Capítulo 3

Botnets: Características e Métodos de Detecção Através do Tráfego de Rede

Kaio R. S. Barbosa, Gilbert B. Martins, Eduardo Souto, Eduardo Feitosa

Abstract

The passive monitoring and analysis enable the identification of suspected communication patterns of infected machines (bots) in network traffic. These hosts are often associated with malicious computers networks (botnets) used to spread malicious code (virus and worms) and denial of service (DDoS) attacks. Although researchers have been creating and evolving techniques to mitigate this type of application, attackers are also developing new mechanisms to hinder such an analysis. Thus, the technique of passive analysis is a viable alternative to find suspicious behaviors in unknown network traffic (zero-day) or when a worm was not fully explored by reverse engineering techniques. This chapter presents the strategies used to detect botnets through passive analysis of network traffic, separated in function of communication protocols commnly used to the implementation of command and control botnet channels, as well as a basic set of tools to support the process to detect this kind of malicious activity.

Resumo

O monitoramento e análise passiva permitem identificar padrões de comunicação suspeitos de máquinas infectadas (bots) no tráfego de rede. Tais hosts frequentemente estão associados a redes de computadores maliciosas (botnets) utilizadas para proliferação de códigos maliciosos (vírus e worms) e ataques de negação de serviço (DDoS). Embora os pesquisadores venham criando e evoluindo técnicas para mitigar esse tipo de aplicação, os atacantes também desenvolvem novos mecanismos para dificultar tal análise. Assim, a técnica de análise passiva é uma alternativa viável para encontrar comportamentos suspeitos no tráfego de rede desconhecidos (zero-day) ou quando um worm ainda não foi totalmente explorado pelas técnicas de engenharia reversa. Este Capítulo apresenta as estratégias utilizadas para detecção de botnets através da análise passiva do tráfego de rede, separadas em função dos protocolos de comunicação comumente utilizados para

implementação dos canais de comando e controle de botnets, bem como um conjunto básico de ferramentas para apoio ao processo de detecção deste tipo de atividade maliciosa.

3.1. 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. Hoje em dia, os sistemas computacionais são infectados principalmente para o roubo de informações ou para o acesso aos recursos computacionais disponíveis nestes sistemas. Em geral, a meta primária dos atacantes é obter o controle total dos recursos sem afetar o comportamento usual do sistema até que estes desejem fazer uso dos recursos que agora estão sob seu controle. Neste modelo de atividade maliciosa, quanto maior for a distribuição do *malware*, maior será o poder computacional à disposição de seus criadores.

Formalmente, um *malware* é definido como um software que "deliberadamente", cumpre a intenção prejudicial de um atacante [Egele et al. 2008]. Contudo, o maior problema dos *malware*s tradicionais está relacionado ao procedimento de manutenção dos mesmos [Tiirmaa-Klaar et al. 2013]. Como os atacantes frequentemente não possuem acesso remoto aos hosts, é dificil corrigir o funcionamento do software malicioso. Por esse motivo, os *malwares* atuais são projetados para permitir que uma entidade externa mantenha o máximo de controle dos dispositivos infectados.

As máquinas contaminadas por *malware*, tipicamente passam a fazer parte de uma rede maliciosa, denominada de *botnet*, e são comumente conhecidas pela alcunha de zumbis ou apenas pelo nome de *bot*. O conceito de *botnets* está associado ao conjunto de máquinas comprometidas que permitem ao atacante controlar remotamente os recursos computacionais e realizar atividades fraudulentas ou ilícitas [McCarty 2003b, Freiling et al. 2005].

O *malware Storm*, que se espalhou por mais de dois milhões de computadores, deu ao seu proprietário um poder computacional agregado superior ao de um supercomputador [Colajanni et al. 2008]. Tal poder computacional desse e de outros *malwares* tem sido usado pelos atacantes, também chamados de *botmasters*, para promover ataques aos mais diversos provedores de serviço, instituições governamentais e outras empresas cujo modelo de negócios é fortemente atrelado ao uso da Internet.

