade de feromônio (τ_{ij}) da cidade de destino, conforme ilustra a Figura 4.2. As Definições 3 e 4 denotam o significado da heurística e feromônio neste trabalho, respectivamente. A Seção 4.3.1 apresenta maiores detalhes do cálculo probabilístico utilizado pelas formigas.

Definição 3 *Heurística é uma função que denota o custo de sair da abordagem i para j.*

Definição 4 *Feromônio é atratividade da aresta entre as abordagens i e j.*

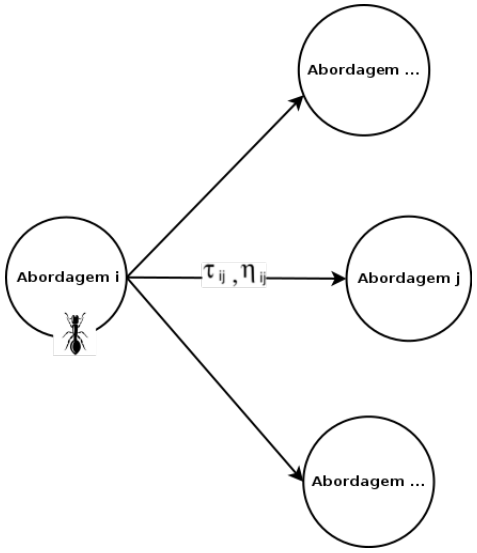


Figura 4.2: Representação das abordagens e relacionamentos (Fonte: Próprio Autor).

Além dos componentes citados acima, cada abordagem selecionada é representada pela tupla composta do *Custo Total de Atributos* (CTA) e *Acurácia* de detecção. Esses

dois componentes refletem quanto tempo uma abordagem precisa monitorar um nome de domínio suspeito para que seja possível determinar corretamente seu comportamento como legítimo ou malicioso. A descrição do `CTA` pode ser observada na Definição 5.

Definição 5 *O Custo Total de Atributos é um método definido empiricamente que representa quanto de esforço (i.e: tempo de coleta de características) é necessário para extrair todas as características utilizas por uma abordagem.*

Adicionalmente, o CTA pode ser representado pela soma de todos os custos por atributo que são utilizados na abordagem dentro de uma janela de monitoramento. É importante notar que no cenário atual, o custo do atributo não considera o protocolo de rede ou procedimento adotado para extrair as características do nome de domínio, quando o ideal é fazer esse escrutínio para distinguir entre abordagens que entregam resultados em poucas consultas (Zhao e Jin, 2015) daquelas que demandam inclusive análise de conteúdo (Atluri e Tran, 2017). A Seção 4.2.2 apresenta mais detalhes sobre a representação de uma abordagem.

Vale lembrar que o principal o objetivo aqui é determinar o comportamento de um nome de domínio no menor tempo e custo (minimização) para obter maior acerto de classificação (maximização). A principal ideia que sustenta essa definição está de acordo com a literatura detalhada no Capítulo 3, ou seja, a maioria dos trabalhos tem por objetivo detectar redes de serviço de fluxo rápido no menor tempo possível.

Sendo um problema de minimização ou maximização, é importante lembrar das propriedades $(\mathcal{S}, f, \Omega)$, tal que $\mathcal{S}$ denota o conjunto de soluções, f a função objetivo e $\Omega(t)$ o conjunto de restrições (Cormen et al., 2009; Dorigo et al., 2000). Neste trabalho, a função objetivo deve *minimizar* a quantidade de atributos e *maximizar* o acerto de classificação considerando as possíveis abordagens que podem ser selecionadas. No contexto de redes de serviço de fluxo rápido, $\mathcal{S}$ é definido pelo *conjunto de atributos de rede correlacionados em diferentes abordagens* que permitem classificar o comportamento de um nome de domínio. A função objetivo $f(s, t)$ verifica se a solução escolhida ($s \in \mathcal{S}$) respeita a restrição t, que nesse caso, é o menor custo que outros modelos de classificação anteriores, permitindo assim que soluções mais rápidas sejam avaliadas primeiro. Além disso, essa função deve maximizar a acurácia do modelo escolhido.

No problema original do Caixeiro Viajante, Dorigo utilizou as formigas para avaliarem os melhores caminhos e determinar a rota mais curta. Em contrapartida, esse processo é visto neste trabalho como um meio de correlacionar diferentes abordagens e suas características. No PCV, a formiga sai da $\texttt{cidade}_i$ para $\texttt{cidade}_j$ caso essa última seja a cidade mais próxima de i. No contexto de detecção de `FFSN`, uma formiga vai para uma

$\texttt{abordagem}_j$ partido da $\texttt{abordagem}_i$, se o custo e acurácia forem atrativos para formiga na $\texttt{abordagem}_i$.

O resultado final no processo combinatório para o PCV é um roteiro de cidades que deve ser seguido pelo vendedor até o retorno à cidade original de saída. Neste trabalho, o caminho é denotado como uma sequência de abordagens que devem ser utilizadas para classificar o nome de domínio.

Os principais componentes descritos na modelagem entre o PCV e o FFSN podem ser observados resumidamente na Tabela 4.1.

Tabela 4.1: Resumo de equivalência entre o modelo do PCV para FFSN.

Definição	**PVC**	**Fast Flux**
Vértices	Cidades	Abordagens
Arestas	Distância	CTA
Objetivo	Menor caminho	Menor custo & Maior acurácia
Movimentação	Probabilidade	Probabilidade
Distância	Entre as cidades	Entre abordagens
Caminho	Sequência de cidades	Sequência de abordagens

As subseções a seguir descrevem com mais detalhes como cada componente citado deve ser modelado. Primeiro, é necessário definir formalmente o grafo que representa as abordagens e suas respectivas relações. Em seguida, é apresentado um modelo representativo de abordagens, o qual pode ser utilizado para mapear as abordagens que serão correlacionadas pelo gerador de modelos `AFFACO`, descrito na Seção 4.3.

4.2.1 Grafo de Correlação de Abordagens

Para entender a relação entre as abordagens de detecção de redes de serviço de fluxo rápido, considere o *Grafo de Correlação de Abordagens*, ilustrado na Figura 4.3. Nesse grafo, os *vértices* estão conectados por uma *aresta*, a qual denota o custo entre as abordagens. Para que todas as abordagens sejam correlacionadas, em cada vértice deve haver pelo menos uma formiga associada. A Definição 6 descreve mais detalhes sobre o Grafo de Correlação de Abordagens.

Definição 6 *Grafo de Correlação de Abordagens é uma estrutura de dados que permite que diferentes abordagens sejam representadas e avaliadas durante processo de busca de soluções do problema.*

É importante observar que no problema original do Caixeiro Viajante, a distância entre as cidades $\texttt{cidade}_i$ e $\texttt{cidade}_j$ é idêntica, i.e $d_{ij} = d_{ji}$. No entanto, para o grafo

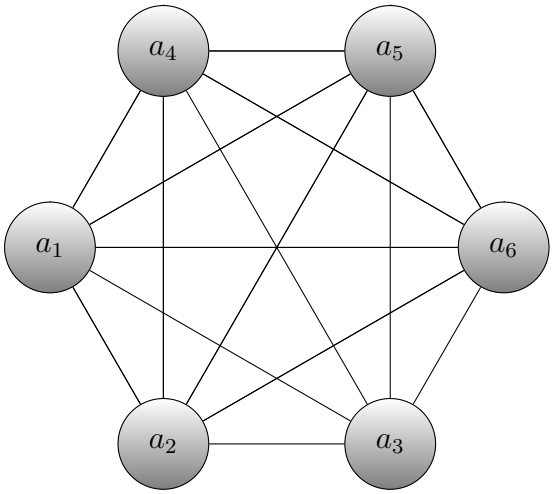


Figura 4.3: Grafo de Correlação de Abordagens.

que denota a relação entre as abordagens de detecção de `FFSN`, tal representação de custo demanda outro tipo de análise, pois cada abordagem selecionada possui características (custo e acurácia) diferentes.

As abordagens ao serem modeladas em um grafo como vértices e conectadas por arestas podem apresentar dois cenários. O primeiro considera que se o custo da abordagem for mapeado diretamente para aresta, então a modelagem deve tratar as distâncias como ($d_{ij} \neq d_{ji}$), o que pela teoria denotaria um Problema do Caixeiro Viajante Assimétrico (PVCA). Vale notar que a diferença entre o PVC e o PVCA está na representação da distância, a qual é inversamente proporcional ao PVC ($\frac{1}{d_{ij}}$) e o movimento das formigas pode mudar (Dorigo et al., 1996; Dorigo e Stützle, 2004).

No segundo cenário, os custos das abordagens permanecem associados aos vértices do grafo ao invés das arestas. Para entender esse modelo considere um grafo hipotético cujos os vértices são representados por $\{A, B, C, D\}$ e seus respectivos valores sejam $\{2, 3, 4, 5\}$, como ilustra a Figura 4.4. Partindo de qualquer vértice inicial, o resultado final será sempre o mesmo, pois o custo da soma dos vértices selecionados (*caminho*) segue o princípio da comutatividade.

Embora a modelagem inicial possa resultar em definições distintas, vale lembrar que as abordagens podem compartilhar comportamentos e características semelhantes como atributos (e.g: total de IP e total de ASN), algoritmos e consultas aos protocolos de rede como `DNS` e `Whois`. Portanto, é razoável assumir que existe uma intersecção em parte dos custos entre as abordagens selecionadas e, por essa razão, é possível abusar da notação para que o custo das abordagens seja distribuído ao longo da aresta, resultando num grafo semelhante ao modelo usado no PVC.

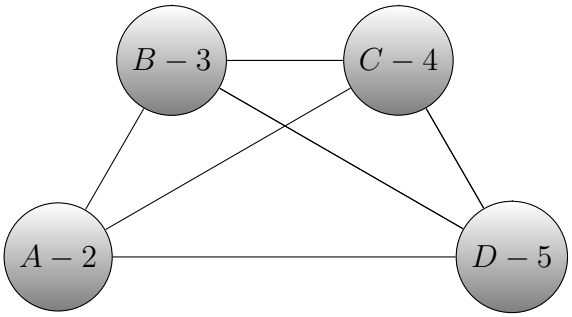


Figura 4.4: Grafo de Correlação de Abordagens.

Para incorporar os custos entre as abordagens na aresta, o cálculo da Distância Euclidiana foi adotado. A seção 4.2.2 apresenta mais detalhes da operação citada.

Dado os critérios acima, seja $G = (V, E)$ o grafo não direcionado que representa as relações entre as abordagens de detecção de rede, onde V e E são os conjuntos de vértices e arestas de G, respectivamente. Seja $H = \{h_1, h_2, \ldots, h_n\}$ o total de n abordagens escolhidas, e $F = \{f_1, f_2, \ldots, f_m\}$ o total de m características utilizadas por todas abordagens. Assim uma abordagem selecionada $a' = (h', f')$, onde $h' \in H$ e $f' \in F'$, com $F' \subset F$, denota o par ordenado da *abordagem* com seus respectivos *atributos* utilizados no processo de classificação. Dado H e F, então o conjunto dos vértices V é formado pela união dos conjuntos H e F, $V = (H \cup F)$ respectivamente. Seja u e $w \in V$, então $e = (u, w)$, com $e \in E$, denota a aresta que liga abordagens para detecção.

Vale acrescentar que cada abordagem selecionada deve possuir um nível de acurácia de detecção e o custo total dos atributos, os quais são utilizados como parâmetros na heurística (η) de seleção de abordagens. A heurística e o feromônio são utilizados no cálculo da *Probabilidade de Seleção de Abordagens*. Esse cálculo probabilístico permite selecionar as abordagens considerando suas características.

4.2.2 Representação das Abordagens

Neste trabalho, as abordagens são definidas como modelos formais que dependem de representações do comportamento (*vetor de características*) de um nome de domínio para determinar se este último é legítimo ou malicioso.

Modelos formais podem ser algoritmos de aprendizado de máquina como algoritmos supervisionados, não supervisionados, híbridos, assim como funções lineares ou heurísticas. Além disso, um modelo formal deve fornecer *Custo Total dos Atributos* (`CTA`), Acurácia de detecção (Δ) e o vetor de atributos. A Tabela 4.2 ilustra um exemplo de como as

abordagens podem ser representadas para serem inseridas no Grafo de Correlações de Abordagens.

Tabela 4.2: Modelo representativo de abordagens.

ID	Método	CTA	Acurácia	Atributos
1	Árvore de Decisão	144	89%	$\{f'_1, f'_2, f'_3\}$
2	Naïve Bayes	339	91%	$\{f'_1, \ldots, f'_6\}$
3	Função Linear	85	78%	$\{f'_1, \ldots, f'_5\}$

Nos exemplos citados, o cálculo do CTA é obtido a partir do tempo total que uma abordagem precisa para extrair todas as características de um nome de domínio. Para encontrar o valor do CTA, uma *janela de monitoramento* w pode ser estabelecida em intervalos onde ocorrem extração e análise de dados. Diferentes trabalhos na literatura estipulam que um intervalo de 5 minutos é apropriado para realizar esses procedimentos. No entanto, no caso da detecção de redes de serviço de fluxo rápido, esse valor pode variar devido ao tempo exigido para que as informações armazenadas localmente no resolvedor de nomes expire.

Para apresentar um exemplo simples do cálculo do CTA, considere uma janela de monitoramento $w = 1h$, dividida em intervalos de 5 minutos. A abordagem 1, ilustrada na Tabela 4.2, é baseada em árvore de decisão e possui 3 atributos. Seja f'_1 o total de endereços IP associados a um nome de domínio. Seja f'_2 a quantidade única de redes da classe C ($a.b.c$) e f'_3 a dispersão de valores em f'_2 cálculada a partir da entropia.

O custo de f'_1 após o término da janela de monitoramento é 48, pois durante a janela w foram enviadas 12 consultas DNS, onde cada consulta enviada tem o custo 1. O mesmo princípio pode ser aplicado para f'_2 e f'_3, resultando em um CTA total de 144, já que as 3 características foram coletadas durante w. É importante notar que o cálculo de f'_3 pode ser realizado ao final do processo de coleta, pois esse atributo depende dos valores obtidos em f'_2. Diante desse contexto, o cálculo do CTA pode ser formalmente definido como a *soma de todos os custos por atributo* durante uma janela de monitoramento, conforme ilustra a Equação 4.1 e a Definição 4.1.

$$\sum_{i=1}^{n} \phi(f'_i),\ \{f'_i | f' \in F\} \tag{4.1}$$

Definição 7 *ϕ é a função que representa o tempo total necessário para coletar os dados de determinada característica.*

Para encontrar o percentual de acurácia da abordagem selecionada, é realizado um procedimento de treino e teste em uma base de dados. Neste trabalho, a base de dados

foi construída observando os atributos de rede utilizados por todas as abordagens. O processo de extração de característica é realizado por *Agentes de Monitoramento* e seu funcionamento é melhor descrito na Seção 4.5. A partir do momento que a modelagem de todas as abordagens foi concluída, o processo de correlação e geração do modelo de classificação pode ser realizado.

Segundo (Zhou, 2015), a extração e o estudo de diferentes características utilizadas por abordagens distintas demonstram que técnicas de detecção de rede de serviço de fluxo rápido ainda não conseguem resolver esse comportamento malicioso, por essa razão, tal premissa reforça a importância no desenvolvimento do algoritmo para combinar essas abordagens, demonimado como `AFFACO-M`.

4.3 Algoritmo AFFACO-M

O algoritmo `AFFACO-M` tem como objetivo correlacionar e gerar modelos de classificação de redes de serviço de fluxo rápido seguindo critérios definidos pela restrição do problema. Uma solução ingênua para gerar modelos é ordenar as abordagens de classificação seguindo algum critério de ranqueamento como maior acurácia ou menor custo. Na literatura, as maiores acurácias estão associadas com longos períodos de monitoramento ou dependem de um número significante de características. Nesses casos o tempo de classificação é proporcional ao tempo de coleta de dados (Berger et al., 2016). Da mesma maneira, abordagens com menor custo podem ser evadidas a partir da manipulação de suas características (Knysz et al., 2011). Por essas razões, o algoritmo de correlação de abordagens `AFFACO-M` foi concebido considerando que o modelo a ser criado deve estabelecer um equilíbrio entre menor custo e maior acurácia, garantindo que um domínio seja observado a partir de diferentes perspectivas.

O funcionamento do algoritmo `AFFACO-M` é semelhante ao comportamento de qualquer algoritmo baseado em Otimização por Colônia de Formigas. Portanto, o algoritmo proposto contempla as etapas de *Construção de Regras*, *Poda de Soluções* e *Atualização de Feromônio*.

Nos passos iniciais do algoritmo `AFFACO-M`, descritos nas linhas $1-2$, é preciso definir as abordagens que serão utilizadas durante a correlação e a quantidade de feromônio inicial. A função `InitModels` *i*) cria o grafo de correlação de abordagens G, *ii*) posiciona aleatoriamente uma quantidade m de formigas, onde $m \geq |V|$, em cada vértice permitindo que todas as abordagens selecionadas sejam correlacionadas, e *iii*) armazena os valores da acurácia das abordagens em estruturas que são consultadas posteriormente.

Algoritmo 3 AFFACO-M

```
1: InitModels()
2: InitPheromone()
3: while condições não forem verdadeiras do
4:    BuildRules()
5:    ValidateRule()
6:    UpdateFeromônio()
7: end while
```

A função `InitPheromone` atribui um valor inicial de feromônio τ para cada aresta que conecta os vértices do grafo G. O valor do feromônio representa a atratividade ou intensidade no caminho entre os vértices. Esse valor é avaliado juntamente com a heurística η, a qual será descrita com mais detalhes na seção 4.3.1. O valor inicial do feromônio é definido pela seguinte operação $\tau = \frac{m}{\sum \Gamma}$, onde m denota o total de formigas no grafo e $\sum \Gamma$ denota a soma do total de características utilizadas por cada abordagem selecionada.

Vale lembrar que o feromônio é utilizado pelas formigas como mecanismo de comunicação no meio ambiente, por isso, é importante definir um valor que permita que tal propriedade contribua com o processo de convergência do sistema para uma solução ótima. Por exemplo, um possível valor inicial é 0.5. No entanto, esse parâmetro pode ser ajustado de acordo com o problema. O Capítulo 5 apresenta mais detalhes sobre esse valor.

As linhas $3-7$ descrevem a principal lógica do algoritmo, isto é, enquanto as condições de parada não são satisfeitas, as abordagens continuam sendo correlacionadas. Para que o algoritmo seja interrompido, é necessário que as formigas convirjam para um único caminho ao longo do processo, tal procedimento também é conhecido como estagnação (Dorigo e Stützle, 2004). Outra situação para finalizar a execução do algoritmo é quando o número de iterações das formigas tenha alcançado um limiar de parada. O objetivo é evitar que na ausência de convergência de um caminho, o algoritmo seja capaz de encontrar uma sequência de cidades abordagens para avaliação.

A função `BuildRules`, exibida na linha 4, representa o procedimento de correlação das abordagens, onde cada formiga avalia as possíveis cidades que podem ser visitadas. Cada abordagem selecionada para visitação faz parte de um conjunto de *soluções parciais* que tentam resolver o problema de classificação no menor tempo e maior acurácia. A formiga termina a correlação de abordagens quando todas as cidades acessíveis forem visitadas e um caminho tenha convergido, resultando assim, numa possível regra (sequência de abordagens) que poderá ser utilizada para classificação de domínios.

Para que a regra gerada seja utilizada, é necessário fazer uma validação para saber se o caminho escolhido pela formiga respeita as restrições do problema, isto é, menor custo e maior acurácia. Caso a regra contemple as restrições, o caminho escolhido é marcado com feromônio artificial para indicar a outras formigas que aquele caminho fora visitado. Os processos de validação de regra e marcação de feromônio são realizados pelas funções `ValidateRule` e `UpdateFeromônio`, respectivamente nas linhas 5 e 6. Ao final do algoritmo, aquele caminho que tiver convergido primeiro (maior número de marcações) é o escolhido. Além disso, dado a modelagem descrita acima, a regra gerada apresenta um modelo de custo crescente. Em outras palavras, abordagens com menor custo possuem maior probabilidade de serem escolhidas primeiro, enquanto as abordagens de maior custo são adicionadas posteriormente.

Vale notar que o algoritmo proposto pode ter seus parâmetros modificados para atender outras restrições. Por exemplo, caso deseje-se gerar regras menos restritivas em relação ao custo, é possível relaxar a função que avalia o custo da incorporação de uma abordagem. O Capítulo de Resultados apresenta mais detalhes sobre esse procedimento de validação e fornece informações relevantes quanto a operação do algoritmo `AFFACO-M`. Nas seções seguintes são apresentadas as etapas do algoritmo de correlação `AFFACO-M` de construção da regra, validação das regras e atualização de feromônio.

4.3.1 Construção e Validação de Regras

No processo de criação de regras (`BuildRules`), cada formiga k utiliza um mecanismo artificial de seleção de abordagens para visitar os vértices do grafo G. Para construir parte de uma regra (visitar uma abordagem), a *Probabilidade de Seleção de Abordagens* é calculada para cada abordagem que se deseja visitar, conforme descrito na Equação 4.2. De maneira simples, essa equação pode ser lida como a probabilidade da formiga k, localizada na abordagem i, visitar a abordagem j no instante t.

$$p_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{l \in \mathcal{N}_i^k}[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}, \text{if } j \in \mathcal{N}_i^k \end{cases} \tag{4.2}$$

Onde:

- k representa a k-ézima formiga;
- η_{ij} é o valor da heurística da Abordagem$_j$ quando acessada partir de i;
- $\tau_{ij}(t)$ é o montante de feromônio disponível (no instante t) na posição i,j;

- $\mathcal{N}_i^k$ são os vizinhos da abordagem$_i$ não visitados pela formiga k;
- α denota o nível de influência do feromônio;
- β representa o nível de influência da heurística.

Os parâmetros α e β permitem controlar o comportamento de escolha das formigas, por exemplo, se $\alpha = 0$, o feromônio é ignorado e a abordagem que tiver maior acurácia e menor quantidade de atributos será escolhida. No entanto, se $\beta = 0$, apenas o feromônio será considerado na seleção de abordagens, resultando em estagnação de caminhos da formiga (Dorigo e Stützle, 2004).

O objetivo da heurística η é identificar abordagens que permitam classificar um domínio obtendo o maior nível de acerto com base na acurácia (Δ) e menor esforço, onde esforço é o *Custo Total de Atributos* (`CTA`), como ilustra a Equação 4.3. É importante notar que acurácia nesse cenário é a execução da abordagem utilizando suas definições originais (parâmetros e pesos) considerando a base de treino definida neste trabalho.

$$\eta_{ij} = \left\{ \frac{\Delta}{\texttt{CTA}} \right. \tag{4.3}$$

Uma vez decidida a abordagem vencedora, a formiga muda de posição e armazena o nó visitado em um conjunto de *abordagens percorridas*, o qual é definido no momento de criação de cada formiga. Para (Dorigo e Stützle, 2004), esse conjunto também pode ser definido como *tour* da formiga.

O processo de seleção de abordagens continua até que essa formiga tenha percorrido todos os vértices do grafo G. Por exemplo, assumindo que a formiga k tenha percorrido 10 abordagens do grafo G, o conjunto de resultados terá valores do tipo $\{a'_7, a'_5, a'_6, a'_4, a'_8, a'_9, a'_{10}, a'_1, a'_2, a'_3\}$, onde a'_i denota a *i-ézima* abordagem selecionada pela formiga k, e a ordem de armazenamento nesse conjunto de resultados, denota a sequência de abordagens que podem ser utilizadas para classificação do nome de domínio ou os vértices que terão o feromônio atualizado.

4.3.2 Validação da Regra

Após a finalização da regra pela formiga k, é preciso avaliar se o caminho gerado está de acordo com as restrições do problema. Esse processo é semelhante a poda de regras, isto é, regras que não contribuem para solução do problema são descartadas. Neste trabalho, a função `ValidateRule` verifica se a soma total do `CTA` das abordagens é menor ou igual ao limite da `Restrição de Custo da Regra`.

Para entender o uso da `Restrição de Custo da Regra` para validação da regra, considere a Figura 4.5. Quando um nome de domínio é classificado, o pior cenário acontece quando *todas as abordagens* são necessárias durante esse processo, isto é, todos os atributos e abordagens da *Regra Completa* devem ser utilizados. O problema nessa situação é que a regra completa possui um *custo fixo*, independente da ordem em que é executada. Por outro lado, considerando o melhor cenário, é possível estabelecer certo nível de confiança, seguindo alguns critérios, que permitem reduzir o custo de classificação de um domínio suspeito.

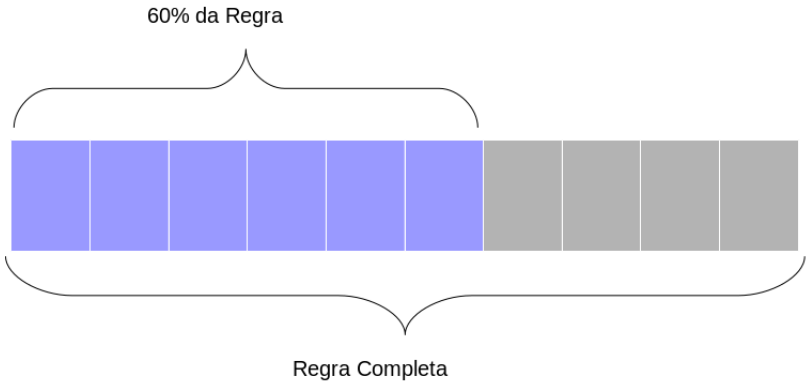


Figura 4.5: Exemplo conceitual da Restrição de Custo da Regra (Fonte: Próprio Autor).