Para reforçar a efetividade dos ataques contra alvos maiores, os *botmasters* necessitam aumentar o número de *bots* e, para isso, empregam diferentes técnicas para espalhar códigos maliciosos. No início, o procedimento para tornar uma máquina infectada não era completamente automatizado e necessitava de interação humana como clicar em um link, que conduzia ao download de códigos maliciosos, ou abrir um e-mail com um software malicioso anexado. Atualmente, este procedimento tornou-se cada vez mais automatizado. Por exemplo, `SDBot` [McFee 2003] pode se propagar usando técnicas como compartilhamento de arquivos, redes `P2P`, *backdoors* deixadas por *worms* ou ainda explorar vulnerabilidades comuns do sistema operacional *Windows*. Além disso, os ataques e a capacidade de comunicação do *bots* tem sido melhorados. Por exemplo, uma máquina infectada com o `Agabot` [Schiller and Binkley 2007] pode participar de

ataques de negação de serviços distribuídos (DDoS), servir como proxy para difusão de spam, criar túneis para outras redes de dados (GRE tunneling) e capturar senhas.

Este Capítulo tem como objetivo fornecer os subsídios para o entendimento das *botnets*. O foco é dado as técnicas de análise passiva do tráfego de rede e o seu emprego na detecção de *botnets*. Para tanto, o Capítulo empregará um enfoque teórico apoiado por exemplos práticos para garantir o entendimento do processo de identificação de *botnets*, desde a estratégia para coletar o tráfego até a confirmação de comportamento suspeito.

3.1.1. Organização do Capítulo

Este Capítulo está organizado da seguinte maneira: a Seção 3.2 apresenta um histórico e as definições 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*. A Seção 3.3 discute métodos e estratégias adotadas para detecção de *botnets* através da análise do tráfego de rede. Nessa Seção, os principais protocolos de rede utilizados para comunicação com os *bots* são descritos. A Seção 3.4 aborda algumas técnicas e ferramentas para coleta e análise do tráfego. Além disso, esta Seção apresenta exemplos de bibliotecas de programação de rede que podem auxiliar no processo de coleta e análise. Por fim, a Seção 3.5 apresenta as considerações finais e os problemas em aberto sobre o tema.

3.2. Botnets

3.2.1. Entendendo o problema

A preocupação com o correto funcionamento na comunicação em redes de computadores tem origem desde a concepção inicial da computação distribuída, na década de 80 [Shoch and Hupp 1982]. Uma aplicação com problema de funcionamento, mesmo que não intencional, poderia interromper a operação e a comunicação da rede, ocasionando problemas governamentais, financeiros e empresariais. Diante desses problemas, em 1987 o primeiro trabalho direcionado à pesquisa de aplicações maliciosas ou vírus de computador foi publicado [Cohen 1987].

Para Cohen, a definição inicial de um vírus de computador denota que um programa pode infectar outros programas com uma versão evoluída ou melhorada de si mesmo. A preocupação de Cohen era que se um vírus pudesse infectar outros computadores, através de redes de computadores ao redor do mundo, a comunicação da sociedade moderna teria inúmeros problemas. Tal fato pôde ser constatado quase 20 anos após a publicação de Cohen, quando Furnel e Ward [Furnell and Ward 2004] apresentaram uma visão geral da evolução dos vírus de computadores, demonstrando que o emprego de técnicas de persuasão para que usuários iniciassem um vírus, assim como previsto por Cohen, já não eram necessárias.

Assim, ao invés de esperar a intervenção do usuário, um software malicioso (*malware*) tornou-se capaz de infectar outras aplicações, explorando as vulnerabilidades do sistema operacional ou de softwares instalados. Além disso, alguns *malwares* já empregavam técnicas sofisticadas (metamorfismo de código, por exemplo) para mudar o seu código a medida que se propagavam ou deixavam portas escondidas abertas (*backdoor*) para que

o computador fosse utilizado para outros serviços, tais como encaminhador de e-mail, servidor de proxy e ataques de negação de serviço (DoS - *Denial of Service*).