Neste trabalho, o parâmetro `Restrição de Custo da Regra` é uma estratégia utilizada pelas formigas durante a seleção das abordagens que farão parte regra (modelo) de avaliação do nome domínio suspeito. É importante notar que esse parâmetro permite gerar regras considerando a relação de equilíbrio entre custo e acurácia. Sem tal parâmetro, as formigas geram regras onde as maiores acurácias serão escolhidas primeiramente. Esse comportamento acontece devido ao modelo probabilístico de seleção das abordagens que é formado pela razão da acurácia pelo número de atributos.

Para estabelecer esse equilíbrio, é necessário definir um limiar que auxilie na construção da regra. Como prova de conceito inicial do modelo proposto, para que uma regra seja considerada como válida, o valor do `CTA` das primeiras abordagens não deve ser superior à 60% do custo da total da *Regra Completa*. Na Figura 4.5, a Restrição do Custo da Regra foi aplicada para que a maioria simples das abordagens não tenham custo superior ao limite definido.

O objetivo dessa restrição é criar regras de classificação que possam determinar o comportamento de um nome de domínio suspeito logo no início da sequência de

abordagens. Além disso, um sistema de classificação que implementa essas regras pode tomar algum tipo de decisão como interromper o processo de extração de características.

Por exemplo, considerando ainda a regra $\{a'_7, a'_5, a'_6, a'_4, a'_8, a'_9, a'_{10}, a'_1, a'_2, a'_3\}$ gerada pela formiga k. Essa regra é valida se e somente se a soma do custos das abordagens $\{a'_7, a'_5, a'_6, a'_4, a'_8, a'_9\}$ for menor ou igual à `Restrição de Custo da Regra`. Caso a regra gerada esteja de acordo com os critérios descritos acima, a formiga k atualiza o feromônio de todo o caminho percorrido. Essa decisão permite que as formigas convirjam mais rápido, diminuindo assim o tempo de correlação de abordagens, pois apenas os melhores caminhos são atualizados com feromônio.

É importante dizer que a `Restrição de Custo da Regra` pode ser aplicada em outros cenários, por exemplo, um operador de rede deseja investigar um nome de domínio a partir de diferentes protocolos de rede em uma determinada sequência. Outro exemplo de restrição pode ser ilustrado, onde deseja-se includir abordagens pela acurácia e reduzir a intersecção entre atributos.

Independente da restrição do problema, as formigas precisam atualizar e evaporar porções do feromônio para que outras formigas também sejam atraídas pelos melhores caminhos.

4.3.3 Atualização do Feromônio

A finalização da construção das regras permite que cada formiga k atualize o total feromônio no caminho percorrido, através da função `UpdatePheromone`, listada na linha 6 do algoritmo de correlação `AFFACO-M`. O processo de *atualização do feromônio* permite que futuras formigas possam ser beneficiadas pelo caminho escolhido pela formiga k. A função `UpdatePheromone` realiza basicamente duas operações: evaporação e atualização de feromônio.

O *processo de evaporação* permite que o feromônio depositado em caminhos não utilizados sejam descartados com o tempo. A Equação 4.4 apresenta esse modelo para evaporar uma porção de feromônio *em todas as arestas* do grafo G a partir de um fator constante ρ, tal que é o coeficiente $(1 - \rho)$, sendo que os valores de ρ podem variar dentro do intervalo $0 < \rho \leq 1$, conforme definição inicial do algoritmo (Dorigo et al., 2000; Dorigo e Stützle, 2004; Parpinelli et al., 2002b).

$$\tau_{ij} \leftarrow (1 - \rho).\tau_{ij}, \forall (i, j) \in E, \tag{4.4}$$

Após o processo de evaporação, todas as formigas realizam o processo de *atualização* de feromônio nas arestas que fazem parte do caminho escolhido por cada formiga k, conforme

ilustra a Equação 4.5. No Capítulo 2, $\Delta\tau_{ij}^{k}$ é definido como $1/C^{k}$, onde C^{k} é o tamanho do caminho percorrido e m representa o número de vértices visitados pela formiga k. No entanto, neste trabalho, a atualização do feromônio deve dar prioridade para regras que tiveram maior acerto.

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{4.5}$$

A Equação 4.6 define as regras para atualização de feromônio considerando o nível de acurácia total que a regra encontrada pela formiga k obteve.

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{\sum Q}{|\Gamma|}, \text{ se aresta } (i,j) \in ant_{k}; \\ 0, \text{ caso contrário;} \end{cases} \tag{4.6}$$

onde:

- $\sum Q$ representa a soma de todas as acurácias escolhidas pela formiga k;
- Γ total de abordagens selecionadas pela formiga k.

A acurácia individual (Q) das abordagens é calculada pela função `InitModels`, na fase de pré-processamento e consultada para atualizar a qualidade da regra encontrada. É importante deixar claro que o emprego da média da acurácia é possível pois esse valor também está entre 0 e 1, onde 1 representa 100% de acerto. No entanto, dependendo do objetivo da classificação de nomes de domínios, é possível dar prioridades para outras métricas como a taxa de falso positivo ou *recall*.

Após o processo de atualização e evaporação das arestas do grafo G, caso as formigas tenham convergido para um único caminho, então essa sequência de abordagens denota um possível modelo que pode ser utilizado para classificar um nome de domínio suspeito.

Para determinar o comportamento dos nomes de domínios suspeitos, é necessário que um sistema de classificador adote as regras geradas pelo algoritmo `AFFACO-M` e consiga definir uma ação com base nos resultados obtidos na sequência de abordagens utilizadas. Para tanto, este trabalho apresenta o algoritmo `AFFACO-P`, o qual será descrito a seguir.

4.4 Sistema de Políticas AFFACO-P

O sistema de políticas `AFFACO-P` é o primeiro esforço de algoritmo que emprega as regras geradas pelo `AFFACO-M` para determinar o comportamento de redes de serviço de fluxo

rápido. Como prova de conceito, o funcionamento do AFFACO-P é baseado num sistema de votação pela maioria e pode ser dividido em dois momentos: *classificação antecipada* ou *classificação pelo último voto*.

Na *classificação antecipada*, o processo de rotulação de um nome de domínio passa por todos os estágios da regra gerada. Por exemplo, se a regra for constituída de 10 abordagens, então um nome de domínio será classificado primeiramente por 6 abordagens (60%), em seguida por 7, 8, 9 ou por todos os métodos selecionados. Todas as abordagens são tratadas de maneira equivalente, portanto os resultados de suas classificações possuem o mesmo peso. Em caso de empate em regras ímpares ou situações de dúvidas pelo classificadores, é necessário realizar a *classificação pelo último voto*.

O processo de classificação pelo último voto é uma estratégia adotada para tentar classificar o nome de domínio caso a maioria não tenha entrado num consenso. Por exemplo, para o conjunto de abordagens adotadas nesta tese de doutorado, existem situações onde os classificadores chegaram ao seguinte resultado: um domínio recebeu 3 votos positivos e 3 votos negativos. Nesse caso, os classificadores estão em dúvida, portanto mesmo que a sétima abordagem defina a natureza do nome de domínio, o comportamento ainda é suspeito.

A mesma premissa pode ser adotada em casos onde a regra possui tamanho par. Para ilustrar esse comportamento, considere um nome de domínio que recebeu a seguinte classificação $\{P, P, P, P, P, N, N, N, N, ?\}$, onde P = positivo (malicioso) e N = negativo (legítimo). Nesse ponto não houve consenso pela maioria, portanto, o décimo voto pode definir a classe de comportamento ou empatar o resultado. Por esses motivos, a classificação pelo último voto é necessária.

É importante deixar claro que a seleção das características e abordagens podem reduzir o uso de estratégias como último voto. No entanto, a intenção neste trabalho é demonstrar que o modelo de regra produzido pelo AFFACO-M pode ser tratado de inúmeras maneiras. Por exemplo, ao invés de utilizar as características em outros classificadores, o mecanismo do último voto pode ser o retorno das consultas enviadas aos sistemas de reputação como PhishTank e Google Safe Browsing.

4.4.1 Algoritmo AFFACO-P

O algoritmo AFFACO-P é um fusor de resultados e que é capaz de tomar decisões com base em políticas estratégicas. Como prova de conceito, esse algoritmo é baseado no conceito de votação simples para tomar uma decisão do comportamento do nome de domínio. As linhas $1 - 5$ do algoritmo AFFACO-P denotam as definições iniciais para

Algoritmo 4 AFFACO-P

```
1: Rule[] ← {a'_1, a'_2, ..., a'_n}
2: d ← domínio suspeito
3: Class_{P|N} ← 0
4: FullStop ← 60%.|Rule|
5: LastVote ← 50%.|Rule|
6: for each a' ∈ Rule[] do
7:   if Class_{P|N} ≠ FullStop then
8:     Class_{P|N} ← a'.classify(d)
9:     if |Class_{P|N}| = LastVote then
10:      Class_{P|N} ← BurdenMethod.classify(d)
11:      return Class_{P|N}
12:    end if
13:  else
14:    return Class_{P|N}
15:  end if
16: end for
```

o funcionamento desse sistema. A regra gerada pelo algoritmo `AFFACO-M` é atribuída à variável `Rule` (linha 1), um tipo de vetor unidimensional capaz de armazenar as abordagens na ordem em que serão utilizadas para classificar o domínio suspeito d, exibido na linha 2. A variável descrita na linha 4 (`FullStop`) é um limite de controle que sinaliza o fim da classificação pelo voto simples de determinada classe. Isto é, houve consenso entre as abordagens e um rótulo de comportamento (legítimo/malicioso) foi escolhido.

A linha 5 ilustra uma variável (`LastVote`) de controle que indica que a classificação do nome de domínio d teve empate, então sendo necessário um procedimento de desempate, denominado como `BurdenMethod`, exibido na linha 10.

A variável $\texttt{Class}_{\{P|N\}}$, linha 3, representa um contador de domínios que foram classificados como positivos (P) ou negativos (N) por cada abordagem definida em `Rule`. Os valores armazenados em $\texttt{Class}_{\{P|N\}}$ são utilizados para determinar o comportamento final de um nome de domínio, isto é, se a maioria das abordagens classificou d como positivo ou negativo. O processo de classificação toma como base que a maioria das abordagens (60%) tenha convergido para uma classe final seja P ou N. Além disso, quanto maior o valor de `FullStop`, maior será o custo para classificar um nome de domínio.

As linhas $6 - 16$ demonstram o processo de classificação de um domínio suspeito d. Cada abordagem em `Rule` é utilizada para classificar d, sendo que o resultado desse processo é armazenado em $\texttt{Class}_{\{P|N\}}$ como positivo ou negativo, respectivamente. Se as abordagens tiverem classificado majoritariamente d (linha 7), então o algoritmo `AFFACO-P` pode tomar uma decisão com base nesse resultado.

Caso ocorra algum tipo de empate de classificação (linha 9), onde o número de positivos e negativos é idêntico ao valor de `LastVote`, é necessário fazer outro tipo classificação mais criteriosa, denominado como `BurdenMethod`. Tal método pode ser qualquer estratégia necessária para classificar o nome de domínio. Neste trabalho, `BurdenMethod` é uma função que classifica d empregando todas as características utilizadas pelas abordagens selecionadas.

A utilização de todas as características parte do princípio que esses dados já foram extraídos do domínio por outras abordagens durante o processo de classificação, então no pior caso, é possível aglutinar todas as características em um único modelo representativo de comportamento. Desta maneira, tal modelo pode ser treinado e testado por algoritmos de aprendizagem de máquina, por exemplo. Para esse método, dois algoritmos são utilizados `Random Forest` e `Logistic Regression`. O Apêndice B apresenta mais detalhes sobre o funcionamento da função `BurdenMethod`.

A partir dos resultados obtidos pelo sistema de votação simples, o algoritmo `AFFACO-P` pode enviar uma mensagem na rede para que o nome de domínio suspeito seja bloqueado no *firewall* da rede.

É importante notar que a extração de características pode ocorrer juntamente com as abordagens selecionadas durante a classificação do nome de domínio. A extração de características é realizada pelo *Agente de Monitoramento*, descrito a seguir.

4.5 Agentes de Monitoramento

Agentes de Monitoramento são sistemas independentes utilizados para acompanhar o estado de um determinado nome de domínio durante diferentes janelas de monitoramento. A partir desse processo, é possível entender a dinâmica e identificar alterações no comportamento de domínios suspeitos.

Além das janelas de monitoramento, os Agentes de Monitoramento devem operar em diferentes regiões do globo incluindo redes e países distintos. Essa estratégia visa evitar restrições impostas por *botmasters* (Hu et al., 2011) e identificar a distribuição geográfica de endereços IP que respondem pelo nome de domínio. Por exemplo, quando o domínio `youtube.com` é consultado do Amazonas/Brasil, os seguintes endereços IP `201.57.54.X` localizados no Brasil são retornados. Por outro lado, o mesmo domínio consultado a partir de Frankfurt/Alemanha, apresenta os endereços da rede `172.217.20.X` localizados nos EUA. Em ambos os casos, as redes retornadas estão mais próximas da origem e o custo para acessar esses IP é em média de 10 saltos. Apesar da distância geográfica no

caso da Alemanha, a rede de distribuição de conteúdo do `Google` faz *peering* de redes para esse bloco de endereço IP para reduzir a carga de dados[5].

O monitoramento e extração de características de um nome de domínio deve acontecer de maneira simples e coordenada, pois o tratamento de exceções de dados é o maior desafio desse processo. Por exemplo, embora a comunicação entre cliente e servidor para o protocolo `Whois` seja trivial, o formato de armazenamento de dados não segue nenhum padrão. Desta forma, enquanto uma abordagem precisa da data de criação do nome de domínio, outra abordagem pode precisar do nome da empresa que oferece o serviço de registro de domínio. Em ambos os casos, o `TLD`, país e língua influenciam a maneira de extrair tais dados.

É possível fazer mesma observação para outros protocolos como `DNS` e `HTTP`. No caso do `DNS`, dependendo da configuração do servidor de nomes com autoridade sob o domínio, é necessário realizar consultas adicionais para que todas as seções da resposta `DNS` estejam completas. Tal exigência se faz presente, pois diversas abordagens utilizam características das três seções do protocolo incluindo *resposta* (`ANSWER`), *autoridade* (`AUTHORITY`) e *adicional* (`ADDITIONAL`).

Para o protocolo `HTTP`, é importante que o Agente de Monitoramento seja capaz de obter o conteúdo da página em um ambiente seguro tipo *sandbox*, pois algumas abordagens fazem esse tipo de solicitação para verificar o tempo de acesso de hosts que hospedam conteúdo em redes de serviço de fluxo rápido.

De maneira geral, a extração de características de domínios deve ser realizada de maneira cautelosa para que esse processo tenha êxito no menor tempo possível, visto que os domínios maliciosos possuem tempo de vida curto.

4.5.1 Algoritmo do Agente de Monitoramento

Neste trabalho, cada Agente de Monitoramento recebe uma notificação de domínio através de uma mensagem `HTTP`, o que permite que múltiplos agentes distribuídos ao redor do globo sejam notificados. Quando um Agente é notificado sobre um nome de domínio, o processo de extração de características é iniciado, conforme ilustra o Algoritmo 5.

As características do nome de domínio d são extraídas caso esse domínio ainda não seja conhecido ou previamente consultado, ou seja, $d \notin D$. Seja $\mathcal{F}$ o conjunto de todas as características que já foram extraídas ao longo do tempo e podem estar armazenados em algum tipo de base de dados. A função `ExtrairCaracterísticas` (linha 5) é o procedimento responsável pela extração das características do domínio monitorado d,

[5] `https://peering.google.com/#/options/peering`

Algoritmo 5 Algoritmo do Agente de Monitoramento

1: $d \leftarrow$ notificação via `API RESTful`
2: $\mathcal{D} \leftarrow$ {conjunto de domínios conhecidos}
3: $\mathcal{F} \leftarrow$ {conjunto de características extraídas}
4: **if** $d \notin \mathcal{D}$ **then**
5: $\quad \mathcal{D} \leftarrow \mathcal{D} \cup d$
6: **end if**
7: $\mathcal{F}_d \leftarrow$ `ExtrairCaracterísticas`(d)
8: `UpdateFeatures`$(d, \mathcal{F}_d)$
9: `Enqueue`(d)
10: `DomainTrack`(d)

onde a relação de atributos a serem extraídos pode ser repassada em uma requisição contendo uma estrutura de dados do tipo `JSON`.

A função `UpdateFeatures` (linha 7) armazena na base de dados $\mathcal{F}$ o vetor de atributos $\mathcal{F}_d$ com dados recém coletados. Essa abordagem tem como objetivo disponibilizar antecipadamente as características que podem ser coletadas no menor tempo para classificação e que não dependem de outras janelas de monitoramento.

Após o armazenamento de dados, o domínio d é adicionado na fila de monitoramento de características (linha 8), da qual é posteriormente processada pela função `DomainTrack`, respeitando o tempo de vida (`TTL`) de cada informação. Por exemplo, seja t o tempo de vida (`TTL`) associado ao nome de domínio d recém consultado. Novas consultas para esse domínio serão enviadas quando um sistema *temporizador* aguardar pelo tempo $t + 1$.

A função `DomainTrack` é responsável pelo acompanhamento dos nomes de domínios enfileirados (linha 9) para que continuem sendo monitorados. O monitoramento de nomes de domínio por longos períodos é adotado em vários trabalhos (Chen et al., 2013a; Hu et al., 2009; Lin et al., 2013), pois permite extrair características que são utilizadas por diferentes abordagens, fontes de dados e métricas.

A extração de características de um nome de domínio deve seguir alguns procedimentos de validação como verificação do *estado* ou *comportamento* do domínio. Por exemplo, seja d o nome de domínio a ser consultado e seja $q_d = 1$ o número de consultas DNS que retornaram algum tipo de erro como:

- Domínio inexistente (`NXDomain`);
- Falha de servidor (`SERVFAIL`);
- Domínio não possui registros do tipo autorizativos (`NS`);
- Domínio possui endereços IP inválidos (e.g: 127.0.0.1)

O agente de monitoramento exclui o nome de domínio d do processo de coleta de características, pois não é possível obter informações mínimas sobre esse domínio. Outro exemplo de exclusão de um nome de domínio acontece quando o *comportamento* do mesmo é conhecido (e.g: malicioso ou legítimo), isto é, o modelo de classificação gerado pelo `AFFACO` foi capaz de classificar o nome de domínio corretamente.

Além disso, é importante notar que se um nome de domínio expirar e, no futuro for registrado novamente para atividades maliciosas, os agentes de monitoramento descartarão esse domínio, pois o comportamento do mesmo ainda é conhecido. Para resolver esse problema, é preciso reavaliar a base de dados assim como seus atributos no entanto, tal procedimento está fora do escopo desse trabalho.

O Anexo D descreve mais detalhes sobre a metodologia utilizada para extrair as características utilizadas neste trabalhos com base na seleção das abordagens.

5

Resultados

Este Capítulo apresenta os resultados do processo de avaliação do *framework* proposto para classificação de redes de serviço de fluxo rápido. Para tanto, os componentes de correlação de abordagens e geração de regras (`AFFACO-M`) e o fusor baseado em sistema de votação pela maioria (`AFFACO-P`) foram utilizados. Além disso, 7 abordagens de detecção de redes de serviço de fluxo rápido foram desenvolvidas para que o algoritmo de correlação de abordagens fosse capaz de produzir regras de classificação.

As seguintes abordagens (Caglayan et al., 2009b), (Chen et al., 2013a), (Grzinic et al., 2014), (Passerini et al., 2008), (Holz et al., 2008) e (Huang et al., 2010) foram selecionadas pois são frequentemente citadas na literatura e, por apresentarem o maior número de características baseadas no tráfego `DNS`. Dada uma restrição da base de dados utilizada neste trabalho, a qual será descrita ao longo desse capítulo, outra abordagem de detecção de redes de serviço de fluxo rápido foi desenvolvida. A abordagem proposta, `EntropyFFNet`, é resultado de um processo de adaptação do trabalho proposto por Zhao e Jin (2015). Os detalhes da proposta `EntropyFFNet` e seus respectivos resultados preliminares podem ser observados no Apêndice A.

Para demonstrar como o algoritmo `AFFACO-P` pode ser utilizado na identificação de nomes de domínio suspeitos, duas novas abordagens de detecção foram desenvolvidas e fazem fazem parte da etapa `BurdenMethod`. Os resultados dessa etapa estão descritos no Apêndice B.

Os resultados contidos neste capítulo estão fundamentados na base de dados obtida durante a realização deste trabalho, a qual é composta exclusivamente de tráfego `DNS`,

no formato da ferramenta `DIG`[1]. Além disso, uma porção de consultas à base `Whois` foi necessária para completar as características utilizadas por algumas abordagens.

A análise dos resultados está dividida em duas etapas. Na primeira, é demonstrada uma visão geral de como os dados estão representados na base de dados. O objetivo aqui é apresentar as principais características utilizadas pelas abordagens, tais como a distribuição de endereços IP associados em domínios legítimos e maliciosos ou a quantidade de sistemas autônomos por tipo de domínio. Além disso, nesse estudo também é demonstrado que algumas características possuem certo tipo de relação quando um comportamento suspeito acontece.

Na segunda etapa, tanto o algoritmo de correlação `AFFACO-M` como o classificador `AFFACO-P` são avaliados individualmente. Por exemplo, no algoritmo de correlação `AFFACO-M` os valores dos componentes α e β sinalizam como as formigas podem ser influenciadas na escolha de uma próxima abordagem a ser visitada. Outros parâmetros desse algoritmo também são investigados com objetivo de identificar valores que podem contribuir positivamente com o processo de correlação de abordagens. A partir da seleção de parâmetros relevantes, o processo de correlação é capaz de gerar regras (sequência de abordagens) que demonstram equilíbrio entre menor custo e maior acurácia. Para o conjunto de abordagens selecionadas neste trabalho, é possível demonstrar que o algoritmo de correlação `AFFACO-M` é capaz de produzir diferentes tipos de sequência de abordagens, isto é, melhores regras ou regras que demandam os maiores custos.

As regras geradas possuem impacto direto no comportamento do algoritmo `AFFACO-P`. Por exemplo, caso uma sequência de abordagens seja gerada considerando exclusivamente os maiores níveis de acurácia, os custos para classificação de um domínio são tão altos que podem inviabilizar o processo de classificação. Ao mesmo tempo, é possível alcançar resultados semelhantes considerando a relação de custo vs acurácia, o que permite demonstrar que o modelo proposto pelo `AFFACO-M` consegue encontrar num espaço de busca, uma solução viável para detectar redes de serviço de fluxo rápido.

Os resultados a seguir mostram que a sequência de regras geradas pelo algoritmo `AFFACO-M` e adotadas pelo sistema de política `AFFACO-P` alcançam 93.56% de acurácia e, que apenas uma porção das abordagens são necessárias para classificar um nome de domínio.

As seções abaixo estão organizadas da seguinte maneira. Primeiro, o protocolo experimental é descrito, em seguida os resultados são apresentados.

[1]DIG: utilitário utilizado para consultar nomes de domínio.

5.1 Protocolo Experimental

Durante o desenvolvimento desse trabalho, diferentes passos foram realizados para construção dos resultados, como ilustra a Figura 5.1. Primeiro, é realizada a caracterização dos atributos associados aos nomes de domínios legítimos e maliciosos. A ideia é demonstrar um estudo comparativo que forneça uma visão geral de como as características de rede são utilizadas em ambos os cenários, reforçando a premissa as redes de serviço de fluxo rápido devem ser analisadas por diferentes ângulos.

Em seguida, cada abordagem individual selecionada neste trabalho é validada. A validação das abordagens tem como objetivo verificar o desempenho de classificação dos nomes de domínios contidos na base de dados e o correto funcionamento dos algoritmos utilizados nesses trabalhos. Ainda nesta etapa, é demonstrado o conjunto de tratativas que foram realizadas para que algumas abordagens pudessem ser incluídas no processo de correlação de abordagens.

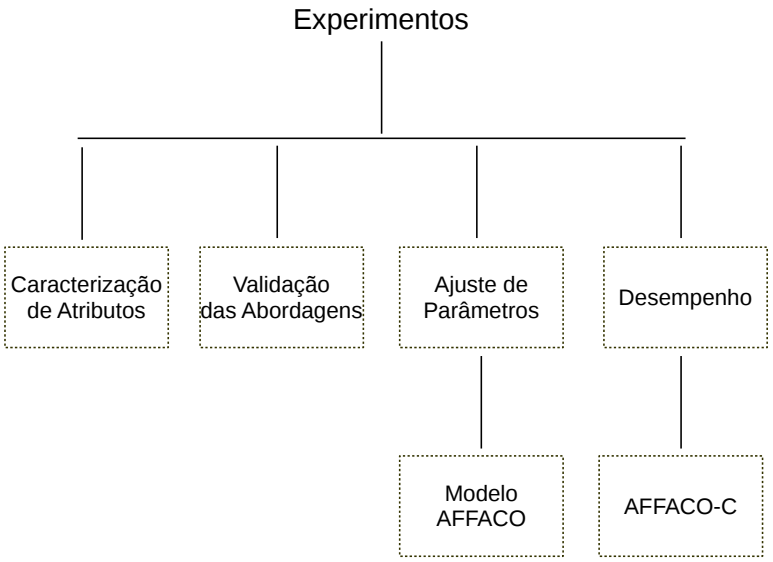


Figura 5.1: Etapas Realizadas no Protocolo Experimental.

No terceiro passo, os parâmetros utilizados pelo algoritmo de correlação de abordagens `AFFACO-M` são analisados para ilustrar que diferentes valores exercem influência direta no processo de geração das regras de classificação. Além disso, nesse estudo são definidos os valores que são empregados nesse processo e que resultaram nas regras descritas na seção de análise de desempenho.

Finalmente, na análise de desempenho, o sistema de votação AFFACO-P é analisado durante investigação de domínios suspeitos. Nesse cenário são demonstrados exemplos de classificação baseados nas regras geradas pelo algoritmo de correlação AFFACO-M.

Para cada abordagem selecionada, includindo o próprio algoritmo AFFACO-P, um conjunto de métricas é aplicado. O objetivo é entender o comportamento dos classificadores a partir de diferentes avaliações. A Tabela 5.1 apresenta a *Tabela de Confusão* que permite visualizar o desempenho geral dos classificadores.

Tabela 5.1: Tabela de Confusão

	+R	**-R**
+P	TP	FP
-P	FN	TN

- Verdadeiro Positivo (TP): especifica o número de domínios legítimos que foram classificados corretamente;
- Verdadeiro Negativo (TN): especifica o número de domínios maliciosos que foram classificados corretamente;
- Falso Positivo (FP): denota o número de domínios maliciosos que foram classificados incorretamente;
- Falso Negativo (FN): métrica que define o número de domínios legítimos que foram classificados incorretamente.

Além da Tabela de Confusão, as seguintes métricas são utilizadas para avaliar as abordagens individuais e o classificador AFFACO-P:

- Recall (Sensitividade) é a divisão entre o número de positivos verdadeiros dividido pela soma de positivos verdadeiros e falsos negativos portanto, está relacionado a quantos resultados aplicáveis são retornados, conforme ilustra a Equação 5.1.

$$\frac{\texttt{TP}}{\texttt{TP + FN}} \tag{5.1}$$

- Especificidade é proporção de casos negativos de fato que foram corretamente classificados como negativos, como ilustra a Equação 5.2.

$$\frac{\texttt{TN}}{\texttt{FP + TN}} \tag{5.2}$$

- Acurácia é medida que avalia quanto que um classificador foi capaz de rotular corretamente como positivo e negativo.

$$\frac{\texttt{TN + TP}}{\texttt{TP + TN + FP + FN}} \tag{5.3}$$

- Erro de Classificação representa o montante que não foi classificado corretamente como positivo ou negativo. É a diferença da acurácia, como ilustra a Equação 5.4.

$$1 - \text{acurácia} \tag{5.4}$$

- Taxa de Falso Positivo é a proporção entre o número de casos negativos reais classificados equivocadamente como positivos e o número total de eventos negativos reais.

$$\frac{\texttt{FP}}{\texttt{FP + TN}} \tag{5.5}$$

- Precisão é a proporção de casos rotulados como positivos que são verdadeiramente positivos, como ilustra a Equação 5.6

$$\frac{\texttt{TP}}{\texttt{TP + FP}} \tag{5.6}$$

- Medida-F (*F-Measure*) é a media harmônica ponderada que leva em consideração os valores em `Recall` e `Precisão`, como ilustra a Equação 5.7.

$$\texttt{F}_1 = \frac{2}{\frac{1}{\texttt{Recall}} + \frac{1}{\texttt{Precisão}}} \tag{5.7}$$

- AUC denota a área abaixo da curva ROC que é formada pela combinação entre a Taxa de Falso Positivo vs Taxa de Verdadeiro Positivo, conforme ilustra a Equação 5.8.

$$\texttt{AUC} = \frac{\texttt{Recall + Especificidade}}{2} \tag{5.8}$$

Para demonstrar a capacidade de classificação utilizando as regras geradas pelo algoritmo de correlação, o algoritmo `AFFACO-P` é avaliado observando 3 cenários. O primeiro cenário tem como objetivo classificar os domínios da base de dados a partir da regra formada pelo *menor custo de atributos*. No segundo momento, os domínios são classificados a partir de uma regra que possui a *maior acurácia*. Finalmente, no terceiro cenário, os domínios são avaliados pela regra gerada pelo algoritmo de correlação `AFFACO-M`, da qual foi combinada considerando a relação *menor custo e maior acurácia*.

É importante notar que a classificação é realizada por cada abordagem utilizada na regra, então o papel do algoritmo `AFFACO-P` é tomar uma decisão a a partir desses resultados.

Para identificar essas regras produzidas, foram geradas manualmente `5.040`, ou seja, 7! combinações de sequências de abordagens. O objetivo desse procedimento é *i*) obter uma visão geral das regras geradas pelo algoritmo de correlação `AFFACO-M`; *ii*) identificar as regras que possuem o maior e o menor custo; e *iii*) analisar a proximidade da regra correlacionada pelas formigas em relação as regras manuais.

É importante ressaltar que o processo manual de geração de regras é possível neste trabalho porque a quantidade de abordagens selecionadas não representa um custo considerável de combinações. No entanto, por exemplo, para 20 abordagens existem 2.43^{18} (20!) possíveis regras a serem geradas, o que torna esse procedimento demorado e inviável para análise.

O principal objetivo nesses cenários é mostrar que para base de dados e abordagens selecionadas neste trabalho, é necessário existir um *equilíbrio* entre custo vs acurácia. Caso contrário, pode-se acreditar que o processo de correlação do `AFFACO-M` pode ser substituído por uma regra da qual, os menores custos ou maiores acurácias estão posicionadas de tal maneira que um sistema de votação pela maioria alcançaria melhores resultados.

Vale ressaltar que os algoritmos das abordagens selecionadas neste trabalho foram implementados utilizando as informações fornecidas pelos autores. Por outro lado, cada componente do algoritmo de correlação `AFFACO-M` foi avaliado para identificar seu impacto tanto na correlação de abordagens como na geração de regras.

5.1.1 Parâmetros na Correlação de Abordagens

As regras foram geradas inicialmente pelo algoritmo de correlação `AFFACO-M` considerando os valores padrões definidos por (Dorigo e Stützle, 2004), onde $\alpha = 0.5, \beta = 0.5$ e $\rho = 0.5$. Em seguida, foram realizadas inúmeras modificações em cada componente desse algoritmo para avaliar a influência na regra gerada, assim como no processo de correlação de abordagens. Tal estratégia permitiu identificar valores que produziram resultados mais significativos quando comparados com os parâmetros determinados originalmente. No processo de *Ajuste de Parâmetros*, descrito na seção 5.1.1, os seguintes parâmetros foram analisados:

- α: influência do feromônio;
- β: influência da heurística;
- ρ: fator de atualização / evaporação do feromônio;

- `Quantidade Formigas`: número de formigas inseridas no Grafo de Correlação G;
- `Número de Iterações`: número que podem ser realizados correlações antes de interromper o algoritmo;
- `Restrição de Custo da Regra`: permite que as formigas busquem diferentes combinações dentro da restrição do problema.
- `Convergência`: número de vezes que um caminho deve ser percorrido para sinalizar que as formigas estagnaram e não há mais nada a ser explorado.

A `Quantidade Formigas` é um valor que depende da base de dados e modelo definido. O valor do `Número de Iterações` pode ser um valor fixo que oferece meios para que o algoritmo termine caso as formigas não consigam convergir. A `Convergência` é um mecanismo (valor inteiro) que permite sinalizar para o algoritmo de correlação `AFFACO-M` que as formigas encontraram um caminho significativo, do qual será utilizado para classificar redes de serviço de fluxo rápido. O valor de `Restrição de Custo da Regra` pode definir se a regra gerada é tendenciosa (valores muito baixo), abrangente (valores alto) ou equilibrada.

É importante deixar claro que, embora exista certa semelhança entre a restrição de 60% na criação da regra durante o processo de correlação de abordagens e o limiar no sistema de votação no algoritmo `AFFACO-P`, é possível que o `AFFACO-P` siga outras políticas para o sistema de votação. A mesma analogia também pode ser estendida ao gerador de regras `AFFACO-M`, onde a restrição do menor custo pode ser substituída pela escolha de abordagens baseadas no tráfego `DNS` seguidas de soluções que usam o tráfego `HTTP`.

Custo da Regra

Para entender os resultados ilustrados na avaliação de desempenho do *framework* `AFFACO`, o conceito do custo da regra deve ser bem definido. Como demonstrado no Capítulo 4, a soma de todos os custos (`CTA`) das abordagens é fixo. Somando o total dos custos entre A_1 até A_7, o valor de 1353 é encontrado, isto é, esse valor representa o custo total para que todas as abordagens selecionadas neste trabalho classifiquem *um* único nome de domínio.

A `Restrição de Custo da Regra` (60%) permite que diferentes combinações de abordagens aconteçam no início da sequência da regra de classificação. O valor 811.8 representa 60% do total de todos os `CTA` utilizados neste trabalho. Portanto, o algoritmo de correlação `AFFACO-M` tenta gerar uma regra da qual o custo inicial é menor ou igual a 60% da soma de todos os `CTA`.

Vale observar que no caso de valores abaixo de 60%, o algoritmo de correlação reduz seu espaço disponível para combinar diferentes abordagens. Por outro lado, se o limite for relativamente superior, as regras geradas são abrangentes e a tendência é que as abordagens com maior acurácia sejam adicionadas primeiro.

Diante desse contexto, o custo da regra descrito na seção de análise de desempenho trata do limite de 60% imposto neste trabalho para que seja possível gerar regras que tenham equilíbrio entre custo e acurácia.

Custo de Classificação

Além das métricas definidas na literatura para análise de desempenho do algoritmo `AFFACO-P` também é avaliado pela quantidade de abordagens utilizadas para classificar um nome de domínio. Neste trabalho, o `AFFACO-P` depende do consenso da maioria para determinar a classe de comportamento de um domínio.

Por exemplo, considere um cenário onde foram geradas duas regras a partir de um conjunto de 10 abordagens de detecção de redes de serviço de fluxo rápido. Seja R_1 uma regra hipotética cujo o custo total denota 1000 pontos e o valor da restrição de 60% representa 600 pontos. Seja R_2 uma regra hipotética cujo o custo da restrição de 60% denota 400 pontos.

Tabela 5.2: Custo da regra R_1.

QtdAbordagens	Domínios Classificados	Custo por classificação	TP	TN	FP	FN
6	11	600	10	1	0	0
7	9	700	0	9	0	0
8	0	800	0	0	0	0
9	0	900	0	0	0	0
Total	**20**	**12900**				

Tabela 5.3: Custo da regra R_2.

QtdAbordagens	Domínios Classificados	Custo por classificação	TP	TN	FP	FN
6	10	400	9	1	0	0
7	9	500	0	9	0	0
8	1	600	0	1	0	0
9	0	900	0	0	0	0
Total	**20**	**9100**				

Assumindo que uma base de dados hipotética seja composta de 20 nomes de domínios (10 legítimos e 10 maliciosos), então R_1 e R_2 podem apresentar os seguintes resultados

como ilustram as Tabelas 5.2 e 5.3, respectivamente. A regra R_1 classificou 11 domínios utilizando as primeiras 6 abordagens a um custo de $11 * 600 = 6600$ e 9 domínios foram classificados usando 7 abordagens a um custo de $9 * 700 = 6300$. Portanto, o custo de classificação da regra R_1 é 12900 pontos, enquanto que R_2 denota 9100 pontos. Para esse caso hipotético, a regra R_2 possui o menor custo de classificação

Os experimentos descritos neste trabalho foram executados em um servidor executando o sistema operacional `Debian GNU/Linux`, 96GB de memória RAM, com 2 processadores `Intel(R) Xeon(R) E5649 - 2.53GHz` contendo 6 núcleos e 12 *threads*.

5.1.2 Base de Dados

Os nomes de domínios foram obtidos e avaliados a partir de duas bases de dados: legítima e maliciosa. A base legítima foi construída a partir de consultas DNS enviadas para os primeiros 30 mil domínios encontrados no `Alexa.com`. A base maliciosa[2] é formada por nomes de domínios *coletados* e *rotulados* por outros trabalhos de classificação de `FFSN` incluindo (Passerini et al., 2008), (Huang et al., 2010) e `ATLAS`[3]. Os trabalhos em (Passerini et al., 2008) e (Huang et al., 2010) utilizados para rotular a base de dados também foram implementados neste trabalho.

A base de dados maliciosa é composta por inúmeros arquivos texto contendo respostas `DNS`, obtidas pelo utilitário `DIG`. Esses arquivos textos contêm nomes de domínio que foram observados durante o mês de março de 2009 e, após pré-processamento, o conteúdo foi armazenado em um sistema de gerenciamento de banco de dados relacional.

O processo de pré-processamento incluiu a validação da integridade dos dados e extração de características. Por exemplo, na validação de integridade foi possível identificar que em diversos casos, as seções de resposta `DNS ANSWER`, `AUTHORITY` e `ADDITIONAL` não estavam presentes. Neste trabalho, um nome de domínio é utilizado se e somente se todas as seções de resposta `DNS` são válidas. Tal decisão permitiu selecionar 485 nomes de domínios maliciosos para análise.

Para confecção da base de dados legítima, as consultas `DNS` foram enviadas por uma ferramenta desenvolvida durante a realização deste trabalho para garantir que as três seções da resposta `DNS` estavam presentes. As requisições de rede foram enviadas para cada nome de domínio durante cinco dias, em intervalos de 5 minutos ou até que o `TTL` tivesse expirado. Para evitar possíveis desbalanceamentos entre as bases de dados, foi

[2] `https://sites.google.com/site/huangpublication/datasets/-1-fast-flux-attaack-datasets`
[3] `https://www.arbornetworks.com/atlas-portal`

estabelecido que número de instâncias da base legítima fosse próximo da base maliciosa, como ilustra a Tabela 5.4

Tabela 5.4: Base de dados utilizada durante o processo de classificação.

Base	Número Domínios	Ano
Legítima	500	2017
Maliciosa	485	2009

Após a construção da base de dados, foi criado um vetor de características contendo 22 atributos para cada uma das 985 instâncias, onde o rótulo da classe de domínios legítimos é marcado com valor positivo (`+1`), enquanto o valor `-1` é atribuído para os domínios maliciosos.

5.2 Resultados

5.2.1 Caracterização dos Atributos

Esta seção apresenta um estudo sobre os atributos de rede utilizados tanto por domínios maliciosos como legítimos. A Tabela 5.5 apresenta a quantidade de endereços IP únicos (60.518), o percentual de endereços IP que são compartilhados e o total de países observados na base de dados.

Tabela 5.5: Visão geral da distribuição dos atributos.

	Total IP	**IP Compartilhado**	**Países**
+P	58.896	21.30%	118
-P	1.622	55.83%	75

A quantidade de endereços IP é o resultado da união de todos os endereços IP encontrados nas seções de resposta e adicional da resposta `DNS`. Os endereços IP compartilhados denotam endereços IP que são utilizados tanto na seção de resposta como na seção adicional. O compartilhamento de endereços IP por domínios maliciosos é superior em relação aos domínios legítimos porque o *botmaster* precisa de pelo menos dois endereços IP para que um nome de domínio seja registrado, segundo as RFC 1537 e 1034. Durante análise da base de dados, foi possível observar que o endereços IP da seção de resposta também estavam na seção adicional. A principal diferença entre os domínios legítimos e maliciosos é que no primeiro caso, os endereço IP estavam associados à outras redes, enquanto no último, o mesmo IP respondia por dois servidores de nome. Por exemplo, `nameserver1.evil.com` e `nameserver2.evil.com` tinham o mesmo endereço IP.

Finalmente, a quantidade de países representa o total de países distintos que estão associados a cada IP por base de dados. Para obter esses dados, foram realizadas consultas à diversas bases incluindo `Team Cymru`, `Whois` e de localização geográfica[4].

O comportamento distinto entre domínios maliciosos e legítimos pode ser visualizado nas Figuras 5.2 e 5.3. As figuras ilustram uma visão geral da relação entre o nome de domínio e endereços IP. Nós verdes denotam nome de domínio, outras cores denotam o grau de entrada para nós de endereço IP, onde os nós cinza denotam maior grau de entrada. Para o processo de geração e análise da figura em questão foi utilizado a ferramenta `gephi`[5].

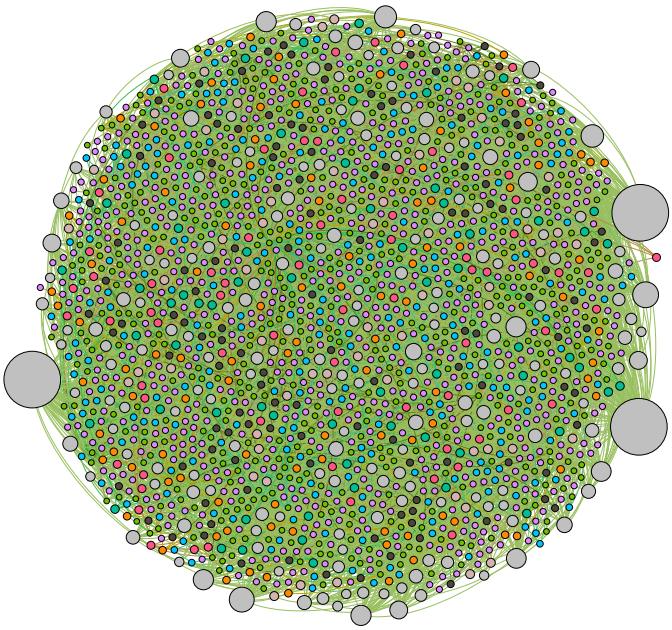


Figura 5.2: Relação entre nomes de domínios e endereços IP observados na base de dados maliciosa. Nós cinza nas extremidades denotam que um número maior de domínios compartilhando mesmo endereço IP.

O tamanho dos nós representa o grau de entrada de um endereço IP, isto é, quantos nomes de domínios apontam para o mesmo endereço. A Figura 5.2 ilustra a base de dados maliciosa enquanto a Figura 5.3 apresenta a relação da base legítima. Domínios legítimos, em média possuem 1.537 endereços IP compartilhados, enquanto os domínios maliciosos 2.847.

Na Figura 5.2 é possível observar que os nós cinza (endereços IP), em sua grande maioria, estão alinhados à margem do grafo. Por outro lado, na Figura 5.3, os nós cinzas

[4] `https://batchgeo.com/`
[5] `https://gephi.org/`

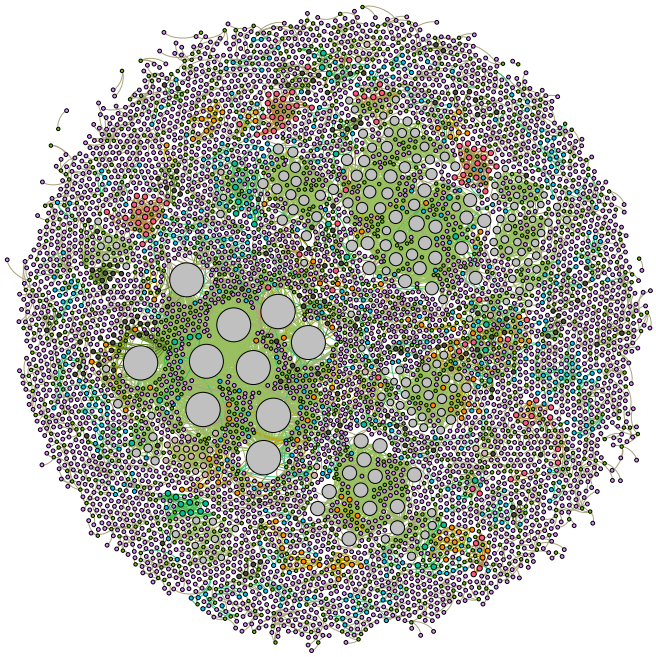


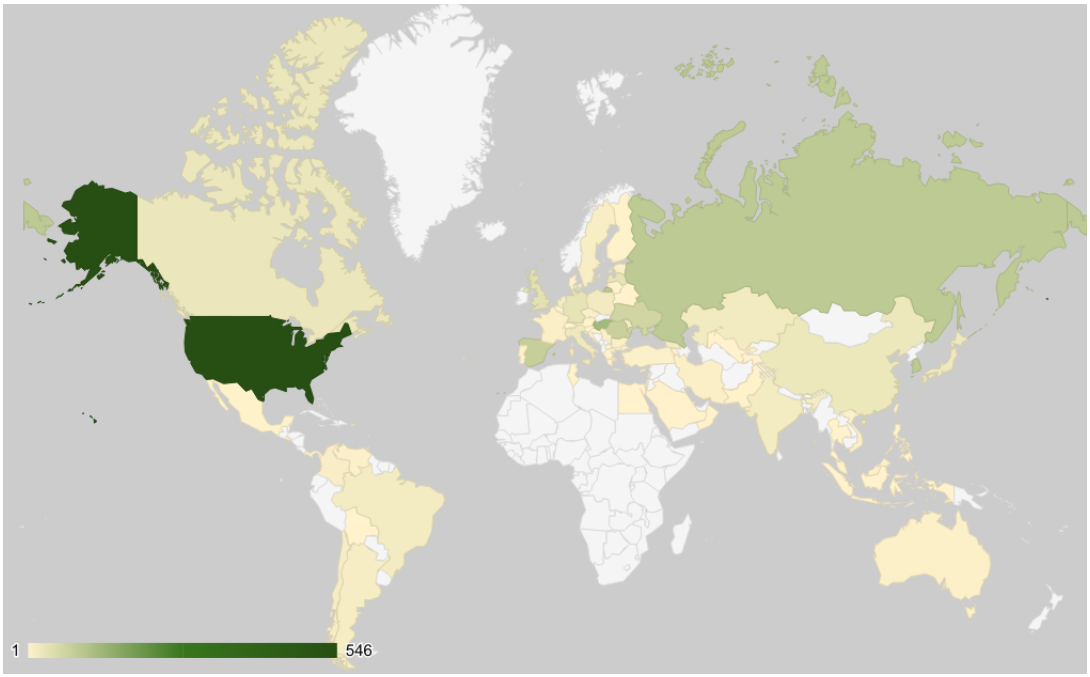
Figura 5.3: Relação entre nomes de domínios e endereços IP observados na base de dados legítima. Organização de nós cinza mais ao centro denota baixo compartilhamento de endereços IP.

são relativamente menores e organizados em grupos centralizados, isto é, os nós estão mais próximos. Isso acontece porque o número de endereços IP na base maliciosa deve atender um número maior de nomes de domínios, pois o *botmaster* precisa criar diferentes nomes para evadir listas negras por outro lado, na base legítima, o compartilhamento de endereços IP é menor visto que a mudança frequente desses endereços de rede podem impactar a reputação do nome de domínio (Nakamura et al., 2018)

A Figura 5.4 ilustra os países utilizados tanto por domínios maliciosos como legítimos. Nessa ilustração, é possível observar que em ambos os casos, os endereços IP estão distribuídos ao redor do globo, onde os EUA apresentam a maior frequência de endereços IP. Vale destacar que algumas abordagens utilizam o código de país como UK, BR e RU como critério para determinar anomalia (Bilge et al., 2014; Zhao e Traore, 2012). Em outras palavras, basta que um endereço IP tenha origem em determinado país para que endereço IP seja considerado como suspeito.

A Figura 5.5 ilustra a quantidade de endereços IP obtidos nas seções de resposta e adicional que estão associados a um nome de domínio malicioso ou legítimo. Para facilitar a visualização, utilizou-se a escala logarítmica no eixo y. É possível observar que domínios legítimos possuem uma distribuição de endereços IP mais concentrada entre os valores 1 a

a)



b)

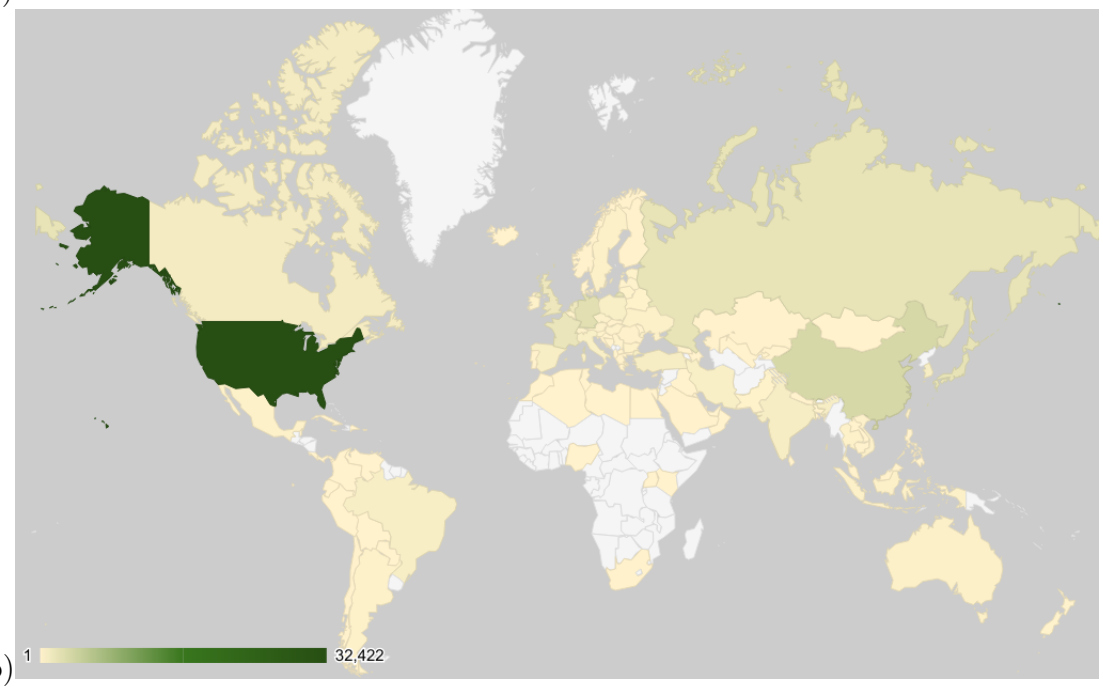


Figura 5.4: Visão geral da distribuição de países que hospedam os nome de domínios da base a) maliciosa e b) legítima.

34, enquanto na base maliciosa, é possível encontrar nomes de domínios com 276 endereços IP associados.