Um marco nesse sentido foi o *malware Code Red*, que em 2001 infectou centenas de milhares de computadores ligados à Internet em um período de 24 horas [Berghel 2001]. Além da infecção, tal *malware* lançava ataques de negação de serviço, em datas específicas, contra o site da Casa Branca (*whitehouse.gov*) nos EUA. No entanto, para encontrar uma nova vítima, o *Code Red* utilizava a geração aleatória de IP, o que muitas vezes resultava em duas ou mais infecções de um mesmo host [Moore et al. 2002]. Outra característica relevante é que este *malware* instalava um *backdoor* que permitia que um host comprometido fosse reaproveitado em outros tipos de ataques.

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

Inicialmente, como proposta, a função de um *bot* era o gerenciamento e controle de canais *IRC*. 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 and Stoppelman 2007] ou aluguel de *botnets* [Studer 2011]. As seções seguintes definem o conceito, os componentes, arquiteturas e protocolos utilizados nas *botnets*.

3.2.2. 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 [McCarty 2003b, Freiling et al. 2005]. Tais máquinas utilizam um software chamado de *bot* (da palavra robô), o qual liga os computadores infectados à uma infraestrutura de Comando e Controle (C&C).

Esta infraestrutura permite que os *bots* se conectem à uma entidade de controle, que pode ser centralizada ou distribuída. Para comunicação com os *bots*, um ou mais protocolos de comunicação são utilizados pelos operadores da rede. Tais estratégias permitem que a *botnet* continue operando 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].

Para controlar as operações de uma *botnet* é necessário uma entidade externa, definida como *botmaster* [Stone-Gross et al. 2009]. Um *botmaster* coordena as ações realizadas por cada *bot*, incluindo estratégias de ataques como negação de serviço e envio de spam em massa. A Figura 3.1 apresenta os principais componentes e interações do uso de *botnets*. Além dos componentes já citados, uma *botnet* deve possuir vetores de propagação (*malwares*) capazes de infectar novos dispositivos. 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].

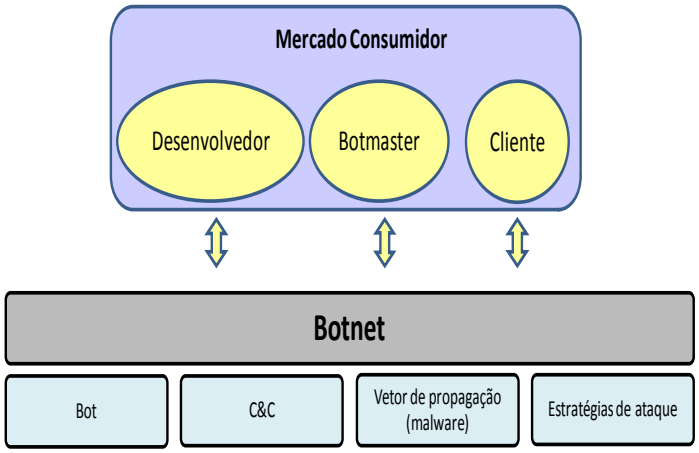

Figura 3.1. Principais componentes e interações de uma *botnet*.

Na autopropagação, o *bot* busca na rede outros dispositivos com vulnerabilidades que possam ser exploradas e que permitam acesso remoto. Por outro lado, na propagação por indução, 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, o host 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 novos métodos de infecção através de comunidades na Internet que pesquisam e vendem softwares maliciosos no mercado negro. 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 comprar aplicações maliciosas pré-compiladas, 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*.

Embora a definição de *botnets* possa atender as principais tecnologias atuais, o termo *botnet* pode ser refinado para abranger novas tendências e arquiteturas computacionais. Por exemplo, recentemente pesquisadores de uma empresa de segurança detectaram que um ataque de spam em massa tinha origem em roteadores, televisores e um refrigerador [ProofPoint 2014]. Tais dispositivos conectados fazem parte da Internet das Coisas (IoT - *Internet of Things*) e, portanto, uma vez infectados podem ser definidos como *bot* das coisas (ou *thingbots*). Além disso, a proliferação de aplicações maliciosas para ambiente móveis (por exemplo, *smartphones*) mostra que qualquer dispositivo conectado à Internet pode ser considerado um *bot* em potencial. Desta forma, esse trabalho define uma *botnet* como um conjunto de dispositivos eletrônicos comprometidos que são controlados remotamente por um ou mais operadores de *bots* (*botmasters*).