Além disso, é preciso destacar que não é trivial estabelecer um número mínimo de endereços IP que sinalizem que um nome de domínio é suspeito. Por exemplo, Chen et

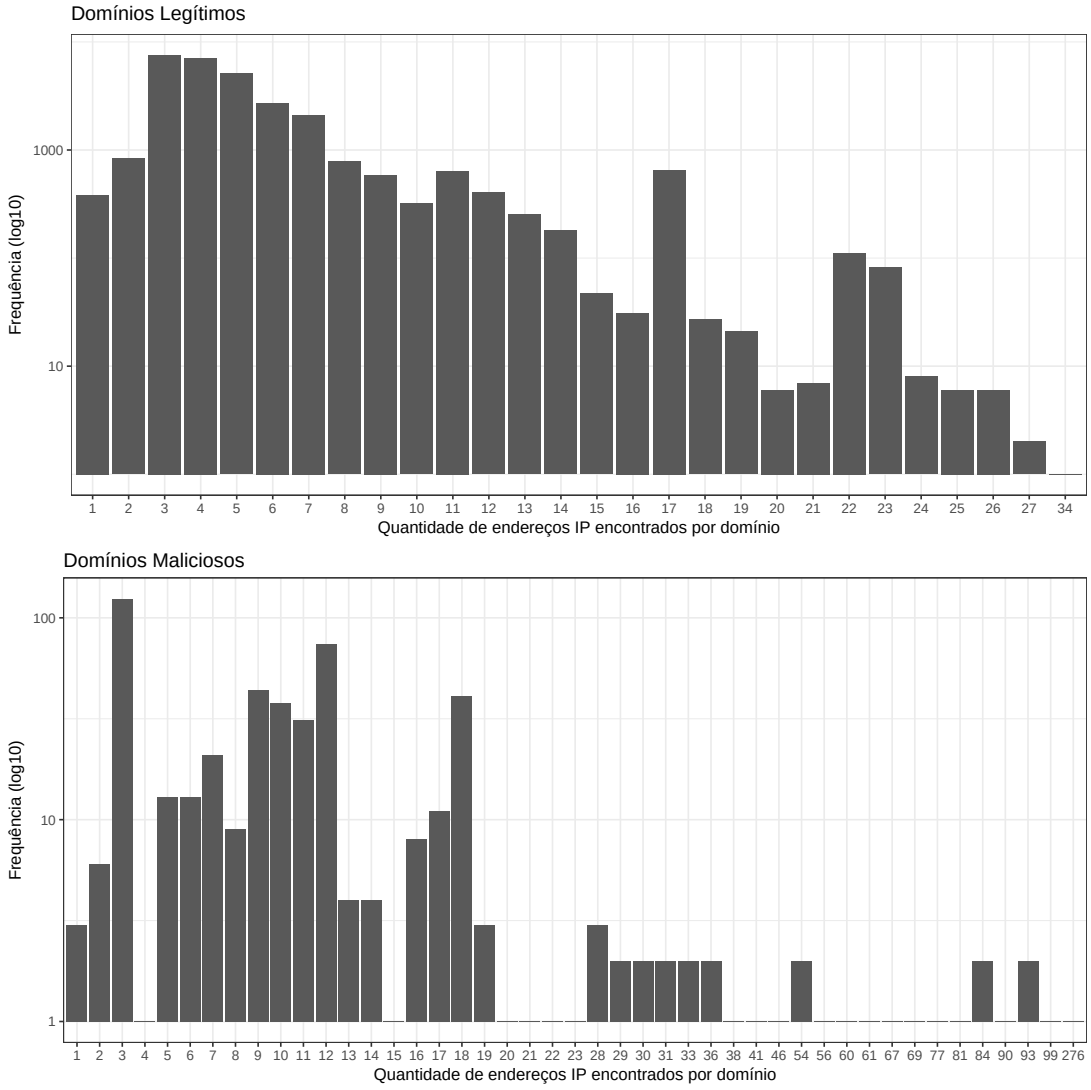


Figura 5.5: Comparação em escala logarítmica de entre quantidade de endereços por nome de domínio em redes legítimas e redes maliciosas.

al. (2013a) mostraram que um dos critérios para investigar esse tipo de comportamento é que o número de endereços IP associados na seção de resposta DNS deve ser maior que 3. Portanto, esse caso resultaria na inclusão de domínios legítimos e suspeitos. No entanto, se o *botmaster* manipular a quantidade de endereços IP na seção de reposta como demonstrado em (Caglayan et al., 2009b), é possível subverter esses critérios de seleção. O comportamento descrito na Figura 5.5 demonstra porque é necessário analisar um nome de domínio a partir de diferentes perspectivas.

A Figura 5.6 demonstra a relação entre a quantidade de Sistemas Autônomos (eixo x) e o total de endereços IP associados em domínios (eixo y). Para os domínios legítimos,

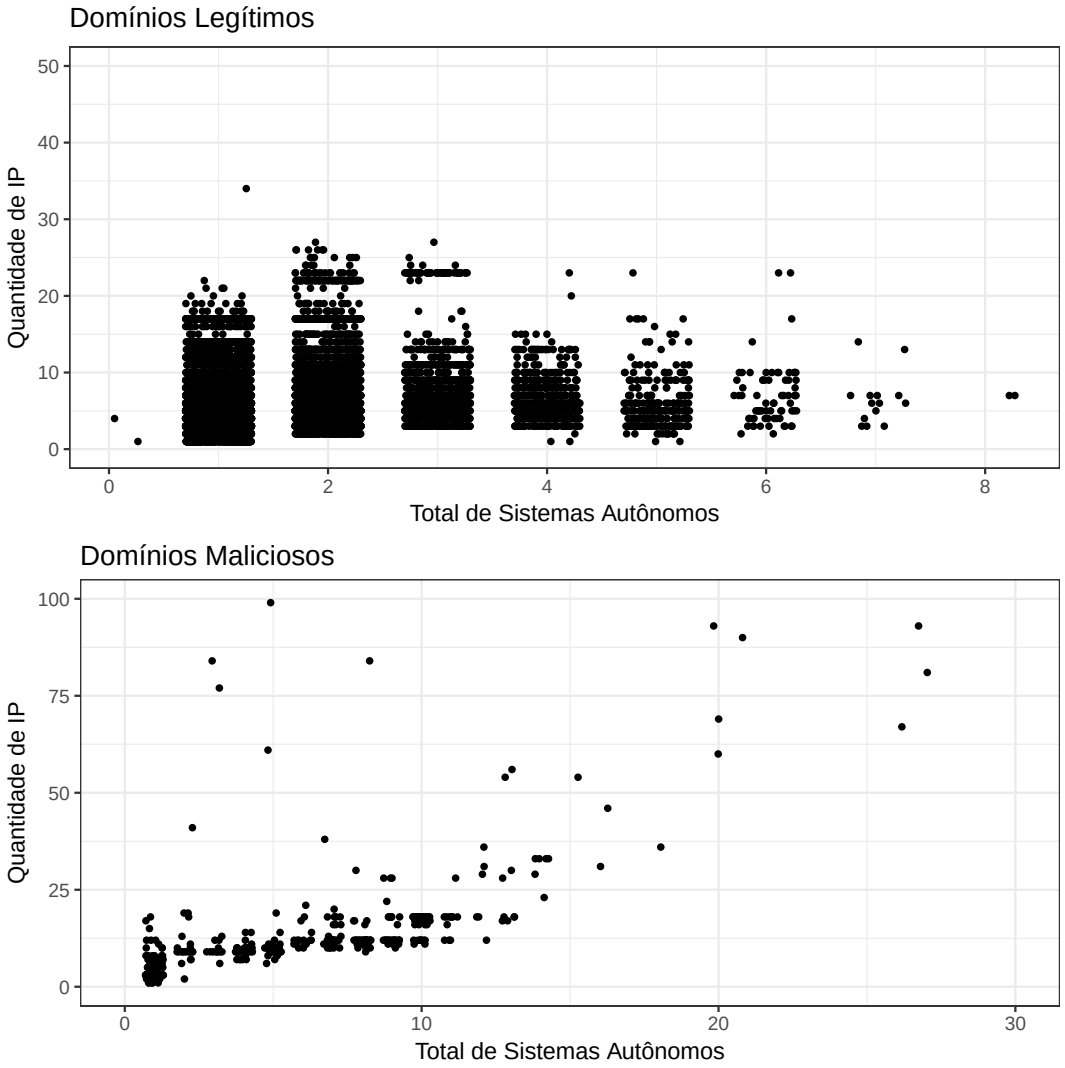


Figura 5.6: Relação entre quantidade de sistemas autônomos vs o total de endereços IP associados aos domínios.

é possível observar que o número de sistemas autônomos por nome de domínio é inferior quando comparado com os domínios da base maliciosa. Além disso, a quantidade de endereços IP associados aos sistemas autônomos é praticamente estável, isto é, 20 endereços IP por ASN.

No caso de domínios maliciosos, a quantidade de sistemas autônomos apresenta valores 3 vezes maior, em média, que os valores associados aos nomes de domínios legítimos. Por exemplo, domínios maliciosos que possuem mais que 25 endereços IP associados também estão atrelados à diversos sistemas autônomos. Esse comportamento acontece porque o *botmaster* sequestra inúmeros computadores ao redor do globo para que a *botnet* continue operando mesmo que algum host seja desativado. A quantidade de `ASN` associada ao nome

de domínio é uma característica relevante e explorada por diferentes trabalhos como (Holz et al., 2008), (Stalmans e Irwin, 2011) e (Grzinic et al., 2014).

Para que o atacante seja capaz de confundir essas abordagens, é necessário um grande número de máquinas infectadas sob seu comando. Com isso, o *botmaster* pode controlar o número de máquinas que são respondidas em uma consulta `DNS`. As políticas de restrição de resposta podem ser baseadas na origem do IP solicitante, ou a partir de um número de consultas repetidas para esse domínio.

Vale lembrar que o controle de inúmeros computadores é factível, o exemplo do `worm Confiker` (Shin e Gu, 2010b) pode ser citado, pois esse código malicioso foi capaz de infectar mais de 10 milhões de computadores ao redor do globo.

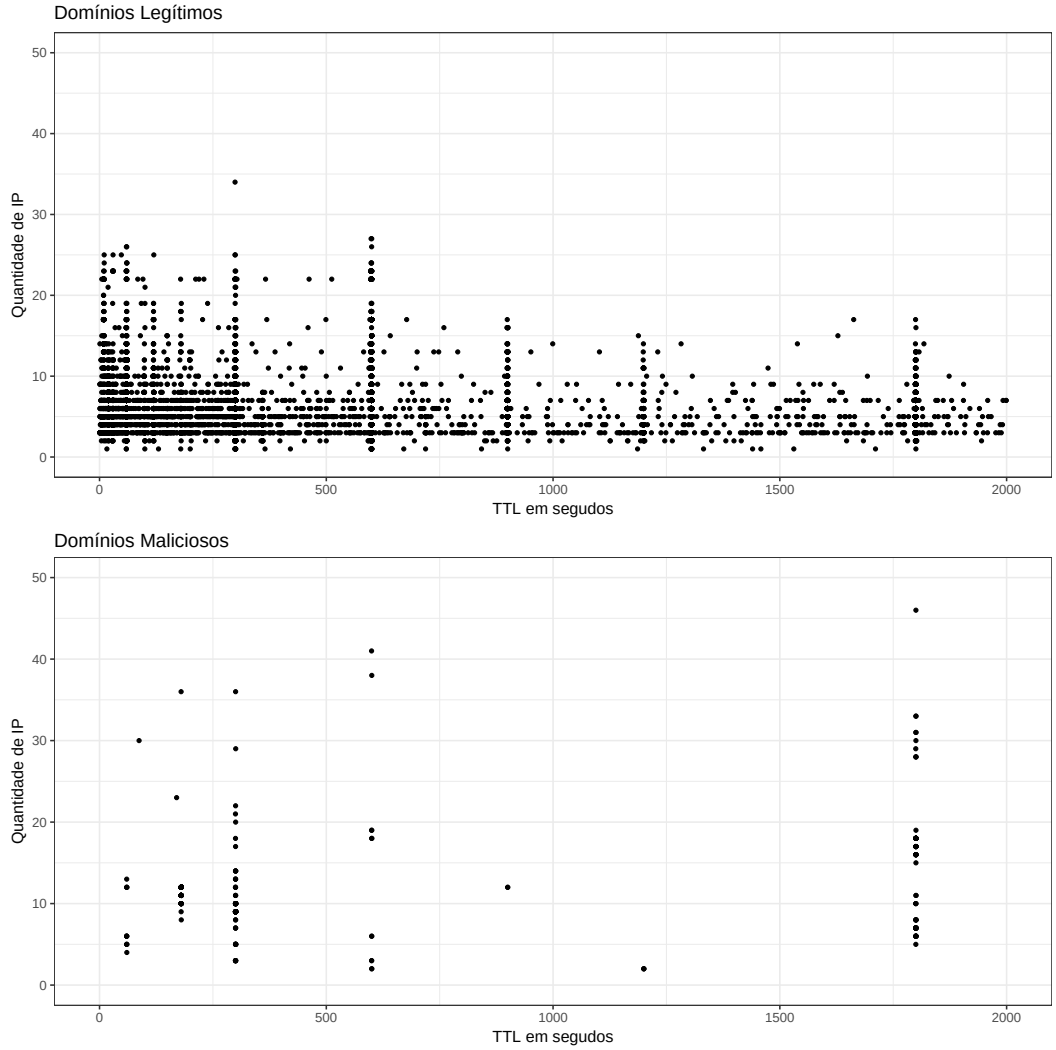


Figura 5.7: Uso do TTL por domínios legítimos e maliciosos em relação à quantidade de endereços IP associados.

No entanto, mesmo que um número significativo de hosts ao redor do globo esteja sob controle de atacantes, é possível a partir do monitoramento *continuo* e *distribuído* identificar comportamentos que são inerentes ao funcionamento de redes maliciosas. Por exemplo, o tempo de viagem de ida e de volta (`RTT`) de hosts maliciosos pode variar significativamente, enquanto que em servidores legítimos esse tempo é razoavelmente estável (Castelluccia et al., 2009; Lin et al., 2013). Além disso, a disponibilidade da máquina infectada também é outra característica que não pode ser controlada pelo atacante, o que pode auxiliar no processo de identificação de redes de serviço de fluxo rápido.

O cenário descrito acima corrobora com as hipóteses deste trabalho, onde a *botnet* está em constante adaptação e que o monitoramento observando diferentes características ao longo do tempo também é necessário para mitigar o avanço dessa ameaça.

Embora existam algumas características que demandem maior tempo de monitoramento, outras são mais simples de serem obtidas, como no caso do `TTL`. A Figura 5.7 ilustra o tempo de vida extraído da base de dados (eixo x) em relação à quantidade de IP associados ao nome de domínio (eixo y). Para facilitar o processo comparativo, os valores no eixo x foram limitados em 2000 segundos (33 minutos). No entanto, os valores observados para domínios legítimos podem alcançar dias, enquanto que os valores de domínios maliciosos permanecem abaixo de 1800 segundos. Tal comportamento é frequentemente identificado por inúmeros trabalhos, pois para haver mudanças mais rápidas de endereços IP esse valor deve ser baixo.

Além das características citadas, a Tabela 5.6 ilustra uma análise do atributo quantidade de endereços IP que pertencem a banda larga. Nesse estudo apenas a base maliciosa foi investigada, pois não foi possível encontrar hosts que possuíam em seu nome de domínio reverso (`PTR`) alguma referência para esse tipo de configuração. Para obter o nome reverso de um endereço IP é realizada uma consulta do tipo `PTR` e o retorno é armazenado para análise posterior.

No processo de análise, o nome de domínio é segmentado tomando como base o ponto que separa os níveis de domínio. Por exemplo, o nome `foo.example.com` resulta três campos dos quais são comparados na base de nomes de banda larga conhecidos. Na tabela em questão, os primeiros 5 nomes mais utilizados são demonstrados. Vale lembrar que tais nomes foram coletados a partir da revisão da literatura, demonstrada no Capítulo 3. É importante observar que 34% de endereços IP não possuem um nome reverso configurado (`NXDomain`), no entanto, para efeito de comparação, o resultado `NXDomain` representa 38% para base legítima.

Tabela 5.6: Comportamento de hosts que pertencem a banda larga

Total	Nome
34%	`NXDomain`
6%	`dyn`
5%	`catv`
3%	`cpe`
3%	`dhcp`

A investigação desse atributo em escala global é difícil, pois embora exista uma definição que formalize que todo endereço IP real na Internet deve ter um nome reverso (Barr, 1996), não há um padrão de utilização desses nomes como `dial-up, home, cpe` ou `customer`. Durante a realização deste trabalho, outros nomes utilizados por provedores de banda larga foram descobertos como `kansai-bb`, `metrocarrier`, `comcast` e `emome-ip` pertencentes ao Japão, Estados Unidos, México e Taiwan, respectivamente.

Para identificar novos nomes de provedores de banda larga, cada endereço IP associado aos domínios da base maliciosa foi consultado. Além disso, consultas `Whois` e pesquisas em máquinas de buscas também foram realizadas para auxiliar nesse processo. No entanto, mesmo que seja possível obter tais informações sobre os endereços IP, a coleta desse tipo de dado ainda é demorada e não fornece todos os dados necessários.

5.3 Desempenho do Framework AFFACO

5.3.1 Algoritmo de Correlação AFFACO-M

Os resultados obtidos pelo algoritmo de correlação de abordagens `AFFACO-M` podem variar devido aos valores estabelecidos inicialmente para seu funcionamento. Esse comportamento acontece devido ao processo de correlação de abordagens ser probabilístico, isto é, a definição de um valor individual pode afetar os demais parâmetros do algoritmo. Por esse motivo, é necessário fazer um estudo da relação entre esses componentes durante o processo de correlação.

Para demonstrar o impacto de cada parâmetro no comportamento do `AFFACO-M`, considere o estudo comparativo entre os possíveis valores para α, β e ρ, conforme ilustrado na Figura 5.8. É importante destacar que os resultados encontrados consideram que o Custo de Classificação (eixo Y) não é superior a 60% do Custo Total dos Atributos. No eixo X é possível observar os possíveis valores em que cada parâmetro pode assumir dentro do intervalo entre 0.1 e 1.0.

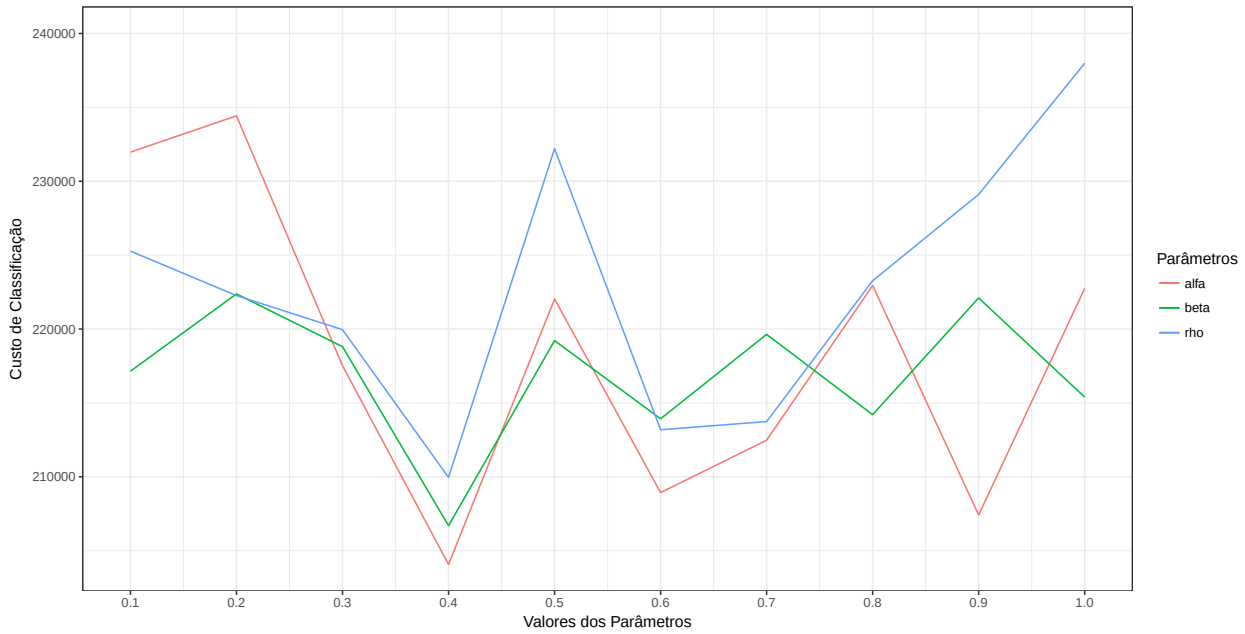


Figura 5.8: Estudo comparativo individual entre os parâmetros α, β e ρ.

Para obter esses resultados, um parâmetro tinha seus valores alterados, enquanto os demais eram fixo. Por exemplo, enquanto os valores de α variavam entre 0.1 e 1.0, os demais parâmetros estavam com valor padrão, isto é 0.5, definido em (Dorigo e Stützle, 2004). O mesmo experimento foi realizado para *beta* e ρ, durante avaliação individual. Os resultados mostram que o valor de 0.4 apresentou melhor retorno para os três parâmetros individualmente.

Embora o valor de 0.4 atribuído para cada parâmetro isoladamente tenha resultado em regras de menor custo, não é possível afirmar que esse mesmo valor atribuído para α, β e ρ também seja capaz de apresentar melhores resultados, como ilustra os histogramas na Figura 5.9. Para comprovar essa observação, foram geradas 100 regras em dois cenários distintos. No primeiro, α, β e ρ estavam com o valor de 0.4. No segundo momento foram atribuídos 0.6, 0.3, 0.7 para os mesmos parâmetros.

Os resultados mostram a frequência dos custos obtidos a partir de 100 regras geradas pelo algoritmo `AFFACO-M`. É possível observar que a configuração 0.6, 0.3 e 0.7 apresenta melhores resultados quando comparados contra o valor de 0.4. Em média, o segundo cenário gerou 37% de regras com custo abaixo de 200000, enquanto que o primeiro gerou apenas 23%. Além disso, a segunda configuração foi capaz de produzir regras mais eficientes; o menor e o maior custo gerado foram 167407 e 266035, respectivamente, enquanto que na primeira opção, os valores encontrados são 168872 e 270513.

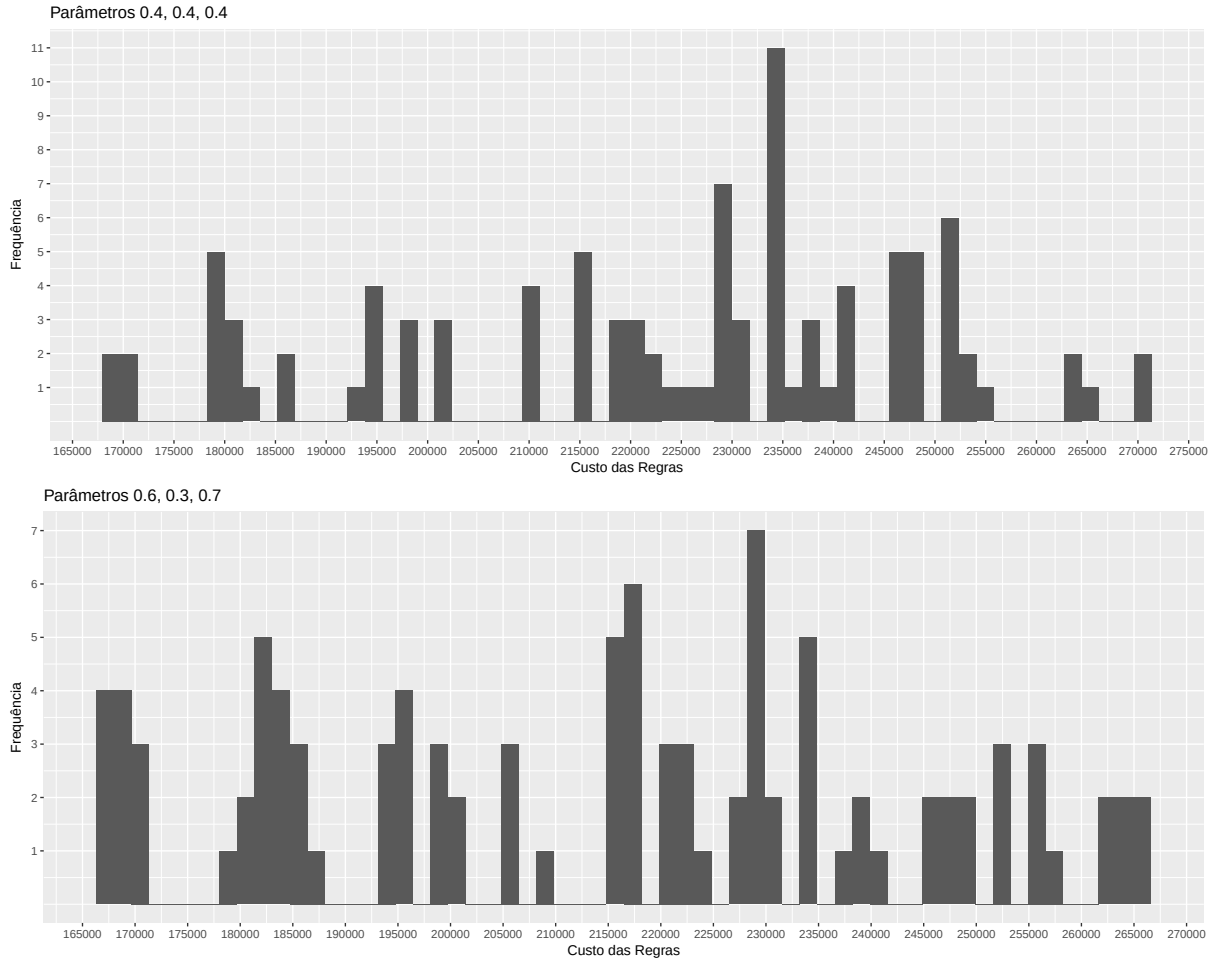


Figura 5.9: Estudo comparativo entre os parâmetros α, β e ρ.

Para encontrar os valores 0.6, 0.3 e 0.7, foram geradas 7! regras, isto é, 5.040 combinações. Esse processo teve dois objetivos *i*) entender como as formigas produzem as regras de classificação e *ii*) identificar as regras com menor e maior custo (acurácia). A partir desse conjunto de regras, foi possível confirmar que tanto a regra com menor custo como a regra com maior nível de acurácia, não conseguem superar as regras que são geradas respeitando o equilíbrio entre custo e acurácia.

A principal dúvida era entender a relação entre os parâmetros do algoritmo `AFFACO-M` para que regras que apresentassem melhor equilíbrio entre custo e acurácia pudessem ser identificadas. Vale lembrar que o processo de correlação de abordagens para gerar uma regra de classificação é um processo probabilístico, então não é possível ajustar os parâmetros para que sejam encontrados os mesmos resultados. As análises a seguir tomam como base os valores 0.6, 0.3 e 0.7 para os parâmetros α, β e ρ, respectivamente.

5.3.2 Quantidade de Formigas vs Tempo de CPU vs Convergência

O processo de correlação de formigas sofre influência das variáveis α, β, ρ, `quantidade de formigas`, `restrição da regra` e `convergência`. A quantidade de formigas é um parâmetro que interfere na velocidade em que uma regra tende a convergir além disso, para que a comunicação entre as formigas funcione (estigmergia), que nesse caso é denotado como a atualização e evaporação do feromônio, é necessário que a quantidade de formigas no grafo permita que as interações propaguem feromônio ao longo das arestas.

Para (Dorigo e Stützle, 2004), o número mínimo de formigas que devem ser associadas ao grafo é equivalente ao número de vértices modelados, pois durante o processo de correlação pode acontecer casos onde a cidade do ponto de partida não é levada em consideração. Tanto em Dorigo e Stützle, como no modelo `AFFACO-M`, o posicionamento de formigas é aleatório.

Partindo da premissa que a quantidade de formigas também exerce função relevante no processo de correlação das abordagens, a Figura 5.10 ilustra um estudo que mostra o tempo de correlação das abordagens (eixo Y) em relação ao número de formigas (eixo X). Foram realizadas 200 execuções do algoritmo `AFFACO-M`, onde a cada 10 intervalos no eixo X, a quantidade de formigas cresce em múltiplos de 7. Isso acontece porque no mínimo o número de formigas deve ser igual ao número de nós do grafo (Dorigo e Stützle, 2004). Por exemplo, entre os intervalos 1 e 10, a quantidade de formigas é o mesmo; 7, já no intervalo entre 11 e 20 a quantidade denota 14 formigas, e assim sucessivamente.

Não obstante, embora a quantidade de formigas, a partir de determinado ponto, não influencie o tempo de execução do algoritmo, é importante investigar se tal configuração pode contribuir com o desempenho das regras produzidas.

Em um estudo empírico foi possível analisar a relação entre a quantidade de formigas e a convergência da regra, isto é, quantas vezes as formigas deveriam passar por um caminho até que o processo de correlação das abordagens fosse interrompido. O algoritmo `AFFACO-M` foi executado 100 vezes variando o valor da convergência a cada 10 intervalos, entre 7 a 140, em outras palavras, um caminho completo deveria ser percorrido pelo menos 7 vezes e no máximo 140 vezes. Durante as análises, foi possível identificar que o número de convergência com valores superiores ao número de formigas pode resultar em regras com maiores custos. Para os resultados a seguir, o valor mais apropriado que leva em consideração tanto o número de formigas como o valor da convergência é 35 e 70, respectivamente.

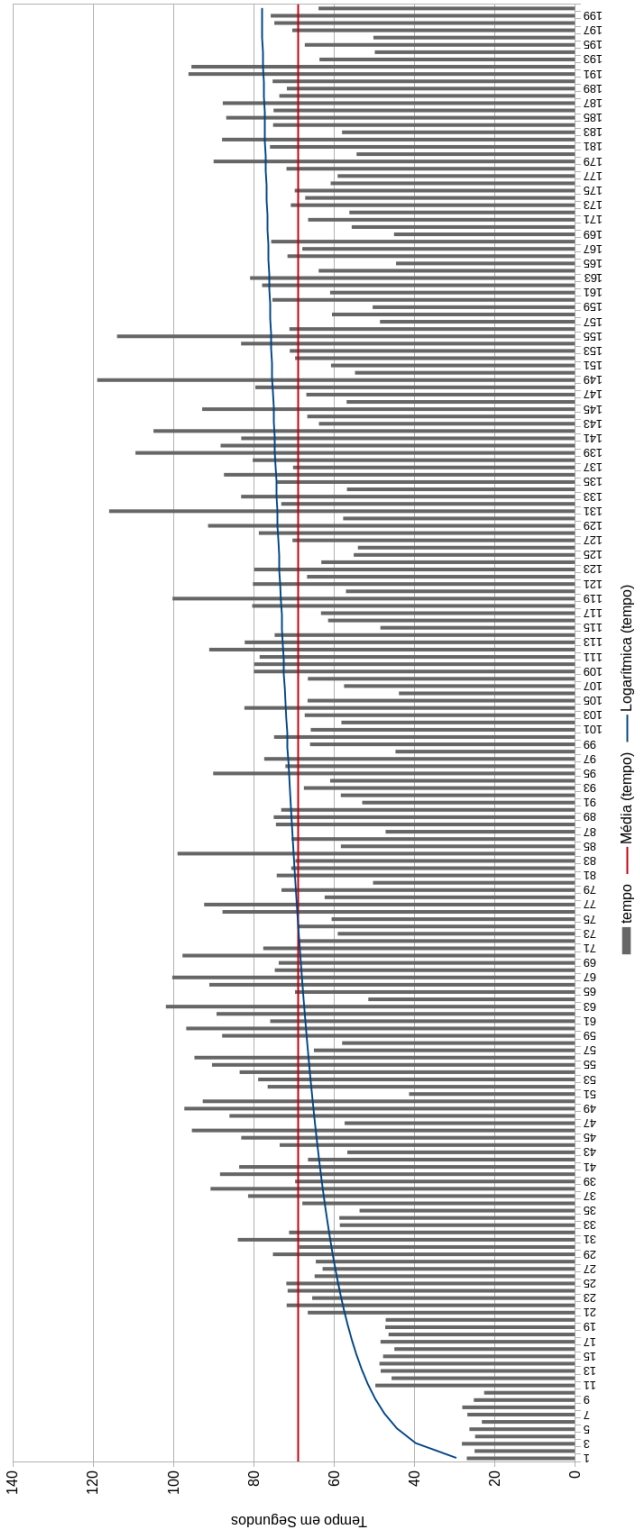


Figura 5.10: Análise do número de formigas em relação ao tempo de CPU

5.3.3 Sistema de Políticas AFFACO-P

Cada regra gerada pelo `AFFACO-M` segue o seguinte formato $[a_1, a_2, \ldots, a_n]$, onde a_i denota a ordem em que cada abordagem será executada pelo algoritmo `AFFACO-P`. A Tabela 5.7 apresenta o identificador da abordagem, citação na literatura, acurácia e `CTA`. O identificador da abordagem informado na tabela em questão é utilizado apenas como referência no `AFFACO-P` e não denota a sua ordem fixa de execução.

Tabela 5.7: Custo e acurácia individual das abordagens

ID	**Abordagem**	**Acurácia**	`CTA`
0	(Huang et al., 2010)	78,46%	118
1	EntropyFFNet	94.52%	72
2	(Passerini et al., 2008)	79,70%	396
3	(Caglayan et al., 2009b)	88,12%	372
4	(Chen et al., 2013a)	82,84%	156
5	(Grzinic et al., 2014)	64,42%	154
6	(Holz et al., 2008)	70,66%	85

A partir do estudo das regras geradas, foi possível identificar as sequências de abordagens que apresentam os menores custos, maiores acurácias e que levam em consideração o equilíbrio entre custo e acurácia. Vale lembrar que o custo da regra faz referência à soma dos `CTA`, enquanto a acurácia denota a soma total das acurácias. Nesta seção é avaliado também o desempenho da regra, isto é, qual custo de classificação por número de abordagens.

Em termos gerais, a acurácia do algoritmo `AFFACO-P` é de 93.56%. Quando comparado com as demais abordagens, os resultados mostram que é possível identificar corretamente os nomes de domínios legítimos, permanecendo em segundo lugar em algumas métricas, como ilustra a Tabela 5.8. É importante trazer para discussão que as abordagens selecionadas possuem um `CTA` fixo, portanto, do ponto de vista macro, a ordem de classificação não modifica o valor final da acurácia para esse conjunto de dados, no caso onde *todas* as abordagens são aplicadas. Nesse caso, é possível fazer uma analogia para um modelo de classificação que emprega um conjunto significativo de características para determinar o comportamento de uma nome de domínio. No entanto, tal estratégia não alcança os objetivos desta Tese de Doutorado; classificar um nome de domínio no menor tempo possível obtendo a maior acurácia.

Para demonstrar que é possível atingir esse objetivo, o parâmetro `Restrição do Custo da Regra`, citado anteriormente, exerce função relevante no processo de correlação

Tabela 5.8: Avaliação individual das abordagens usando validação cruzada de 10 partições (*folds*).

Métrica	A_1	A_2	A_3	A_4	A_5	A_6	A_7	AFFACO-P
Acurácia	88.12	82.84	61.42	79.70	70.66	78.48	94.52	93.56
E. Classificação	11.88	17.16	38.58	20.30	29.34	21.52	5.48	6.44
Recall	89.80	89.40	100.00	97.00	98.40	92.40	96.20	100.00
Specificity	86.39	76.08	21.65	61.86	42.06	64.12	92.78	86.90
FPR	13.61	23.92	78.35	38.14	57.94	35.88	7.22	13.10
Precisão	87.18	79.40	56.82	72.39	63.65	72.64	93.22	88.76
F-Measure	88.47	84.10	72.46	82.91	77.30	81.34	94.69	94.04
AUC	88.10	82.74	60.82	79.43	70.23	78.26	94.49	93.45

de abordagens. Por exemplo, considere os cenários que descrevem 4 tipos de combinações das regras na classificação de redes de serviço de fluxo rápido, como ilustra a Tabela 5.9.

Tabela 5.9: Possíveis cenários para aplicação de regras de classificação.

ID	**Regra**	**Custo 60%**	**Tipo**
R_1	[1, 3, 4, 2, 0, 6, 5]	996	Maior Acurácia
R_2	[6, 0, 1, 5, 3, 2, 4]	429	Menor Custo
R_3	[0, 1, 4, 6, 5, 3, 2]	431	Melhor Equilíbrio
R_4	[5, 4, 3, 2, 0, 6, 1]	1078	Maior Custo

No cenário da regra R_1, as melhores acurácias estão distribuídas ao longo de 60% da regra, isto é, nas 4 primeiras abordagens, conforme ilustrado na Tabela 5.10. Nessa configuração, 195 nomes de domínios (49.36%) foram classificados usando as 4 primeiras abordagens, denotando 94.87% de acurácia; 85 domínios usaram 5 abordagens; 9 domínios demandaram 6 abordagens; e 6 domínios precisaram do último voto. No total, 395 nomes de domínios foram classificados a um custo de 308704 pontos.

Tabela 5.10: Custo da regra R_1 - [1, 3, 4, 2, 0, 6, 5].

N abordagens	Classificados	Custo por classificação	TP	TN	FP	FN
4	195	996	113	72	10	0
5	85	1114	27	46	11	1
6	9	1202	5	1	2	1
9	6	1356	3	3	0	0
Total	**395**	**308704**				

A Tabela 5.11 apresenta o desempenho de classificação da regra R_2, a qual denota o menor custo. Nessa configuração, 162 nomes de domínios (41.01%) foram classificados usando as 4 primeiras abordagens, denotando 91.97% de acurácia; 60 domínios usaram

5 abordagens; 28 domínios demandaram 6 abordagens; e 45 domínios precisaram do último voto. Embora os custos de classificação da regra R_2 sejam menores que a regra R_1, os resultados em R_2 precisam de mais abordagens do último voto para definir o comportamento dos domínios. No total, 395 nomes de domínios foram classificados a um custo de 212358 pontos.

Tabela 5.11: Custo da regra R_2 - [6, 0, 1, 5, 3, 2, 4].

N abordagens	Classificados	Custo por classificação	TP	TN	FP	FN
4	162	429	130	19	13	0
5	60	804	12	38	8	2
6	28	1200	2	24	2	0
9	45	1356	4	41	0	0
Total	**395**	**212358**				

Enquanto as regras R_1 e R_2 apresentam a maior acurácia e o menor custo, respectivamente. A regra R_3 demonstra como o processo de correlação de abordagens no algoritmo `AFFACO-M` é capaz de gerar regras que representam o equilíbrio entre esses dois critérios de restrição, como ilustra a Tabela 5.12. Nessa configuração, 191 nomes de domínios (48.35%) foram classificados usando as 4 primeiras abordagens, denotando 93.72% de acurácia; 44 domínios usaram 5 abordagens; 54 domínios demandaram 6 abordagens; e 6 domínios precisaram do último voto. A diferença na acurácia entre R_1 e R_3 é de 1,15 ponto percentual. Além disso, para atingir 94.87% de acurácia em 4 abordagens, a regra R_1 utilizou mais que o dobro de custo para alcançar esse resultado. No total, 395 nomes de domínios foram classificados a um custo de 168037 pontos.

Tabela 5.12: Custo da regra R_3 - [0, 1, 4, 6, 5, 3, 2].

N abordagens	Classificados	Custo por classificação	TP	TN	FP	FN
4	191	431	126	53	10	2
5	44	585	21	11	12	0
6	54	960	1	46	7	0
9	6	1356	0	6	0	0
Total	**395**	**168037**				

Finalmente, a regra R_4 representa a sequência de abordagens que possuem o maior custo, conforme ilustrado na Tabela 5.13. Em comparação com as outras regras, essa configuração apresentou o maior número de falso positivo e menor acurácia para as primeiras 4 abordagens (83.73%). Os resultados ainda mostram que 91 domínios utilizaram 5 abordagens. Apenas um nome de domínio teve que ser investigado por 6 abordagens e 37 domínios foram classificados pelo último voto.

Tabela 5.13: Custo da regra R_4 - [5, 4, 3, 2, 0, 6, 1].

N abordagens	Classificados	Custo por classificação	TP	TN	FP	FN
4	166	1078	120	19	25	2
5	91	1199	27	60	4	0
6	1	1284	1	0	0	0
9	37	1356	0	37	0	0
Total	**395**	**340011**				

A Figura 5.11 apresenta um estudo comparativo entre as regras de menor custo (minor), maior acurácia (acc) e melhor equilíbrio (best). Nesse estudo, o algoritmo `AFFACO-P` foi executado 100 vezes para 100 amostras de base de dados. Para gerar 100 amostras da base de dados, o módulo `sample`[6] da linguagem de programação `R` foi aplicado. Para cada amostra de dados, tanto o conjunto de treino como de teste eram criados.

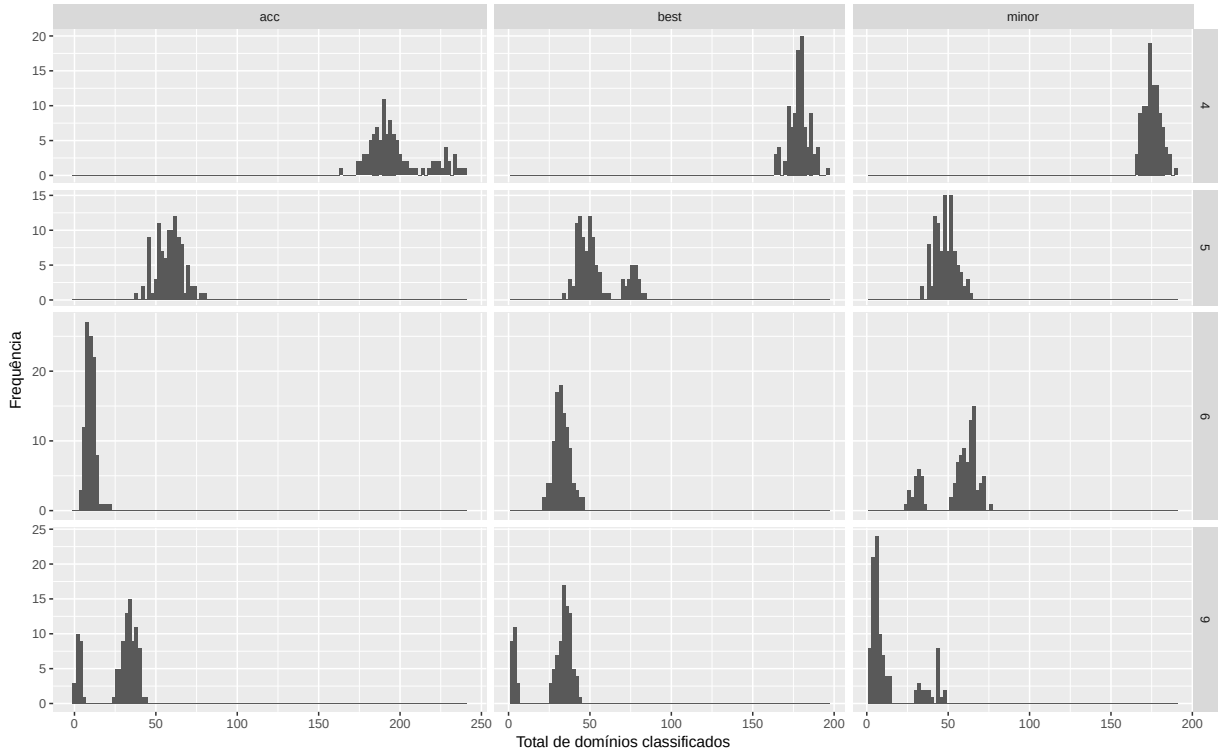


Figura 5.11: Estudo comparativo entre as regras de menor custo, maior acurácia e realizadas pelo `AFFACO-P`.

Os resultados mostram o desempenho de classificação para cada conjunto de regras 4, 5, 6 e 9. Vale lembrar que o sistema de políticas `AFFACO-P` utiliza 9 abordagens quando existe algum tipo de empate durante o processo de investigação. Para 4 abordagens,

[6]https://stat.ethz.ch/R-manual/R-devel/library/base/html/sample.html

a regra do melhor equilíbrio apresenta resultados mais frequentes que a regra do menor custo. Por outro lado, a base do histograma da regra com maior acurácia denota que mais domínios foram classificados por essa regra no entanto, a frequência é inferior aos valores encontrados pelo melhor equilíbrio. Para regra do menor custo, o resultado mostra que mais domínios precisaram do último voto (9 abordagens), isto é, o desempenho total da regra foi inferior quando comparada com outras sequências.

5.3.4 Discussão dos Resultados do Framework AFFACO

O problema de detecção de *botnets* demanda que novas estratégias sejam criadas para tentar mitigar esse tipo de comportamento malicioso. O *framework* proposto apresentou uma alternativa que está sustentada na análise de redes de serviço de fluxo rápido a partir de diferentes perspectivas. Partindo dessa tratativa, é possível definir diferentes abordagens de detecção que podem seguir critérios definidos por operadores de rede. Isto é, a principal vantagem do `AFFACO` é que não é necessário depender de uma única visão para tentar resolver o problema desse tipo de rede maliciosa.

O sistema de políticas `AFFACO-P` é um protótipo utilizado para demonstrar que o algoritmo `AFFACO-M` é capaz de produzir modelos de detecção que podem combinar menor custo e maior acurácia. Além disso, o *framework* proposto demonstrou que num ambiente real, a coleta de características no menor tempo é essencial para para interromper um ataque mais rápido.

No entanto, o *framework* proposto possui limitações, por exemplo o algoritmo `AFFACO-P` é um fusor de resultados e que pode tomar decisões a partir de um sistema de votação pela maioria. Isso significa que se a maioria das abordagens apresentarem um resultado equivocado, então as ações do algoritmo `AFFACO-P` serão acionadas a partir de tais resultados. Além disso, caso um atacante descubra a sequência da regra gerada pelo algoritmo `AFFACO-M`, é possível manipular um conjunto mínimo de abordagens para evadir do sistema de detecção

Para tentar reverter esse cenário de contra-ataque, algumas estratégias podem ser adotadas. Por exemplo, uma maneira para evitar esses mecanismos de subversão é buscar algoritmos mais eficientes para o `AFFACO-P`. Ao invés de um sistema por votação simples, o classificador pode pontuar de maneira distinta abordagens que possuem melhores resultados em determinada métrica como Especificidade ou Sensitividade.

No algoritmo `AFFACO-P`, cada abordagem utilizada possui a mesma pontuação no sistema de voto pela maioria. Tal método de pontuação pode elevar o tempo total de classificação do `AFFACO-P` caso o tamanho da regra seja consideravelmente grande. Isto

é, não é possível interromper a sequência de classificação de um nome de domínio se a maioria das abordagens não entrar em um tipo num consenso.

Outra limitação está no movimento probabilístico de seleção de abordagens, descrito no Capítulo 4. No modelo atual, caso duas abordagens possuam a mesma acurácia e `CTA`, o resultado dessa seleção é incerto, principalmente na primeira interação das formigas, onde todas as arestas possuem o mesmo feromônio. Além disso, o parâmetros utilizados no processo de correlação de abordagens não permitem selecionar abordagens baseadas em outro critério como algoritmos de classificação supervisionados.

Finalmente, embora o modelo representativo de abordagens, descrito no Capítulo 4, permita que diferentes conceitos e estratégias de detecção de anomalias incluindo algoritmos de aprendizagem de máquina e redes neurais profundas sejam associados ao *framework* proposto. Não é possível criar uma representação que inclua abordagens baseadas em sistemas de reputação como `PhishTank` ou `Google Safe Browsing`, pois tais sistemas não oferecem `CTA` ou nível de acurácia. No entanto, esses sistemas podem ser utilizados como abordagens auxiliares na função `BurdenMethod`.

6

Conclusão e Trabalhos Futuros

Este Capítulo apresenta as principais contribuições alcançadas durante a formalização de um modelo baseado em Otimização por Colônia de Formigas para detecção de Redes de Serviço de Fluxo Rápido, assim como novos desafios que podem ser estendidos a partir do modelo proposto.

O modelo proposto é um algoritmo bioinspirado no comportamento artificial das formigas e que são utilizadas no tratamento de problemas de otimização combinatória. Este trabalho demonstrou que a detecção de redes de serviços de fluxo rápido é desafiadora e que tal processo pode ser modelado como um problema de otimização combinatória. Além disso, até onde vai o conhecimento deste autor, este trabalho é a primeira tratativa baseada em Otimização por Colônias de Formigas que tenta mitigar esse tipo de rede maliciosa.

Dentro do contexto geral do processo de detecção de redes de serviço de fluxo rápido, é preciso deixar claro que o `AFFACO-M` é o algoritmo responsável pela correlação de abordagens selecionadas e, seu produto resultante, é um modelo que pode ser utilizado por algum tipo de classificador. O algoritmo `AFFACO-P` é um sistema capaz de indicar o comportamento de um domínio suspeito a partir da fusão dos resultados obtidos por cada abordagem definida no modelo gerado pelo algoritmo `AFFACO-M`. Os resultados desse processo de classificação denotam 93.65% de acurácia e que o processo de correlação de abordagens levou em consideração o custo e a acurácia das abordagens avaliadas. Além disso, foi possível demonstrar que a correlação de abordagens é uma alternativa viável para investigar o problema de redes de serviço de fluxo rápido a partir de diferentes perspectivas e que selecionar abordagens aleatoriamente para realizar esse tipo de análise não representou melhores resultados.

Entre as contribuições deste trabalho, é importante destacar também o processo de modelagem do problema de detecção de redes de serviço de fluxo rápido em um problema semelhante ao Problema do Caixeiro Viajante, onde as cidades foram tratadas como abordagens e as arestas mapeadas para definir relação de custo entre as abordagens, possibilitando que outros métodos de detecção possam ser incluídos no modelo formal.

Durante a realização deste trabalho, os custos dos atributos e a restrição de custo da regra representaram maior influência do processo de correlação de abordagens e desempenho de classificação de nomes de domínio. Por exemplo, existem atributos que podem ser obtidos em questões de segundos e que são simples de estipular o custo de obtenção. Por outro lado, existem atributos que dependem de tráfego histórico de dados e que podem envolver diferentes custos para sua coleta total. Um típico exemplo é o atributo que verifica se um servidor de nomes está associado a múltiplos nomes de domínio. Portanto, dependendo do tempo de período de coleta o custo desse atributo pode variar, pois quanto maior for o volume de informações na base de dados maiores são as chances de encontrar servidores sendo compartilhados.

Além das contribuições do *framework* proposto, este trabalho apresentou uma abordagem denominada como `EntropyNET` baseada em entropia capaz de identificar redes de serviço de fluxo rápido. Vale ressaltar que durante a construção do `BurdenMethod` foi realizado um estudo comparativo entre os algoritmos `Random Forest` e `Logistic Regression` utilizando todos os atributos selecionados nesse trabalho. Pois, o principal objetivo do `BurdenMethod` é fornecer uma análise complementar e de maior abrangência sobre o domínio suspeito.

6.1 Lições Aprendidas

Durante o processo de coleta de dados, o pesquisador deve implementar sua própria ferramenta de consultas `DNS`, pois ocorreram inúmeros casos onde o `DIG` não foi capaz de obter as três seções completas da resposta DNS. No início dos trabalhos, como alguns nomes de domínios não forneciam esses dados, era necessário mandar consultas específicas para as seções pendentes, como por exemplo, o comando `dig -t NS foo.exemple.com` obtem a lista de servidores de nome com autoridade pela zona de domínio. Para cada nome de servidor nessa lista, novas consultas do tipo `A` foram enviadas.