3.2.3. Evolução dos Ataques

A evolução de estratégias e protocolos usados nas *botnets* pode ser observada em diversos trabalhos [Peng et al. 2007, Feily et al. 2009, Rodríguez-Gómez et al. 2013]. Em 2003,

os *bots* utilizavam o protocolo `IRC` para entrar em contato com o C&C e os vetores de propagação englobavam o uso de mensagens spam, cavalos de Troia e vírus [McCarty 2003a]. Em meados de 2004, a exploração de vulnerabilidades nos navegadores Internet, especialmente o *Internet Explorer* da *Microsoft*, permitia que mais máquinas fossem comprometidas por *worms*. No ano seguinte, os ataques passaram a ser direcionados a redes de menor escala, permitindo aos atacantes ações como roubo de informações, fraudes e extorsões.

Em 2006, os ataques direcionados a usuários domésticos continuaram a crescer devido à exploração de vulnerabilidades nos navegadores *Internet Explorer* e *Firefox*. No ano de 2007, os EUA registraram os maiores números de servidores de comando e controle enquanto que a China o maior número de máquinas infectadas [Turner et al. 2007].

A mudança de estratégia dos ataques, em 2008, com foco para aplicações Web, permitiu que um número expressivo de computadores fossem atacados a partir de sites legítimos com problemas de vulnerabilidades como *cross-site scripts*, *phishing* e componentes de terceiros inseguros (*plugins*). Tal mudança partiu do princípio que esses ataques são mais difíceis de serem identificados por filtros da rede.

Os anos seguintes apresentaram um crescimento no número de máquinas comprometidas pertencentes a países emergentes. Enquanto em 2007 e 2008, o EUA possuía o maior número de máquinas infectadas, em 2010, o Brasil assumiu a terceira posição da lista dos países com maior número de atividades maliciosas. Uma razão para esse crescimento foi a melhoria da infraestrutura de banda larga de Internet no Brasil, permitindo que atacantes utilizem as máquinas remotamente por mais tempo [Fossi et al. 2010]. Outro fato relevante em 2010 foi a divulgação do kit de desenvolvimento do *malware* Zeus [Binsalleeh et al. 2010], o que permitiu que atacantes com pouca habilidade de programação pudessem criar novas *botnets*.

Em 2011 foi observada uma diversificação nos ataques em diferentes áreas [Fossi et al. 2011]. Em comparação com o ano de 2010, os ataques Web tiverem um aumento de 93%. Os ataques direcionados a dispositivos móveis também registraram um crescimento de 42% e mais de um milhão de *bots* foram identificados sob o controle da *botnet Rustock* [Fossi et al. 2011, Thonnard and Dacier 2011]. Além disso, também foram identificadas ameaças direcionadas a alvos industriais e corporativos, como *worm Stuxnet* [Rebane 2011] e *Hydraq* [Ferrer et al. 2010].

Apesar do número de spam ter reduzido em 2012 [CISCO 2013], o número de ataques e máquinas comprometidas não seguiu o mesmo comportamento devido ao uso de kits de *malware* que facilitaram a criação de novas *botnets*. Além disso, os ataques com objetivo de espionagem industrial ganharam mais evidência [Virvilis et al. 2013].

Finalmente, em 2013, a tendência em ataques à alvos pré-determinados continuou em alta, principalmente contra empresas de pequeno porte. Além disso, a tendência do aumento no número de ameaças para dispositivos móveis foi comprovada e o avanço de ameaças persistentes avançadas (APT - *Advanced Persistent Threat*) em ambientes corporativos [CISCO 2014].

3.2.4. Ciclo de vida dos bots

Para que uma *botnet* continue operando é importante que novos hosts sejam constantemente recrutados, uma vez que *bots* identificados por sistemas de detecção são comumente cadastrados em algum tipo de lista negra [Ishibashi et al. 2005]. Por isso, identificar hosts vulneráveis e comprometê-los é uma atividade vital 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 dos *bots*. Muitos trabalhos exploram o ciclo de vida das *botnets* 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 3.2 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 dos vetores de propagação, 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* para, finalmente, ingressar no canal de comando e controle (*Passo 4*).

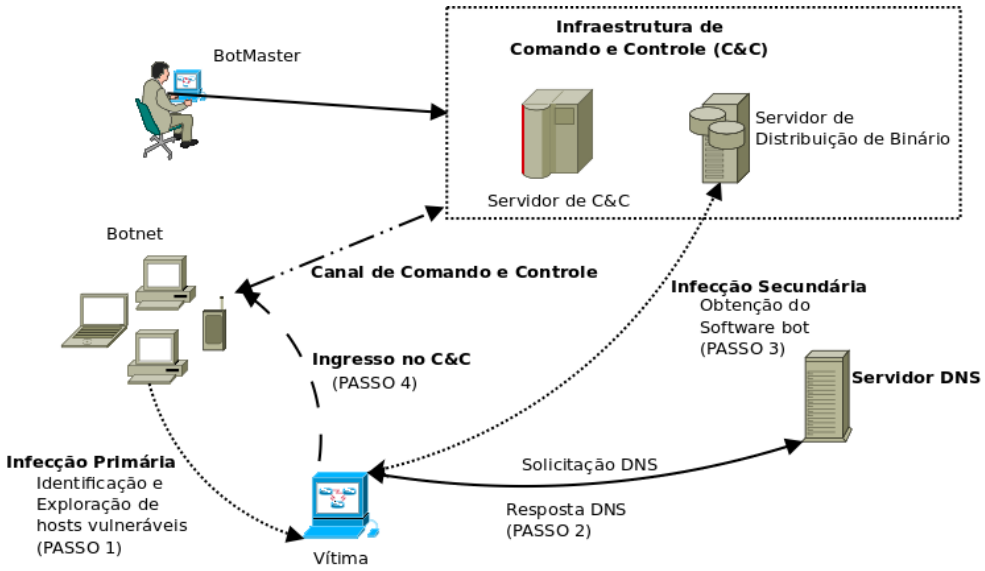

Figura 3.2. Ciclo de vida da *botnet*.

Para alguns trabalhos [Bazrafshan et al. 2013, Li et al. 2009a], o binário malicioso ou *backdoor* já faz parte do *malware* que infectou o dispositivo. Por outro lado, 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 a presença do binário malicioso durante o estágio inicial de infecção, um dispositivo zumbi somente se torna útil à *botnet* a partir do momento em que o *botmaster* sabe da sua existência.

Como mostrado na Figura 3.2, a infecção do dispositivo pode ser dividida em duas etapas: primária e secundária. Na infecção primária, o host vulnerável é infectado por um

malware (vírus, *worms*, *trojans*, entre outros). Em seguida, na infecção secundária, o host infectado inicia o download do código malicioso para ingressar no C&C. A vantagem dessa estratégia é que a *botnet* pode suportar versões específicas do binário malicioso para arquiteturas de computadores distintas. Portanto, computadores e outros dispositivos com poderes limitados podem ingressar no C&C.

Para encontrar o canal de comando e controle e unir-se aos demais membros da *botnet*, o *bot* realiza um procedimento chamado *rallying*. O termo *rallying* refere-se ao momento em que um *bot* está se autenticando no servidor de comando e controle [Schiller and 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 se encontra codificado no próprio código que o infectou [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 retirada de funcionamento [Egele et al. 2008].