Durante a criação do vetor de atributos, inúmeras consultas `SQL` foram desenvolvidas para criar ou correlacionar as características utilizadas pelas abordagens. É importante conhecer previamente todas as abordagens e atributos que serão armazenados na base de dados. Conforme o crescimento da base pela inclusão de novos nomes de domínios foi

necessário remodelar o esquema que dava suporte aos dados. Numa próxima modelagem de dados, uma sugestão é que a implementação possa ser feita usando soluções distribuídas como `ElasticSearch` ou `Hadoop`.

6.2 Trabalhos futuros

O processo de correlação da regra basicamente avalia se a acurácia da regra gerada é superior a regra anterior. Ao invés do modelo depender da acurácia como critério de seleção, podem ser outras métricas como a taxa de falso positivo ou a sensitividade.

A heurística deve considerar o processo de extração ou protocolo de rede utilizado para coleta de características. Por exemplo, abordagens que dependem de alguma base de dados externa como geolocalização, reputação de endereço IP e listas negras, não deveriam ter o mesmo custo que outras abordagens baseadas em características de baixa interação como uma única consulta `DNS`. Da mesma forma, abordagens que combinam diferentes protocolos de rede como `P2P`, `HTTP` e `DNS` devem ser tratadas de maneira distinta daquelas que usam exclusivamente um único protocolo.

Na modelagem inicial do `AFFACO-M`, o peso das arestas entre as abordagens é obtido pela distância euclidiana dos `CTA`. Isso acontece porque em determinado instante, as características são compartilhadas entre si, isto é, ocorre a comutatividade de custos. Por esse motivo, um estudo inicial poderia considerar a intersecção de características entre as abordagens e calcular o custo dessa área criada.

Os parâmetros que influenciam diretamente o funcionamento do processo de correlação de abordagens também devem ser analisados. Por exemplo, qual é o melhor valor entre o número de formiga e a convergência das regras? É possível assumir que as variações estejam associadas ao modelo utilizado para correlação por isso, para cada conjunto de dados é necessário reavaliar esses parâmetros.

Em caso de classificação duvidosa, mesmo que seja possível adotar a estratégia do último voto, seria interessante trabalhar com uma terceira opção de rótulo para o nome de domínio; positivo, negativo e suspeito. No caso suspeito, por exemplo, a tratativa seria reduzir a largura de banda destinada a este endereço e consultar a reputação desse domínio em bases externas como `PhishTank.com`, `MalwareDomains.com` e `Google Safe Browsing`, além de continuar a extração paralela de dados.

Nesses casos, a heurística pode também considerar os algoritmos utilizados pelas abordagens, ou que não haja sobreposição de características em determinada composição da regra.

REFERÊNCIAS

Abadi, M.; Jalili, S. An ant colony optimization algorithm for network vulnerability analysis. *Iranian Journal of Electrical and Electronic Engineering*, v. 2, n. 3, p. 106–120, 2006.

Abu Rajab, M.; Zarfoss, J.; Monrose, F.; Terzis, A. A multifaceted approach to understanding the botnet phenomenon. In: *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, IMC '06, New York, NY, USA: ACM, 2006, p. 41–52 (*IMC '06*,).

Al-Duwairi, B. N.; Al-Hammouri, A. T. Fast flux watch: A mechanism for online detection of fast flux networks. *Journal of Advanced Research*, v. 5, n. 4, p. 473–479, cyber Security, 2014.

Al-Nawasrah, A.; Al-Momani, A.; Meziane, F.; Alauthman, M. Fast flux botnet detection framework using adaptive dynamic evolving spiking neural network algorithm. In: *2018 9th International Conference on Information and Communication Systems (ICICS)*, 2018, p. 7–11.

Almomani, A. Fast-flux hunter: a system for filtering online fast-flux botnet. *Neural Computing and Applications*, p. 1–11, 2016.

Antonakakis, M.; April, T.; Bailey, M.; Bernhard, M.; Bursztein, E.; Cochran, J.; Durumeric, Z.; Halderman, J. A.; Invernizzi, L.; Kallitsis, M.; Kumar, D.; Lever, C.; Ma, Z.; Mason, J.; Menscher, D.; Seaman, C.; Sullivan, N.; Thomas, K.; Zhou, Y. Understanding the mirai botnet. In: *26th USENIX Security Symposium (USENIX Security 17)*, Vancouver, BC: USENIX Association, 2017, p. 1093–1110.

Antonakakis, M.; Perdisci, R.; Dagon, D.; Lee, W.; Feamster, N. Building a dynamic reputation system for dns. In: *Proceedings of the 19th USENIX conference on*

Security, USENIX Security'10, Berkeley, CA, USA: USENIX Association, 2010, p. 18–18 (*USENIX Security'10*,).

Antonakakis, M.; Perdisci, R.; Lee, W.; Vasiloglou, II, N.; Dagon, D. Detecting malware domains at the upper dns hierarchy. In: *Proceedings of the 20th USENIX conference on Security*, SEC'11, Berkeley, CA, USA: USENIX Association, 2011, p. 27–27 (*SEC'11*,).

Antonakakis, M.; Perdisci, R.; Nadji, Y.; Vasiloglou, N.; Abu-Nimeh, S.; Lee, W.; Dagon, D. From throw-away traffic to bots: Detecting the rise of dga-based malware. In: *Proceedings of the 21st USENIX Conference on Security Symposium*, Security'12, Berkeley, CA, USA: USENIX Association, 2012, p. 24–24 (*Security'12*,).

Atluri, A. C.; Tran, V. *Botnets threat analysis and detection* Cham: Springer International Publishing, p. 7–28, 2017.

Baig, A. R.; Shahzad, W. A correlation-based ant miner for classification rule discovery. *Neural Computing and Applications*, v. 21, n. 2, p. 219–235, 2012.

Barbosa, K. R. S. *Detecção de anomalias na internet através da análise do tráfego dns.* Dissertação de Mestrado, Universidade Federal do Amazonas – Instituto de Computação, Universidade Federal do Amazonas, 2010.

Barbosa, K. R. S.; Souto, E.; Feitosa, E.; El-Khatib, K. Identifying and classifying suspicious network behavior using passive dns analysis. In: *The 15th IEEE International Conference on Computer and Information Technology (CIT-2015)*, 2015.

Barbosa, K. R. S.; Souto, E.; Feitosa, E.; gilbert Martins *Livro de minicursos do xiv simpósio brasileiro em segurança da informação e de sistemas computacionais sbseg.*, cáp. Botnets: Características e Métodos de Detecção Através do Tráfego de Rede. SBC, p. 99–144, 2014.

Barr, D. RFC 1912: Common DNS operational and configuration errors. http://www.ietf.org/rfc/rfc1912.txt, 1996.
Disponível em `http://www.ietf.org/\-rfc/\-rfc1912.txt`

Bazrafshan, Z.; Hashemi, H.; Fard, S.; Hamzeh, A. A survey on heuristic malware detection techniques. In: *Information and Knowledge Technology (IKT), 2013 5th Conference on*, 2013, p. 113–120.

Berger, A.; D'Alconzo, A.; Gansterer, W. N.; Pescapé, A. Mining agile {DNS} traffic using graph analysis for cybercrime detection. *Computer Networks*, v. 100, p. 28 – 44, 2016.

Bilge, L.; Sen, S.; Balzarotti, D.; Kirda, E.; Kruegel, C. Exposure: A passive dns analysis service to detect and report malicious domains. *ACM Trans. Inf. Syst. Secur.*, v. 16, n. 4, p. 14:1–14:28, 2014.
Disponível em `http://doi.acm.org/10.1145/2584679`

Binsalleeh, H.; Ormerod, T.; Boukhtouta, A.; Sinha, P.; Youssef, A.; Debbabi, M.; Wang, L. On the analysis of the zeus botnet crimeware toolkit. In: *Privacy Security and Trust (PST), 2010 Eighth Annual International Conference on*, 2010, p. 31–38.

Blum, C.; Roli, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, v. 35, n. 3, p. 268–308, 2003.

Bonabeau, E.; Dorigo, M.; Theraulaz, G. *Swarm intelligence: from natural to artificial systems*. N. 1. Oxford university press, 1999.

Borkar, G. M.; Pund, M. A.; Jawade, P. Implementation of round robin policy in dns for thresholding of distributed web server system. In: *Proceedings of the International Conference & Workshop on Emerging Trends in Technology*, ICWET '11, New York, NY, USA: ACM, 2011, p. 198–201 (*ICWET '11*,).

Born, K.; Gustafson, D. Detecting DNS tunnels using character frequency analysis. *CoRR*, v. abs/1004.4358, 2010.

Caglayan, A.; Toothaker, M.; Drapaeau, D.; Burke, D.; Eaton, G. Behavioral analysis of fast flux service networks. In: *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, CSIIRW '09, New York, NY, USA: ACM, 2009a, p. 48:1–48:4 (*CSIIRW '09*,).

Caglayan, A.; Toothaker, M.; Drapeau, D.; Burke, D.; Eaton, G. Real-time detection of fast flux service networks. In: *Conference For Homeland Security, 2009. CATCH '09. Cybersecurity Applications Technology*, 2009b, p. 285–292.

Calder, M.; Fan, X.; Hu, Z.; Katz-Bassett, E.; Heidemann, J.; Govindan, R. Mapping the expansion of google's serving infrastructure. In: *Proceedings of the 2013*

Conference on Internet Measurement Conference, IMC '13, New York, NY, USA: ACM, 2013, p. 313–326 (*IMC '13*,).

Campbell, S.; Chan, S.; Lee, J. R. Detection of fast flux service networks. In: *Proceedings of the Ninth Australasian Information Security Conference - Volume 116*, AISC '11, Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2011, p. 57–66 (*AISC '11*,).

Castelluccia, C.; Kaafar, M. A.; Manils, P.; Perito, D. Geolocalization of proxied services and its application to fast-flux hidden servers. In: *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '09, New York, NY, USA: ACM, 2009, p. 184–189 (*IMC '09*,).

Chen, C.-M.; Cheng, S.-T.; Chou, J.-H. Detection of fast-flux domains. *Journal of Advances in Computer Networks*, v. 1, n. 2, 2013a.

Chen, C.-M.; Huang, M.-Z.; Ou, Y.-H. Detecting web-based botnets with fast-flux domains. In: Pan, J.-S.; Yang, C.-N.; Lin, C.-C., eds. *Advances in Intelligent Systems and Applications - Volume 2*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013b, p. 79–89.

Chen, H.-H.; Huang, S.-K. b. Lddos attack detection by using ant colony optimization algorithms. *Journal of Information Science and Engineering*, v. 32, n. 4, p. 995–1020, cited By 0, 2016.

Ciornei, I.; Kyriakides, E. Hybrid ant colony-genetic algorithm (gaapi) for global continuous optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, v. 42, n. 1, p. 234–245, 2012.

Colajanni, M.; Gozzi, D.; Marchetti, M. Collaborative architecture for malware detection and analysis. In: Jajodia, S.; Samarati, P.; Cimato, S., eds. *Proceedings of The Ifip Tc 11 23rd International Information Security Conference*, v. 278 de *IFIP - The International Federation for Information Processing*, Springer US, p. 79–93, 2008.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. *Introduction to algorithms, third edition.* 3rd ed. The MIT Press, 2009.

Cover, T. M.; Thomas, J. A. *Elements of information theory 2nd edition.* Wiley Series in Telecommunications and Signal Processing, 2 ed. Wiley-Interscience, 2006.

Daswani, N.; Stoppelman, M. The anatomy of clickbot.a. In: *Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets*, HotBots'07, Berkeley, CA, USA: USENIX Association, 2007, p. 11–11 (*HotBots'07*,).

Davis, C.; Fernandez, J.; Neville, S.; McHugh, J. Sybil attacks as a mitigation strategy against the storm botnet. In: *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, 2008, p. 32–40.

Dietrich, C.; Rossow, C.; Freiling, F.; Bos, H.; van Steen, M.; Pohlmann, N. On botnets that use dns for command and control. In: *Computer Network Defense (EC2ND), 2011 Seventh European Conference on*, 2011, p. 9 –16.

Dipple, A. C. Standing on the shoulders of ants: Stigmergy in the web. In: *Proceedings of the 20th International Conference Companion on World Wide Web*, WWW '11, New York, NY, USA: ACM, 2011, p. 355–360 (*WWW '11*,).

Donis-Díaz, C. A.; Bello, R.; Kacprzyk, J. Using ant colony optimization and genetic algorithms for the linguistic summarization of creep data. In: Angelov, P.; Atanassov, K.; Doukovska, L.; Hadjiski, M.; Jotsov, V.; Kacprzyk, J.; Kasabov, N.; Sotirov, S.; Szmidt, E.; Zadrożny, S., eds. *Intelligent Systems'2014*, Cham: Springer International Publishing, 2015, p. 81–92.

Dorigo, M.; Bonabeau, E.; Theraulaz, G. Ant algorithms and stigmergy. *Future Generation Computer Systems*, v. 16, n. 8, p. 851 – 871, 2000.

Dorigo, M.; Gambardella, L. M. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, v. 1, n. 1, p. 53–66, 1997.

Dorigo, M.; Maniezzo, V.; Colorni, A. Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, v. 26, n. 1, p. 29–41, 1996.

Dorigo, M.; Stützle, T. *Ant colony optimization*. Scituate, MA, USA: Bradford Company, 2004.

Dorigo, M.; Stützle, T. *Handbook of metaheuristics*, cáp. Ant Colony Optimization: Overview and Recent Advances Boston, MA: Springer US, p. 227–263, 2010.

Ebrahimi, N. Families of distributions characterized by entropy. *IEEE Transactions on Information Theory*, v. 47, n. 5, p. 2042–2044, 2001.

Egele, M.; Scholte, T.; Kirda, E.; Kruegel, C. A survey on automated dynamic malware-analysis techniques and tools. *ACM Comput. Surv.*, v. 44, n. 2, p. 6:1–6:42, 2008.

Elz, R.; Bush, R.; Bradner, S.; Patton, M. *Selection and operation of secondary dns servers*. Relatório Técnico, 1997.
Disponível em `http://www.rfc-editor.org/info/rfc2182`

Feamster, N.; Konte, M.; Jung, J. *Fast flux service networks: Dynamics and roles in hosting online scams*. Relatório Técnico, Georgia Institute of Technology, 2008.

Feily, M.; Shahrestani, A.; Ramadass, S. A survey of botnet and botnet detection. In: *Emerging Security Information, Systems and Technologies, 2009. SECURWARE '09. Third International Conference on*, 2009, p. 268 –273.

Freiling, F.; Holz, T.; Wicherski, G. Botnet tracking: Exploring a root-cause methodology to prevent distributed denial-of-service attacks. In: Vimercati, S.; Syverson, P.; Gollmann, D., eds. *Computer Security - ESORICS 2005*, v. 3679 de *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, p. 319–335, 2005.

Futai, Z.; Siyu, Z.; Weixiong, R. Hybrid detection and tracking of fast-flux botnet on domain name system traffic. *China Communications*, v. 10, n. 11, p. 81–94, 2013.

Gajpal, Y.; Abad, P. An ant colony system (acs) for vehicle routing problem with simultaneous delivery and pickup. *Computers & Operations Research*, v. 36, n. 12, p. 3215 – 3223, new developments on hub location, 2009.

García-Martínez, C.; Lozano, M. *Advances in metaheuristics for hard optimization*, cáp. Local Search Based on Genetic Algorithms Berlin, Heidelberg: Springer Berlin Heidelberg, p. 199–221, 2008.

Ge, H.; Hu, T. Genetic algorithm for feature selection with mutual information. In: *Computational Intelligence and Design (ISCID), 2014 Seventh International Symposium on*, 2014, p. 116–119.

Geng, X.; Liu, T.-Y.; Qin, T.; Li, H. Feature selection for ranking. In: *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, New York, NY, USA: ACM, 2007, p. 407–414 (*SIGIR '07*,).

Grzinic, T.; Perhoc, D.; Maric, M.; Vlasic, F.; Kulcsar, T. Croflux – passive dns method for detecting fast-flux domains. In: *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on*, 2014, p. 1376–1380.

Hands, N. M.; Yang, B.; Hansen, R. A. A study on botnets utilizing dns. In: *Proceedings of the 4th Annual ACM Conference on Research in Information Technology*, RIIT '15, New York, NY, USA: ACM, 2015, p. 23–28 (*RIIT '15*,).

Haq, O.; Ahmed, W.; Syed, A. Titan: Enabling low overhead and multi-faceted network fingerprinting of a bot. In: *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on*, 2014, p. 37–44.

Haroun, S. A.; Jamal, B.; et al. A performance comparison of ga and aco applied to tsp. *International Journal of Computer Applications*, v. 117, n. 20, 2015.

Hawkinson, J.; Bates, T. *Guidelines for creation, selection, and registration of an autonomous system (as)*. Relatório Técnico, 1996.
Disponível em `http://www.rfc-editor.org/info/rfc1930`

Hodnefjell, S.; Costa Junior, I. *Classification rule discovery with ant colony optimization algorithm* Berlin, Heidelberg: Springer Berlin Heidelberg, p. 678–687, 2012.

Hoffman, K. L.; Padberg, M.; Rinaldi, G. *Traveling salesman problem* Boston, MA: Springer US, p. 1573–1578, 2013.

Holz, T.; Gorecki, C.; Rieck, K.; Freiling, F. C. Measuring and detecting fast-flux service networks. In: *NDSS*, 2008.

Hsu, C.-H.; Huang, C.-Y.; Chen, K.-T. Fast-flux bot detection in real time. In: *Proceedings of the 13th International Conference on Recent Advances in Intrusion Detection*, RAID'10, Berlin, Heidelberg: Springer-Verlag, 2010, p. 464–483 (*RAID'10*,).

Hu, X.; Knysz, M.; Shin, K. Measurement and analysis of global ip-usage patterns of fast-flux botnets. In: *INFOCOM, 2011 Proceedings IEEE*, 2011, p. 2633–2641.

Hu, X.; Shin, K. G.; Knysz, M. Rb-seeker: Auto-detection of redirection botnets. In: *Network and Distributed System Security Symposium*, 2009.

Huang, S.-Y.; Mao, C.-H.; Lee, H.-M. Fast-flux service network detection based on spatial snapshot mechanism for delay-free detection. In: *Proceedings of the 5th ACM*

Symposium on Information, Computer and Communications Security, ASIACCS '10, New York, NY, USA: ACM, 2010, p. 101–111 (*ASIACCS '10*,).

Hussain, K.; Mohd Salleh, M. N.; Cheng, S.; Shi, Y. Metaheuristic research: a comprehensive survey. *Artificial Intelligence Review*, 2018.

Irwin, B. A network telescope perspective of the conficker outbreak. In: *2012 Information Security for South Africa*, 2012, p. 1–8.

Ishibashi, K.; Toyono, T.; Toyama, K.; Ishino, M.; Ohshima, H.; Mizukoshi, I. Detecting mass-mailing worm infected hosts by mining dns traffic data. In: *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*, MineNet '05, New York, NY, USA: ACM, 2005, p. 159–164 (*MineNet '05*,).

Jensen, R. Performing feature selection with aco. In: *Swarm Intelligence in Data Mining*, Springer, p. 45–73, 2006.

Jiang, C.-B.; Li, J.-S. Exploring global ip-usage patterns in fast-flux service networks. *JCP*, v. 12, n. 4, p. 371–379, 2017.

Jiang, N.; Cao, J.; Jin, Y.; Li, L.; Zhang, Z.-L. Identifying suspicious activities through dns failure graph analysis. In: *Network Protocols (ICNP), 2010 18th IEEE International Conference on*, 2010, p. 144–153.

John, J. P.; Moshchuk, A.; Gribble, S. D.; Krishnamurthy, A. Studying spamming botnets using botlab. In: *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, NSDI'09, Berkeley, CA, USA: USENIX Association, 2009, p. 291–306 (*NSDI'09*,).

John, R.; Cherian, J. P.; Kizhakkethottam, J. J. A survey of techniques to prevent sybil attacks. In: *Soft-Computing and Networks Security (ICSNS), 2015 International Conference on*, 2015, p. 1–6.

Knysz, M.; Hu, X.; Shin, K. Good guys vs. bot guise: Mimicry attacks against fast-flux detection systems. In: *INFOCOM, 2011 Proceedings IEEE*, 2011, p. 1844–1852.

Li, C.; Jiang, W.; Zou, X. Botnet: Survey and case study. In: *Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference on*, 2009, p. 1184–1187.

Li, S.-H.; Kao, Y.-C.; Zhang, Z.-C.; Chuang, Y.-P.; Yen, D. C. A network behavior-based botnet detection mechanism using pso and k-means. *ACM Trans. Manage. Inf. Syst.*, v. 6, n. 1, p. 3:1–3:30, 2015.

Li, W.; Chen, H. Identifying top sellers in underground economy using deep learning-based sentiment analysis. In: *Intelligence and Security Informatics Conference (JISIC), 2014 IEEE Joint*, 2014, p. 64–67.

Lin, H.-C.; Chen, C.-M.; Tzeng, J.-Y. Flow based botnet detection. In: *Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference on*, 2009, p. 1538–1541.

Lin, H.-T.; Lin, Y.-Y.; Chiang, J.-W. Genetic-based real-time fast-flux service networks detection. *Computer Networks*, v. 57, n. 2, p. 501 – 513, botnet Activity: Analysis, Detection and Shutdown, 2013.

Liu, J.; Xiao, Y.; Ghaboosi, K.; Deng, H.; Zhang, J. Botnet: Classification, attacks, detection, tracing, and preventive measures. In: *Proceedings of the 2009 Fourth International Conference on Innovative Computing, Information and Control*, ICICIC '09, Washington, DC, USA: IEEE Computer Society, 2009, p. 1184–1187 (*ICICIC '09*,).

Liu, L.; Vel, O. D.; Han, Q.; Zhang, J.; Xiang, Y. Detecting and preventing cyber insider threats: A survey. *IEEE Communications Surveys Tutorials*, v. 20, n. 2, p. 1397–1417, 2018.

Lombardo, P.; Saeli, S.; Bisio, F.; Bernardi, D.; Massa, D. Fast Flux Detection via Data Mining on Passive DNS Traffic. *ArXiv e-prints*, 2018.

Lombardo, P.; Saeli, S.; Bisio, F.; Bernardi, D.; Massa, D. Fast flux service network detection via data mining on passive dns traffic. In: Chen, L.; Manulis, M.; Schneider, S., eds. *Information Security*, Cham: Springer International Publishing, 2018, p. 463–480.

Luenberger, D. G.; Ye, Y. *Linear and nonlinear programming.* 0884-8289, 3 ed. Springer US, 2008.

Martens, D.; Baesens, B.; Fawcett, T. Editorial survey: swarm intelligence for data mining. *Machine Learning*, v. 82, n. 1, p. 1–42, 2011.

Martinez-Bea, S.; Castillo-Perez, S.; Garcia-Alfaro, J. Real-time malicious fast-flux detection using dns and bot related features. In: *PST*, 2013, p. 369–372.

McCarty, B. Botnets: Big and bigger. *IEEE Security and Privacy*, v. 1, n. 4, p. 87–90, cited By (since 1996)41, 2003.

McGrath, D. K.; Kalafut, A.; Gupta, M. Phishing infrastructure fluxes all the way. *IEEE Security Privacy*, v. 7, n. 5, p. 21–28, 2009.

Mockapetris, P. RFC 1034: Domain names - concepts and facilities. http://www.ietf.org/rfc/rfc1034.txt, 1987.

Mokhtari, N.; Ghezavati, V. Integration of efficient multi-objective ant-colony and a heuristic method to solve a novel multi-objective mixed load school bus routing model. *Applied Soft Computing*, v. 68, p. 92 – 109, 2018.

Morales, J.; Al-Bataineh, A.; Xu, S.; Sandhu, R. Analyzing dns activities of bot processes. In: *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on*, 2009, p. 98–103.

Mowbray, M.; Hagen, J. Finding domain-generation algorithms by looking at length distribution. In: *Software Reliability Engineering Workshops (ISSREW), 2014 IEEE International Symposium on*, 2014, p. 395–400.

Musashi, Y.; Kumagai, M.; Kubota, S.; Sugitani, K. Detection of kaminsky dns cache poisoning attack. In: *Intelligent Networks and Intelligent Systems (ICINIS), 2011 4th International Conference on*, 2011, p. 121–124.

Nakamura, Y.; Kanazawa, S.; Inamura, H.; Takahashi, O. Classification of unknown web sites based on yearly changes of distribution information of malicious ip addresses. In: *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2018, p. 1–4.

Nazario, J.; Holz, T. As the net churns: Fast-flux botnet observations. In: *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, 2008, p. 24–31.

Parpinelli, R. S.; Lopes, H. S.; Freitas, A. A. Data mining with an ant colony optimization algorithm. *IEEE Transactions on Evolutionary Computation*, v. 6, n. 4, p. 321–332, 2002a.

Parpinelli, R. S.; Lopes, H. S.; Freitas, A. A. Data mining with an ant colony optimization algorithm. *IEEE Transactions on Evolutionary Computation*, v. 6, p. 321–332, 2002b.

Passerini, E.; Paleari, R.; Martignoni, L.; Bruschi, D. Fluxor: Detecting and monitoring fast-flux service networks. In: Zamboni, D., ed. *Detection of Intrusions and Malware, and Vulnerability Assessment*, v. 5137 de *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, p. 186–206, 2008.

Paul, T.; Tyagi, R.; Manoj, B. S.; B., T. Fast-flux botnet detection from network traffic. In: *2014 Annual IEEE India Conference (INDICON)*, 2014, p. 1–6.

Peng, T.; Leckie, C.; Ramamohanarao, K. Survey of network-based defense mechanisms countering the dos and ddos problems. *ACM Comput. Surv.*, v. 39, n. 1, 2007.

Perdisci, R.; Corona, I.; Dagon, D.; Lee, W. Detecting malicious flux service networks through passive analysis of recursive dns traces. *Computer Security Applications Conference, Annual*, v. 0, p. 311–320, 2009.

Ranga, V.; Mandhar, V. Ant colony based ip traceback scheme. *International Journal of Information Technology*, 2018.

Rekhter, Y.; Li, T.; S. Hares, E. *A border gateway protocol 4 (bgp-4)*. Relatório Técnico, 2006.
Disponível em `http://www.rfc-editor.org/info/rfc4271`

Rodríguez-Gómez, R. A.; Maciá-Fernández, G.; García-Teodoro, P. Survey and taxonomy of botnet research through life-cycle. *ACM Comput. Surv.*, v. 45, n. 4, p. 45:1–45:33, 2013.

Royal, P. Analysis of the kraken botnet. *Damballa*, v. 9, 2008.

Salusky, W.; Danford, R. Know your enemy: Fast-flux service networks. an ever changing enemy. *The Honeynet Project*, 2008.
Disponível em `http://www.honeynet.org/papers/ff`

Schiller, C.; Binkley, J. *Botnets: The killer web applications*. Syngress Publishing, 2007.

Shelokar, P.; Jayaraman, V.; Kulkarni, B. An ant colony approach for clustering. *Analytica Chimica Acta*, v. 509, n. 2, p. 187 – 195, 2004.

Shin, S.; Gu, G. Conficker and beyond: A large-scale empirical study. In: *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, New York, NY, USA: ACM, 2010a, p. 151–160 (*ACSAC '10*,).

Shin, S.; Gu, G. Conficker and beyond: A large-scale empirical study. In: *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, New York, NY, USA: ACM, 2010b, p. 151–160 (*ACSAC '10*,).

Shin, S.; Lin, R.; Gu, G. Cross-analysis of botnet victims: New insights and implications. In: Sommer, R.; Balzarotti, D.; Maier, G., eds. *Recent Advances in Intrusion Detection*, v. 6961 de *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, p. 242–261, 2011.

Shukla, A. S.; Maurya, R. Entropy-based anomaly detection in a network. *Wireless Personal Communications*, 2018.

Sinclair, G.; Nunnery, C.; Kang, B. B. The waledac protocol: The how and why. In: *2009 4th International Conference on Malicious and Unwanted Software (MALWARE)*, 2009, p. 69–77.

Sinha, P.; Boukhtouta, A.; Belarde, V.; Debbabi, M. Insights from the analysis of the mariposa botnet. In: *Risks and Security of Internet and Systems (CRiSIS), 2010 Fifth International Conference on*, 2010, p. 1–9.

Soltanaghaei, E.; Kharrazi, M. Detection of fast-flux botnets through dns traffic analysis. *Scientia Iranica. Transaction D, Computer Science & Engineering, Electrical*, v. 22, n. 6, p. 2389, 2015.

Souza, A.; Barbosa, K. R. S.; Feitosa, E. Identificando spam no twitter através da anáise empírica dos trending topics brasil. In: *SBSeg 2013 - WTICG (2013)*, 2013, p. 394–403.

Stalmans, E.; Hunter, S.; Irwin, B. Geo-spatial autocorrelation as a metric for the detection of fast-flux botnet domains. In: *Information Security for South Africa (ISSA), 2012*, 2012, p. 1–7.

Stalmans, E.; Irwin, B. A framework for dns based detection and mitigation of malware infections on a network. In: *2011 Information Security for South Africa*, 2011, p. 1–8.

Stamos, K.; Pallis, G.; Vakali, A.; Katsaros, D.; Sidiropoulos, A.; Manolopoulos, Y. Cdnsim: A simulation tool for content distribution networks. *ACM Trans. Model. Comput. Simul.*, v. 20, n. 2, p. 10:1–10:40, 2010.

Stevanovic, M.; Pedersen, J. M.; D'Alconzo, A.; Ruehrup, S. A method for identifying compromised clients based on dns traffic analysis. *International Journal of Information Security*, p. 1–18, 2016.

Stone-Gross, B.; Cova, M.; Cavallaro, L.; Gilbert, B.; Szydlowski, M.; Kemmerer, R.; Kruegel, C.; Vigna, G. Your botnet is my botnet: Analysis of a botnet takeover. In: *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, New York, NY, USA: ACM, 2009, p. 635–647 (*CCS '09*,).

Studer, R. Economic and technical analysis of botnets and denial-of-service attacks. *Communication systems IV*, p. 19, 2011.

Su, A.-J.; Choffnes, D. R.; Kuzmanovic, A.; Bustamante, F. E. Drafting behind akamai: Inferring network conditions based on cdn redirections. *IEEE/ACM Trans. Netw.*, v. 17, n. 6, p. 1752–1765, 2009.

Sundaram, R. K. *A first course in optimization theory.* 1996.

Suwa, S.; Yamai, N.; Okayama, K.; Nakamura, M.; Kawano, K.; Gada Spam mail discrimination system based on behavior of dns servers associated with urls. In: *Applications and the Internet (SAINT), 2012 IEEE/IPSJ 12th International Symposium on*, 2012, p. 381–386.

Talbi, E.-G. *Metaheuristics: From design to implementation.* Wiley Publishing, 2009.

Thangamani, C.; Chidambaram, M. An efficient hybrid of continuous ant colony optimization and weighted crossover genetic algorithm for optimal solution. *Fuzzy Systems*, v. 10, n. 1, 2018.

Tiirmaa-Klaar, H.; Gassen, J.; Gerhards-Padilla, E.; Martini, P. Botnets: How to fight the ever-growing threat on a technical level. In: *Botnets*, SpringerBriefs in Cybersecurity, Springer London, p. 41–97, 2013.

Truong, D.-T.; Cheng, G. Detecting domain-flux botnet based on dns traffic features in managed network. *Security and Communication Networks*, v. 9, n. 14, p. 2338–2347, sCN-15-0084.R2, 2016.

Wang, P.; Lin, H. T.; Wang, T. S. A revised ant colony optimization scheme for discovering attack paths of botnet. In: *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*, 2011, p. 918–923.

Wang, P.; Wang, T. C.; Kuo, P.-T.; Wang, C. P. An analysis model of botnet tracking based on ant colony optimization algorithm. In: *Networked Computing and Advanced Information Management (NCM), 2010 Sixth International Conference on*, 2010, p. 606–611.

Wu, J.; Zhang, L.; Liang, J.; Qu, S.; Ni, Z. A comparative study for fast-flux service networks detection. In: *Networked Computing and Advanced Information Management (NCM), 2010 Sixth International Conference on*, 2010, p. 346–350.

Wu, W.; Alvarez, J.; Liu, C.; Sun, H.-M. Bot detection using unsupervised machine learning. *Microsystem Technologies*, v. 24, n. 1, p. 209–217, 2018.

Yadav, S.; Reddy, A. K. K.; Reddy, A. L. N.; Ranjan, S. Detecting algorithmically generated domain-flux attacks with dns traffic analysis. *IEEE/ACM Trans. Netw.*, v. 20, n. 5, p. 1663–1677, 2012.

Ye, Y.; Li, T.; Adjeroh, D.; Iyengar, S. S. A survey on malware detection using data mining techniques. *ACM Comput. Surv.*, v. 50, n. 3, p. 41:1–41:40, 2017.

Yu, B.; Smith, L.; Threefoot, M. *Semi-supervised time series modeling for real-time flux domain detection on passive dns traffic* Cham: Springer International Publishing, p. 258–271, 2014.

Zeng, B.; Chen, B. Sybilaco: Ant colony optimization in defending against sybil attacks in the wireless sensor network. In: *2010 International Conference on Computer and Communication Technologies in Agriculture Engineering*, 2010, p. 357–360.

Zhao, D.; Traore, I. P2p botnet detection through malicious fast flux network identification. In: *2012 Seventh International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, 2012, p. 170–175.

Zhao, Y.; Jin, Z. Quickly identifying ffsn domain and cdn domain with little dataset. In: *4th International Conference on Mechatronics, Materials, Chemistry and Computer Engineering (ICMMCCE 2015)*, 2015, p. 1999–2004.

Zhou, C. V.; Leckie, C.; Karunasekera, S.; Peng, T. A self-healing, self-protecting collaborative intrusion detection architecture to trace-back fast-flux phishing domains. In: *Network Operations and Management Symposium Workshops, 2008. NOMS Workshops 2008. IEEE*, 2008, p. 321–327.

Zhou, S. A survey on fast-flux attacks. *Inf. Sec. J.: A Global Perspective*, v. 24, n. 4-6, p. 79–97, 2015.

İlker Özçelik; Brooks, R. R. Deceiving entropy based dos detection. *Computers & Security*, v. 48, p. 234 – 245, 2015.

A

Abordagem EntropyFFNet

Este Anexo apresenta a construção da abordagem `EntropyFFNet` a partir da modificação do trabalho proposto por Zhao e Jin (2015). A principal ideia é utilizar os conceitos de Teoria da Informação para substituir um atributo ausente durante a realização desse trabalho. Enquanto no trabalho original os autores enviam diferentes solicitações HTTP para o nome de domínio suspeito para obter a volatilidade de tempo de acesso, a abordagem `EntropyFFNet` utiliza os próprios endereços IP já coletados em outras fases da abordagem, o que resulta na redução de comunicação entre os agentes de monitoramento e o domínio suspeito. Para definir qual algoritmo deveria ser utilizado nessa nova abordagem, foi realizado um estudo comparativo entre os algoritmos `CART`, `Naïve Bayes`, `Logistic Regression` e `SVM`. Os resultados mostram que na proposta original a acurácia de detecção é de 89.1%, enquanto na abordagem `EntropyFFNet` esse valor é de 94.52%.

A.1 EntropyNET

A falta de clareza na descrição do tempo de coleta dos atributos utilizados, ausência de alguns atributos na base de dados e o processo empírico para extração de características dificultaram o desenvolvimento das abordagens selecionadas neste trabalho. Um típico exemplo desses problemas foi o trabalho proposto em (Zhao e Jin, 2015), onde os autores analisam o tráfego `DNS` em combinação do tráfego `HTTP` dentro de uma janela de monitoramento estabelecida em 4 horas.

A Tabela 1.1 apresenta os atributos utilizados originalmente pelos autores, dos quais $\mathcal{F}_1$ representa a quantidade de endereços IP associados ao nome de domínio, $\mathcal{F}_2$ denota a

quantidade de classes B (`a.b`) associados ao nome de domínio. Finalmente, $\mathcal{F}_3$ representa a volatilidade de tempo de acesso para o domínio malicioso. Tal atributo é obtido a partir do cálculo da média dos tempos de conexão de consultas enviadas através do tráfego `HTTP`. A principal ideia que sustenta esse atributo é que hosts infectados operam com algum tipo de serviço de proxy entre o cliente e o verdadeiro host malicioso (*mothership*). Portanto, o tempo de acesso ao conteúdo solicitado pode mudar dependendo do tipo de conexão que o host infectado possui.

ID	**Descrição**
$\mathcal{F}_1$	Quantidade de IP associados
$\mathcal{F}_2$	Quantidade de redes classe B
$\mathcal{F}_3$	Volatilidade de tempo de acesso

Tabela 1.1: Atributos utilizados em (Zhao e Jin, 2015).

Vale lembrar que os domínios maliciosos não estavam mais disponíveis durante a realização deste trabalho, por esse motivo, o atributo $\mathcal{F}_3$ foi substituído por outra característica rede que pudesse ser baseada no tráfego `DNS`. Os autores utilizam o cálculo da volatilidade para avaliar as mudanças ou dispersões dos dados e, por isso, esse cálculo também pode ser considerado como uma medida de dispersão (Ebrahimi, 2001).

Entre as medidas de dispersão, o cálculo da Entropia (Teoria da Informação) é a métrica que possui maior capilaridade em diferentes áreas do conhecimento (Cover e Thomas, 2006), tanto em redes de computadores (Shukla e Maurya, 2018; İlker Özçelik e Brooks, 2015) como análise do tráfego `DNS` (Barbosa, 2010; Born e Gustafson, 2010; Dietrich et al., 2011). Portanto, é razoável assumir que o cálculo da entropia seja uma métrica válida em relação à proposta original dos autores.

A entropia pode ser aplicada em diferentes características do tráfego `DNS` como a seção de resposta e seção adicional. No entanto, neste trabalho, essa métrica é aplicada para identificar a dispersão de sistemas autônomos associados ao nome de domínio. Devido à falta de informação no trabalho original sobre quais seções são utilizadas para coleta de tráfego DNS, nesta proposta de abordagem, denominada como `EntropyNET`, os endereços IP são extraídos da seção de resposta e da seção adicional.

O objetivo é identificar domínios que estejam concentrados ou que são gerenciados pelo mesmo sistema autônomo, mesmo que possuam um número considerável de endereços IP. Um típico exemplo desse cenário é o nome de domínio `instagram.com`, onde 8 endereços IP (seção de resposta) oriundos de redes 7 redes distintas, pertencem à dois sistemas autônomos de uma mesma empresa.

Tabela 1.2: Estudo empírico comparativo da abordagem `EntropyFFNet`.

Métrica	`CART`	`Naïve Bayes`	`Logistic Regression`	`SVM`
Acurácia	94.52	83.96	93.91	94.21
E. Classificação	5.48	16.04	6.09	5.79
Recall	96.20	97.20	95.40	95.20
Specificity	92.78	70.31	92.37	93.20
FPR	7.22	29.69	7.63	6.80
Precisão	93.22	77.14	92.80	93.52
F-Measure	94.69	86.02	94.08	94.35
AUC	94.49	83.75	93.89	94.20

Para demonstrar o desempenho da abordagem proposta, considere um estudo *empírico comparativo* entre os algoritmos `CART`, `Naïve Bayes`, `Logistic Regression` e `SVM`, descrito na Tabela 1.2. Nessa comparação, os algoritmos foram implementados sem qualquer tipo de ajuste de parâmetro utilizando a biblioteca `scikit-learn` da linguagem de programação `Python`. O objetivo é entender o comportamento de diferentes classificadores a partir das características selecionadas. Na proposta original de (Zhao e Jin, 2015), os resultados denotam 89.1% e 10.9% de acurácia e erro de classificação, respectivamente. Em contrapartida, abordagem `EntropyFFNet` analisa os dados disponíveis da base de dados para todos o período de coleta, o que pode ter contribuído com melhores resultados.

No geral, o classificador `CART` apresentou resultados semelhantes ao classificador `SVM` considerando as métricas de acurácia, erro de classificação, precisão, `f-measure` e área abaixo da curva. As diferenças mais significativas estão no `recall`, especificidade e taxa de falso positivo. Em comparação com o trabalho de (Zhao e Jin, 2015), é necessário investigar o comportamento da abordagem `EntropyFFNet` dentro de uma janela de monitoramento de 4 horas para conclusões mais contundentes. Por isso, como trabalho futuros, é necessário fazer uma análise mais detalhada do `Custo Total de Atributos` associado à abordagem `EntropyFFNet`.

B

Avaliação da Função BurdenMethod

No algoritmo `AFFACO-P` caso os resultados não convirjam para um consenso, então é necessário iniciar o conceito do último voto. Neste trabalho, o conceito do último voto é realizado pela função `BurdenMethod`. Este Anexo descreve um estudo sofre essa função descrita no Capítulo 4 deste trabalho. Para implementação desse método, duas novas abordagens foram definidas usando os algoritmos `Logistic Regression` e `Random Forest`. A principal objetivo dessa função é utilizar todos os atributos coletados anteriormente por cada abordagem. Isto é, no pior caso a classificação do nome de domínio contemplá todas as perspectivas em um só método. Os resultados mostram que em ambos os algoritmos o nível de acurácia é de 99%.

B.1 BurdenMethod

A principal ideia que sustenta a utilização do último voto é que as porções isoladas que representam o comportamento do nome de domínio não foram suficiente para determinar sua natureza, seja pela falta de diversidade dos atributos ou efetividade das abordagens. O objetivo na verdade é que nesse ponto da classificação, todas as características já foram extraídas e aplicadas por diferentes abordagens, no entanto, ao serem combinadas em um grade vetor de características, é possível ter uma visão do problema como todo. A Tabela 2.1 ilustra o resultado da classificação da base de dados pela adoção dos algoritmos de aprendizagem de máquina `Logistic Regression` (LR) e `Random Forest` (RF), ambos desenvolvidos usando o módulo `scikit-learn`.

Tabela 2.1: Comparação de desempenho entre os classificadores `Logistic Regression` e `Random Forest` usando 21 característica.

Métrica	**LR**	**RF**
Acurácia	99.49	99.70
E. Classificação	0.51	0.30
Recall	100.00	99.40
Specificity	98.97	100.00
FPR	1.03	0.00
Precisão	99.01	100.00
F-Measure	99.50	99.70
AUC	99.48	99.70

C

Abordagens para Classificação de FFSN

O presente anexo descreve um resumo das abordagens utilizadas e apresenta o processo de validação realizado para classificar redes de serviço de fluxo rápido. No entanto, é importante destacar que algumas características adotadas foram adequadas para atender a disponibilidade de informações na base de dados utiliza neste trabalho. Ao final da seção é realizada uma discussão sobre a substituição de algumas características e a proposta de uma nova abordagem.

Tabela 3.1: Resumo das Abordagens Utilizadas.

ID	Referência	Algoritmo	Acurácia	CTA
A_1	(Caglayan et al., 2009b)	Naïve Bayes	88.12%	372
A_2	(Chen et al., 2013a)	Função Linear	82.84%	156
A_3	(Grzinic et al., 2014)	Função Linear	64.42%	154
A_4	(Passerini et al., 2008)	Naïve Bayes	79.70%	396
A_5	(Holz et al., 2008)	Função Linear	70.66%	85
A_6	(Huang et al., 2010)	Naïve Bayes	78.46%	118
A_7	EntropyFFNet	CART	94.52%	72

A Tabela 3.1 apresenta um identificador utilizado para análise, o nome das abordagens utilizadas, o algoritmo de classificação adotado, acurácia de detecção e o Custo Total dos Atributos (CTA). Cada abordagem listada aqui foi analisada observando as métricas informadas anteriormente.

Cada abordagem foi desenvolvida na linguagem de programação Python e os algoritmos de aprendizagem de máquina são baseados no módulo scikit-learn[1] que facilita o uso de tais tecnologias.

[1] http://scikit-learn.org/

Para garantir que a implementação dessas abordagens em `Python` estavam corretas, o *framework* `Weka` foi utilizado como parâmetro. Por exemplo, em `Python` a acurácia da abordagem proposta em (Passerini et al., 2008) denota 79.70%, enquanto o resultado no *framework* `Weka`, também usando o algoritmo `Naïve Bayes`, representa valores praticamente idênticos, 79.81%.

Esse processo de comparação permitiu confirmar que os dados obtidos podem ser reproduzidos tanto pelos algoritmos em `scikit-learn` como no `Weka`. Para os demais algoritmos, os valores obtidos seguem a mesma analogia, com exceção das abordagens que utilizam funções lineares como A_2, A_3 e A_5, que representam algoritmos de classificação próprios e que não estão incorporados no *framework* `Weka`.

A partir do momento que os resultados das abordagens foram validados, algumas funcionalidades da ferramenta `Weka` foram incorporadas como a possibilidade de gerar modelos usando técnicas de validação cruzada (10 partições) ou a partir de uma única partição (*holdouts*). Com isso, foi possível incorporar esses recursos de classificação no algoritmo de correlação `AFFACO-M` e no sistema `AFFACO-P`.

É importante deixar claro que as abordagens selecionadas são utilizadas em dois momentos. No primeiro momento, o processo de correlação de abordagens utiliza seus resultados de classificação para modificarem os parâmetros do modelo formal, descritos no Capítulo 4. No segundo momento, o classificador utiliza as mesmas abordagens para classificar os nomes de domínio da base de dados.

A abordagem identificada como $\mathtt{A}_7$ representa uma nova proposta de detecção de redes de serviço de fluxo rápido. Tal abordagem é reflexo do processo de incorporação no grafo de correlação dos trabalhos descritos na literatura. O Anexo A apresenta uma análise preliminar dos resultados obtidos na abordagem `EntropyFFNet`.

Dado à limitação da base de dados, como explicitado anteriormente, o processo de seleção de abordagens neste trabalho levou em consideração aquelas que utilizam o maior número de características do tráfego `DNS`. No entanto, alguns atributos foram adaptadas para que a abordagem fosse utilizada no processo de correlação de abordagens.

As primeiras modificações foram realizadas no trabalho proposto em (Passerini et al., 2008), que originalmente demanda 9 características para classificar um nome de domínio, conforme ilustrado na Tabela 3.2. O atributo $\mathcal{F}_2$ representa o nome da empresa que oferece o serviço de `Whois` quando o nome de domínio é registrado. Porém, de maneira diferente da data de registro do domínio, a qual ainda pode ser identificada observando diferentes caracteres como $\{/, :, -\}$, o nome da empresa pode ser qualquer texto entre os resultados da consulta `Whois`. Em alguns `TLDs` o termo `Registra` é utilizado, mas não há um padrão formal para esse objetivo.

ID	Descrição
$\mathcal{F}_1$	Data de registro do domínio
$\mathcal{F}_2$	Nome do *Registra*
$\mathcal{F}_3$	Número distinto de registros do tipo `A`
$\mathcal{F}_4$	Tempo de vida (`TTL`) do domínio
$\mathcal{F}_5$	Número distinto de redes
$\mathcal{F}_6$	Número distinto de `ASN`
$\mathcal{F}_7$	Número distinto de `FQDN`
$\mathcal{F}_8$	Número redes (empresas) associadas ao domínio
$\mathcal{F}_9$	Número de organizações

Tabela 3.2: Atributos utilizados em (Passerini et al., 2008).

Durante o processo de extração dessas características, foi possível observar que diferentes servidores de `Whois` não disponibilizam tal informação por essa razão, esse atributo foi removido do vetor de características que representam um nome de domínio. Outro atributo de rede removido denota o número de organizações ($\mathcal{F}_8$) associado ao domínio. Para obter esses dados, o comando `traceroute` deve ser empregado e a partir do seu resultado, é feito o cálculo de quantas redes estão associadas ao domínio.

Por exemplo, considere a saída do comando `traceroute` para um dos endereços IP (`98.138.252.38`) que respondem pelo nome de domínio `yahoo.com`, conforme ilustra a Figura 3.1. Cada endereço IP denota a sequência de roteadores que devem ser acessados antes de alcançar o IP associado ao domínio em questão. É possível observar que cada roteador dessa sequência pertence a uma única empresa ou nome de domínio, o que segundo os autores é uma característica positiva para domínios legítimos.

```
10  * * *
11  ae-1.pat2.brz.yahoo.com (200.152.160.135)  72.894 ms  71.651 ms  68.566 ms
12  xe-0-0-9.pat1.nyc.yahoo.com (216.115.101.14)  245.037 ms  245.554 ms  215.609 ms
13  ae-5.pat1.dcz.yahoo.com (216.115.104.196)  234.657 ms  234.647 ms  234.624 ms
14  UNKNOWN-216-115-96-X.yahoo.com (216.115.96.0)  262.328 ms  248.867 ms  252.285 ms
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  media-router-fp1.prod.media.vip.ne1.yahoo.com (98.138.252.38)  312.315 ms  306.823 ms  407.097 ms
```

Figura 3.1: Exemplo da saída do comando `traceroute`.

Embora essa premissa se aplique para alguns domínios, não é possível assumir que todo domínio legítimo apresente o mesmo comportamento. Por exemplo, considere a saída do comando `traceroute` para os IP que atendem o domínio `ufam.edu.br` (`200.129.163.8`) e `twitter.com` (`104.244.42.65`), ilustrados na Figura 3.2a) e 3.2b), respectivamente.

Em ambos os casos, não foi possível identificar qual nome de domínio está associado aos endereços IP, o que dificulta o processo de análise de comportamento do domínio.

```
traceroute to ufam.edu.br (200.129.163.8), 30 hops max, 60 byte packets
 1  gateway (192.168.0.1)  1.945 ms  2.595 ms  3.932 ms
 2  * * *
...
16  * * *
17  * * *
18  200.129.156.126 (200.129.156.126)  506.563 ms  506.324 ms  507.608 ms

a) Traceroute enviado para domínio: ufam.edu.br

 traceroute to twitter.com (104.244.42.193), 30 hops max, 60 byte packets
 1  gateway (192.168.0.1)  1.679 ms  1.899 ms  1.890 ms
 2  * * *
 ...
 6  196.43.9.130 (196.43.9.130)  248.403 ms  243.853 ms  244.210 ms
 7  80.81.192.10 (80.81.192.10)  244.194 ms  220.859 ms  220.811 ms
 8  * 104.244.42.193 (104.244.42.193)  221.162 ms  219.912 ms

b) Traceroute enviado para domínio: twitter.com
```

Figura 3.2: Exemplo da saída do comando `traceroute` para o IP que responde pelo domínio a) `ufam.com.br` e b) `twitter.com`.

Pelos motivos expostos, a proposta de (Passerini et al., 2008), neste trabalho, utiliza 7 atributos para classificação de redes de serviço de fluxo rápido. Além disso, vale notar que os resultados individuais implementados aqui são semelhantes às conclusões que os autores originais encontram.

C.1 Avaliação das Abordagens

Cada abordagem selecionada foi desenvolvida seguindo as informações disponíveis nos trabalhos publicados, dos quais não ofereciam dados ou ajustes de parâmetros que pudessem favorecer o desempenho dos classificadores. Por esse motivo, após constatação dos resultados encontrados[2], foram realizados novos testes usando a ferramenta `Weka` para confrontar os dados dispostos na Tabela 3.3.

Para entender o processo de classificação por meio do algoritmo `AFFACO-M`, primeiramente, cada abordagem selecionada foi aplicada na classificação dos nomes de domínio

[2]Resultados encontrados por meio dos algoritmos disponíveis no módulo `scikit-learn`.

da base de dados e seus resultados distribuídos na Tabela 3.3. Na avaliação individual as seguintes abordagens foram estudadas (Caglayan et al., 2009b), (Chen et al., 2013a), (Grzinic et al., 2014), (Passerini et al., 2008), (Holz et al., 2008), (Huang et al., 2010) e `EntropyFFNet`, como $\mathtt{A}_1$, $\mathtt{A}_2$, $\mathtt{A}_3$, $\mathtt{A}_4$, $\mathtt{A}_5$, $\mathtt{A}_6$ e $\mathtt{A}_7$, respectivamente.

Tabela 3.3: Avaliação individual das abordagens usando validação cruzada de 10 partições (*folds*).

Métrica	A_1	A_2	A_3	A_4	A_5	A_6	A_7
Acurácia	88.12	82.84	61.42	79.70	70.66	78.48	94.52
E. Classificação	11.88	17.16	38.58	20.30	29.34	21.52	5.48
Recall	89.80	89.40	100.00	97.00	98.40	92.40	96.20
Specificity	86.39	76.08	21.65	61.86	42.06	64.12	92.78
FPR	13.61	23.92	78.35	38.14	57.94	35.88	7.22
Precisão	87.18	79.40	56.82	72.39	63.65	72.64	93.22
F-Measure	88.47	84.10	72.46	82.91	77.30	81.34	94.69
AUC	88.10	82.74	60.82	79.43	70.23	78.26	94.49

A Tabela 3.4 apresenta as abordagens selecionadas e ordenadas em ordem crescente de `CTA`. O menor custo está associado à abordagem `EntropyFFNet` e o maior custo é pertence à (Passerini et al., 2008). As seções a seguir consideram tanto os valores da acurácia como `CTA` para cada abordagem selecionada. Esses valores são utilizados pelo algoritmo `AFFACO-M` para gerar regras de classificação de redes de serviço de fluxo rápido.

Tabela 3.4: Custo e acurácia individual das abordagens

Abordagens	**Acurácia**	`CTA`
EntropyFFNet	94.52%	72
(Holz et al., 2008)	70.66%	85
(Huang et al., 2010)	78.46%	118
(Grzinic et al., 2014)	64.42%	154
(Chen et al., 2013a)	82.84%	156
(Caglayan et al., 2009b)	88.12%	372
(Passerini et al., 2008)	79.70%	396

A falta de clareza na descrição do tempo de coleta dos atributos utilizados e o processo empírico para extração de características dificultaram o desenvolvimento dessas abordagens. Por exemplo, diferentes trabalhos (Caglayan et al., 2009b; Grzinic et al., 2014) utilizam os endereços IP da resposta `DNS`, mas não determinam de qual seção é realizado o estudo, tampouco é informado se houve consulta a partir de diferentes pontos do globo. Além disso, como citado anteriormente, algumas abordagens utilizam atributos que não estavam disponíveis durante a realização desta tese de doutorado.

D

Metodologia de Extração de Características

Neste trabalho foram selecionadas 7 abordagens que, na sua maioria, utilizam características do tráfego `DNS` (Caglayan et al., 2009b), (Chen et al., 2013a), (Grzinic et al., 2014), (Passerini et al., 2008), (Holz et al., 2008), (Huang et al., 2010) e (Zhao e Jin, 2015). Essa decisão foi motivada devido à base de dados possuir exclusivamente consultas e respostas `DNS`. Além disso, os domínios registrados na base já haviam sido removidos de operação, então abordagens que utilizam tráfego `HTTP` são descartadas ou adaptadas. O Capítulo de Resultados fornece mais detalhes sobre a composição da base de dados utilizada.

Os dados armazenados na base de dados estão no formato do utilitário `DIG`, o qual fornece três seções de informações: resposta (`ANSWER`), autoridade (`AUTHORITY`) e adicional (`ADDITIONAL`), conforme ilustra a Figura 4.1 para o nome de domínio `yahoo.com.br`.

Dentro de cada seção existe uma divisão entre colunas, onde o campo divisor pode ser um caractere de *espaço vazio* ou *tab*. Na seção de resposta, cada coluna pode ser lida (da esquerda para direita) da seguinte maneira: a primeira coluna faz referência ao nome de domínio; a segunda denota o TTL; terceira a classe; quarta o tipo de registro de recurso; e a última é o endereço IP. Essas informações também podem ser aplicadas à seção adicional no entanto, a seção autoridade define o nome do servidor que possui autoridade sob o domínio ao invés do endereçamento IP. Além disso, dependendo do tipo de registro de recurso, as informações podem divergir na quarta e quinta coluna (Mockapetris, 1987).

Para ilustrar o processo de extração de característica, considere um atributo que é definido pelo número de redes classe C (`a.b.c`) associadas ao nome de domínio

```
;; QUESTION SECTION:
;yahoo.com.br.                  IN      A

;; ANSWER SECTION:
/ahoo.com.br.           300     IN      A       212.82.100.151
/ahoo.com.br.           300     IN      A       124.108.115.101
/ahoo.com.br.           300     IN      A       106.10.248.151
/ahoo.com.br.           300     IN      A       98.136.103.24
/ahoo.com.br.           300     IN      A       74.6.136.151

;; AUTHORITY SECTION:
/ahoo.com.br.           44949   IN      NS      ns5.yahoo.com.
/ahoo.com.br.           44949   IN      NS      ns2.yahoo.com.
/ahoo.com.br.           44949   IN      NS      ns4.yahoo.com.
/ahoo.com.br.           44949   IN      NS      ns1.yahoo.com.
/ahoo.com.br.           44949   IN      NS      ns3.yahoo.com.

;; ADDITIONAL SECTION:
ns1.yahoo.com.          292835  IN      A       68.180.131.16
ns2.yahoo.com.          292835  IN      A       68.142.255.16
ns3.yahoo.com.          292835  IN      A       203.84.221.53
ns4.yahoo.com.          292835  IN      A       98.138.11.157
ns5.yahoo.com.          292835  IN      A       119.160.253.83
ns1.yahoo.com.          121747  IN      AAAA    2001:4998:130::1001
ns2.yahoo.com.          121747  IN      AAAA    2001:4998:140::1002
ns3.yahoo.com.          121747  IN      AAAA    2406:8600:b8:fe03::1003
```

Figura 4.1: Formato de dados da ferramenta `DIG` (Fonte: Próprio Autor).

ilustrado na Figura 4.1. Nesse caso, o domínio `yahoo.com.br` possui 5 redes associadas: $\{212.82.100, 124.108.115, 106.10.248, 98.136.103, 74.6.136\}$ na seção de resposta. No entanto, dependendo da abordagem, é necessário também processar a seção de adicional para obter o total de classes C que respondem pelo nome de domínio.

Vale ressaltar que a ferramenta `DIG` não garante que as três seções estejam completas em muitos casos, apenas a seção de resposta possui dados para extração, pois é possível configurar um tipo de resposta mínima de informações[1] para reduzir a carga de dados nos servidores de nome. Nesses casos, é necessário realizar novas consultas para obter os dados das seções pendentes quando o nome de domínio ainda está em operação.

Para tentar identificar nomes de domínios que podem ser utilizados para análise, um *parser* na linguagem de programação *Perl* foi desenvolvido para tratar os dados da base de dados, ou seja, que na verdade é um arquivo texto contendo inúmeras consultas `DNS` que foram obtidas pela ferramenta `DIG`. Após identificação dos nomes de domínios válidos, ocorre de fato a extração das características baseadas no tráfego `DNS`. No entanto, nem todas as características estão disponíveis nesse arquivo. Algumas características como total de Sistemas Autônomos (`AS`) e total de países associados ao nome de domínio devem

[1] `http://www.zytrax.com/books/dns/ch7/queries.html#minimal-responses`

ser obtidas em bases ou acessos externos, tal como serviço de tradução do IP para ASN `IP-to-ASN`[2], ou serviço de informação geolocalização de IP[3].

Além do protocolo `DNS`, é necessário consultar o protocolo `Whois` para extrair a data de registro do nome de domínio e o nome da empresa que ofereceu o serviço de registro de nomes (Passerini et al., 2008; Wu et al., 2010). Embora o uso do protocolo `Whois` seja simples, é importante lembrar que a disposição e os campos de dados não seguem qualquer padrão, e a consulta deve ser direcionada para o servidor `Whois` que responde pelo `TLD`. Por exemplo, o domínio `ufam.edu.br` tem seus dados armazenados em `whois.registro.br`, o qual é responsável pelo `TLD` do Brasil `.br`.

A extração da data de registro do nome de domínio é realizada por meio de consultas enviadas por um cliente `whois`, desenvolvido na linguagem `Perl`. Devido o procolo `whois` não possuir um padrão que determine a estrutura ou formato das informações acessíveis, esse cliente busca padrões de datas que representam o registro do nome de domínio. A Tabela 4.1 ilustra o nome de domínio, servidor `Whois` que responde pela zona de domínio e o formato da data de registro, o qual pode conter diferentes espaços, representações de data e idioma.

Tabela 4.1: Exemplo de data de registro obtidos de servidores `Whois`.

Domínio	**Whois**	**Padrão**
ufam.edu.br	whois.registro.br	created: 19971217
debian.org	whois.pir.org	Creation Date: 1999-03-10T05:00:00Z
monster.ca	whois.cira.ca	Creation date: 2000/10/22
bolivia.bo	whois.nic.bo	Fecha de registro: 2008-03-26

De uma maneira geral, existem 1540 domínios de primeiro nível incluindo domínios de códigos de países e domínios genéricos[4]. Cada `TLD` está associado a um servidor de `Whois`, por isso, antes de extrair qualquer informação, é necessário identificar o servidor que possui esses dados. No entanto, mesmo consultando os servidores diretamente, alguns `TLDs` como `.DE`, `.AU`, `.CO.UK`, `.CL` entre outros, não permitem ou subtraem esses dados de seus resultados, seja por política de privacidade ou segurança. Para esses casos, é preciso atribuir um valor que descrimine que tal nome de domínio não possui essas informações.

No total foram mapeados **41 atributos**, sendo que apenas **23** são utilizados pelas abordagens selecionadas. Essas características foram identificadas através de revisão da literatura, descrita no Capítulo 3 e de experimentos iniciais de análise de tráfego DNS. A seguir são descritos com mais detalhes algumas dessas dessas características utilizadas.

[2] `http://www.team-cymru.org/IP-ASN-mapping.html`
[3] `http://dev.maxmind.com/geoip/legacy/geolite/`
[4] `https://www.iana.org/domains/root/db`

C_1: Número de endereços IP encontrados na seção de resposta a partir de uma única consulta DNS. Por exemplo, o nome de domínio nfl.com responde apenas um único endereço IP enquanto o domínio youtube.com retorna 16 endereços IP.

C_2: O tempo de vida (TTL) identificado em uma resposta DNS determina o máximo de tempo (em segundos) que uma informação pode ficar armazenada no resolvedor de nomes do cliente. O TTL baixo permite que o atacante mantenha rotação de endereços IP com maior frequência. Diferentes trabalhos têm identificado que valores de até 1800 segundos são frequentemente utilizados para manter o domínio FFSN acessível (Lin et al., 2013; Nazario e Holz, 2008).

C_3: Número de endereços IP encontrados na seção adicional. Para um nome de domínio ser alcançado na Internet é necessário que *servidores de nomes* sejam designados e que possam responder às consultas DNS do domínio em questão. O objetivo é identificar se existem alterações no endereço IP do servidor que responde pelo domínio.

C_4: A data de registro de um nome de domínio é utilizada para identificar a idade do domínio, pois alguns trabalhos descrevem a duração de vida de domínios FFSN em dias ou semanas, podendo permanecer em operação por até 5 semanas no máximo (Chen et al., 2013a; Passerini et al., 2008).

C_5: Quantidade de redes classe C na seção de resposta. O objetivo é identificar domínios que possuem um número representativo de endereços de redes distintos. Por exemplo, seja d = twitter.com e seja $\Psi_d = \{104.244.42.129, 104.244.42.193\}$ o conjunto de endereços IP do domínio d. Então, seja f uma função que identifica a quantidade de redes classe C distintas em d, logo $f(\Psi_d) = 1$, pois todos IP estão na mesma rede 104.244.42. No entanto, se essa função for aplicada no conjunto de endereços IP que respondem pelo nome domínio legítimo debian.org, o valor encontrado é $\Psi_d = 6$, pois esse domínio utiliza RRDNS para distribuir a carga entre os endereços.

C_6: Quantidade de rede classe B na seção adicional. Por definição do protocolo DNS (Mockapetris, 1987), a distribuição de redes nesta seção deve apresentar pelo menos dois endereços IP em redes distintas para garantir disponibilidade de acesso. A Figura 4.2 ilustra um exemplo da seção adicional para o nome de domínio uol.com.br, o qual é atendido por 3 endereços IP sendo que dois endereços fazem parte do mesmo segmento de rede 200.147 e um na rede 200.221.

```
;; ADDITIONAL SECTION:
eliot.uol.com.br.   86400   IN  A   200.221.11.98
borges.uol.com.br.  86400   IN  A   200.147.255.105
charles.uol.com.br. 86400   IN  A   200.147.38.8
```

Figura 4.2: Seção adicional da resolução de nomes para o domínio `uol.com.br`

C_7: Quantidade de `ASN` na seção de resposta. Provedores de Internet e grandes Corporações podem definir políticas de roteamento para gerenciar endereços IP que estão sob seu controle, essas políticas fazem parte de um Sistema Autônomo - `AS` (Hawkinson e Bates, 1996). Cada `AS` opera internamente de maneira independente de outros sistemas autônomos e a comunicação entre esses sistemas é realizada pelo protocolo de roteamento de borda (Rekhter et al., 2006). A partir de um endereço IP, é possível identificar o número do sistema autônomo (`ASN`) que gerencia as informações sobre qualquer endereço IP.

```
;; ANSWER SECTION:
instagram.com. 60 IN A 54.173.161.156
instagram.com. 60 IN A 54.172.125.33
instagram.com. 60 IN A 54.172.44.240
instagram.com. 60 IN A 54.152.56.115
instagram.com. 60 IN A 54.164.46.213
instagram.com. 60 IN A 54.164.255.125
instagram.com. 60 IN A 54.173.152.112
instagram.com. 60 IN A 54.165.39.209
```

Figura 4.3: Endereços IP que respondem pelo nome de domínio `instagram.com`.

A identificação da quantidade de `ASN` na seção de reposta tem como objetivo avaliar se a distribuição de endereços IP pertencem ao mesmo `ASN` (Provedor). Por exemplo, seja o nome de domínio $d = instagram.com$ e $f(\Psi_d) = 8$ a quantidade de redes distintas, conforme ilustra a Figura 4.3. Vale acrescentar que, embora as redes sejam distintas, os endereços IP que respondem pelo domínio d fazem parte do mesmo `ASN`. No entanto, no caso de redes maliciosas baseadas em fluxo rápido, os endereços IP fazem parte de sistemas autônomos distribuídos ao redor do globo, pois o objetivo dessas redes é garantir disponibilidade (Al-Duwairi e Al-Hammouri, 2014).

C_8: Quantidade de `ASN` na seção adicional. Essa característica permite identificar servidores DNS respondem pelo nome de domínio, mas que não estão associados ao `ASN` do domínio.

C_9: A *intersecção* entre os endereços IP da seções de resposta e adicional tem como objetivo identificar práticas não convencionais e suspeitas de operação de nome domínio. O objetivo é identificar nomes de domínios onde um endereço IP é responsável tanto pelo armazenamento de conteúdo como por respostas sobre a zona de domínio. Por exemplo, redes maliciosas podem utilizar essa estratégia para manter os requisitos mínimos de registro de um nome de domínio, onde o mesmo IP é atribuído para hospedar o conteúdo da página e o serviço de nomes, conforme ilustra a Figura 4.4 que possui apenas um endereço IP `109.169.57.109` para atender os serviços de rede[5].

```
;; ANSWER SECTION:
cialis5mg-20mg.com.     14400   IN      A       109.169.57.109

;; AUTHORITY SECTION:
cialis5mg-20mg.com.     172798  IN      NS      ns1.onlinecheapest-kamagra.net.
cialis5mg-20mg.com.     172798  IN      NS      ns2.onlinecheapest-kamagra.net.

;; ADDITIONAL SECTION:
ns1.onlinecheapest-kamagra.net. 172799 IN A     109.169.57.109
ns2.onlinecheapest-kamagra.net. 172799 IN A     109.169.57.109
```

Figura 4.4: Domínio malicioso que utiliza único endereço IP em múltiplos serviços do domínio (Fonte: Próprio Autor).

É importante deixar claro que a definição do protocolo do DNS (Mockapetris, 1987) não restringe as operações realizadas por servidores de nome ou servidores que hospedam conteúdo no entanto, existem recomendações de boas práticas de configuração para que servidores de nome sejam distribuídos ao redor do globo, permitindo que as informações da zona estejam disponível aos clientes pela Internet (Elz et al., 1997).

C_{10}: O cálculo de similaridade entre endereços IP listados na seção de resposta do tráfego DNS tem como objetivo identificar hosts localizados em diferentes redes atendendo um nome de domínio. Entre os cálculos de similaridades, as abordagens podem utilizar a *Distância Euclidiana* ou *Índice de Jaccard*. A Distância Euclidiana é definida como a raiz quadrada da soma da diferença entre x e y em suas respectivas dimensões, conforme ilustra a Equação D.1.

[5]Domínio obtido da base de spam: `http://phishtank.com/`

$$distance = \sqrt{\sum_{j=1}^{n}(x-y)^2} \tag{D.1}$$

Seja n o número de valores no vetor de características, então $x - y$, é a diferença normalizada do j-ésimo elemento do vetor. Para diferença entre dois endereços IP no seguinte formato {`a.b.c`} (classe C), a distância euclidiana pode ser encontrada pela seguinte Equação D.2:

$$d(IP_i, IP_{i-1}) = \sqrt{\sum_{j=1}^{3}(X_{i,j} - X_{i-1,j})^2} \tag{D.2}$$

Onde, IP_i e IP_{i-1} são dois endereços IPv4, que terão seus 3 primeiros octetos como características, $IP_i = X_{i,1}, X_{i,2}, X_{i,3}$ e $IP_{i-1} = X_{i-1,1}, X_{i-1,2}, X_{i-1,3}$. A distância euclidiana dos seguintes endereços IP IP_i = 192.168.1 e IP_{i-1} = 192.168.2, é dada por $d(IP_i, IP_{i-1}) = 1$. Os valores obtidos nesse cálculo podem variar e terem diferentes interpretações por cada abordagem selecionada.

Outra maneira também utilizada para identificar a similaridade entre objetos é o *Índice de Jaccard*, o qual pode variar entre 0 e 1, sendo 1 para objetos iguais (Perdisci et al., 2009).

C_{11}: Domínios CDN utilizam como artifício o registro de recurso `CNAME` para redirecionar o cliente para infraestrutura CDN, como ilustra a Figura 4.5. Por exemplo, o nome de domínio `www.yahoo.com` é um *nome canônico* para o endereço endereço `fd-fp3.wg1.b.yahoo.com`, que por sua vez é atendido pelo conjunto de endereços IP `98.139.180.149` e `98.139.183.24`. No caso do nome de domínio `www.instagram.com`, existe uma referência `CNAME` (`black.ish.instagram.com`) que aponta para outro tipo `CNAME` (`instagram.c10r.facebook.com`), que por sua vez é atendido pelo endereço IP `31.13.73.52`.

```
;; ANSWER SECTION:
www.yahoo.com. 207 IN CNAME fd-fp3.wg1.b.yahoo.com.
fd-fp3.wg1.b.yahoo.com. 25 IN A 98.139.180.149
fd-fp3.wg1.b.yahoo.com. 25 IN A 98.139.183.24
```

Figura 4.5: Registro de Recurso do tipo `CNAME` utilizado em infraestrutura CDNs.

O objetivo é identificar se o nome de domínio possui o registro de recurso do tipo `CNAME`, assim como a quantidade de registros dessa natureza na zona de domínio. Vale notar que o uso do registro do tipo `CNAME` não é exclusividade de nomes de domínio legítimos no entanto, para que um domínio baseado em `FFSN` utilize esse registro, o nome canônico deve estar associado a um conjunto de endereços de troca rápida, caso contrário, tais endereços IP podem ser identificados e cadastrados em listas negras.

C_{12}: Quantidade de nomes suspeitos associados aos IP. Validar se o nome reverso dos endereços IP (PTR) está associado à redes de Internet banda larga, como `dialup, customer, velox`, pois tais nomes indicariam que usuários domésticos estão sendo utilizados em *botnets*.

C_{13}: A razão entre número de endereços IP da classe C pelo número total de endereços IP permite identificar a variação ou a dispersão de endereços IP que respondem por um nome de domínio dentro de uma janela de monitoramento (Perdisci et al., 2009).

C_{14}: Redes de distribuição de conteúdo distribuem endereços IP conforme a localização geográfica da origem da consulta, conforme demonstrado anteriormente. Esse comportamento também pode acontecer caso o *botmaster* tenha controle mais avançado sobre a operação de sua rede e hosts, como número grande de endereços IP disponíveis por localização e disponibilidade, respectivamente (Knysz et al., 2011).

Para ilustrar a distribuição de endereços IP utilizados por uma `CDN`, o nome de domínio `instagram.com` foi consultado em dois momentos distintos T_1 e T_2, conforme Tabela 4.2.

Tempo T_1	Tempo T_2
52.202.183.204	52.22.186.20
52.202.219.253	52.22.200.81
52.21.37.241	52.22.207.179
52.22.102.96	52.22.230.122
52.22.151.190	52.22.41.220
52.22.159.222	52.22.71.95
52.22.179.84	52.4.142.143
52.22.186.110	52.4.16.217

Tabela 4.2: Endereços IP que respondem pelo nome de domínio `instagram.com`.

A variação de endereços IP pode ser obtida pela diferença entre os tempos $T_1 - T_2$. O resultado mostra que esse nome de domínio possui um padrão de respostas, a cada

nova consulta são retornados 8 endereços IP. Embora essa diferença se aproxime de algumas redes maliciosas (Al-Duwairi e Al-Hammouri, 2014), é possível validar a legitimidade do domínio a partir da análise do número de endereços IP da classe C ou da quantidade de ASN, como citado nas características anteriores.

Outra alternativa para encontrar a taxa de crescimento de endereços IP é baseada na divisão do total de endereços únicos, pelo total de consultas enviadas para um domínio dentro de uma janela de monitoramento. Dessa forma, quanto menor o valor mais estável é a variação de novos endereços IP encontrados para esse nome de domínio (Perdisci et al., 2009). Por exemplo, numa janela de monitoramento de 20 minutos foram enviadas 40 consultas (30 segundos de TTL) para o domínio `instagram.com`, sendo 124 endereços únicos de um total de 252 endereços IP, resultando numa taxa de $\frac{124}{40} = 3.5$, enquanto num domínio maliciosa essa taxa pode ser 5 vezes maior.

C_{15}: A quantidade de novos números de sistemas autônomos (ASN) observados durante a janela de monitoramento W permite identificar se o nome de domínio utilizou redes distribuídas no globo para manter seu nome de domínio em operação. Por exemplo, apesar do número de endereços IP ter mudado para o nome de domínio `instagram.com`, conforme ilustrado na Tabela 4.2, esses endereços ainda pertencem ao mesmo `ASN` (`14618`).

C_{16}: Total de endereços IP do tipo `NS` que responderam pelo nome de domínio durante uma janela de monitoramento. A distribuição de endereços IP em redes válidas (CDNs e RRDNS) deve ter um limite que ao ser atingido, é possível conhecer todos os IP que respondem por um nome de domínio. No entanto, diversos autores mostram que o total de endereos IP de uma rede `FFSN` tende a crescer enquanto o domínio estiver ativo (Al-Duwairi e Al-Hammouri, 2014; Hu et al., 2011; Passerini et al., 2008).

C_{17}: Histórico suspeito. *Botmasters* podem utilizar o mesmo conjunto ou uma parcela de endereços IP para responderem por diferentes nomes de domínios maliciosos. Por exemplo, a Figura 4.6 ilustra 3 nomes de domínio que foram obtidos a partir de uma base de spam e utilizam o endereço IP `109.169,57.109` tanto para hospedar o conteúdo como para operar o servidor de nomes.

O objetivo dessa característica é identificar se um nome de domínio está utilizando qualquer endereço IP marcado anteriormente como suspeito ou desconhecido. Mais precisamente, seja $f(d)$ a função que verifica se qualquer endereço IP associado

```
1) ;; ANSWER SECTION:
prednisone-order-20mg.com.    21600    IN    A    109.169.57.109
...
2) ;; ANSWER SECTION:
pills-viagrageneric.com.    21600    IN    A    109.169.57.109
...
3) ;; ANSWER SECTION:
cialis5mg-20mg.com.com.    21600    IN    A    109.169.57.109
```

Figura 4.6: Nomes de domínio que são hospedados pelo mesmo endereço IP.

ao domínio d já foi utilizado em outras atividades suspeitas como operação ao mesmo tempo de hospedagem e servidor de nomes; cadastrado em listas negra; ou classificado como malicioso pelo algoritmo AFFACO-M.

C_{18}: Média de endereços IP por ASN. Métrica utilizada para saber quantos endereços IP associados à determinado nome de domínio fazem parte do mesmo ASN, como ilustra a Equação D.3. Tal método é utilizado por (Chen et al., 2013a) para identificar se domínios maliciosos estão imitando comportamento de domínio legítimo.

$$w = \frac{1}{\text{total ASN}} * \sum_{i=1}^{\text{total ASN}} \frac{1}{\text{Total IP por ASN}} \tag{D.3}$$

É importante ressaltar que outras características são combinações dos atributos citados em diferentes janelas de monitoramento ou seções da resposta DNS. No entanto, nem todos os atributos utilizados pelas abordagens selecionadas estão disponíveis na base de dados portanto, foi necessário fazer um ajuste e manipulação de atributos para que fosse possível incluir a abordagem no Grafo de Correlação. O Anexo A descreve com mais detalhes o processo de substituição de característica utilizado neste trabalho.