Na abordagem dinâmica, o *bot* consulta servidores DNS (comprometidos ou não) para encontrar o endereço de nome de domínio que responde pelo C&C. Tal estratégia permite ao atacante realocar a sua rede de maneira rápida, caso servidores sejam sequestrados. Se a conexão falhar, o *bot* envia uma consulta DNS para receber o novo nome de domínio do servidor C&C. Existem alguns sites que fornecem esse serviço gratuitamente, como `dyndns.com`, onde é possível criar seu próprio domínio `yourname.dydns.com` e atribuir um IP dinâmico para este nome. Botnets recentes, por exemplo `Torpig` [Stone-Gross et al. 2009], usam domínios de fluxo rápido (*Fast Flux Domains*), onde cada *bot* independentemente usa um algoritmo de geração de nomes de domínios (DGA - *Domain Generatin Algorithm*) para computar uma lista de nomes de domínios. Quando nomes de domínios são usados, é necessário usar o sistema DNS para encontrar os endereços IP da máquinas a serem contactadas.

O sucesso de uma *botnet* está diretamente relacionado ao tempo que o *botmaster* mantém a rede em funcionamento. Por isso, diferentes arquiteturas tem sido utilizadas para tornar o canal de Comando e Controle mais resiliente.

3.2.5. Arquiteturas das Botnets

3.2.5.1. Botnets baseadas em Arquiteturas Centralizadas

O funcionamento das *botnets* baseadas em arquiteturas centralizadas é semelhante ao clássico modelo cliente-servidor [Rodríguez-Gómez et al. 2013]. Neste modelo, uma infraestrutura de C&C centralizada é utilizada para estabelecer uma conexão entre o *botmaster* e os clientes da *botnet*, como ilustrado na Figura 3.3. Diferentes protocolos podem ser empregados para estabelecer a comunicação entre os *bot* e o canal de controle, como `IRC` [Schiller and Binkley 2007], `HTTP` [Perdisci et al. 2010] e `DNS` [Dietrich et al. 2011]. No caso do protocolo `IRC` , os *bots* se conectam diretamente no canal de comando e controle, permitindo ao *botmaster* avaliar os resultados de comandos enviados em um menor tempo. Em contra partida, *bots* que utilizam o tráfego `HTTP` para se comunicar com o C&C, devem solicitar em intervalos de tempo novas instruções de ataques. Tal modo de operação é conhecido como `PULL` [Binsalleeh et al. 2010].

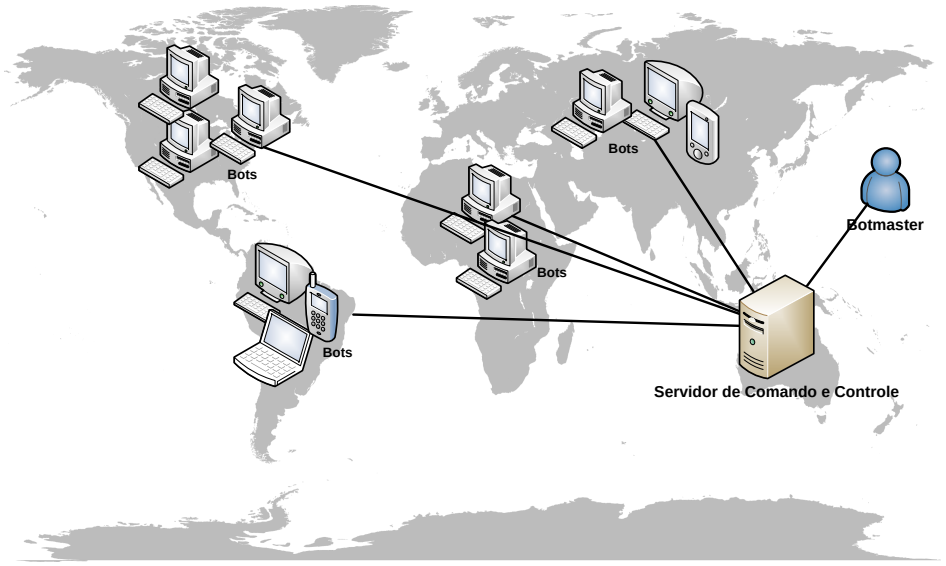

Figura 3.3. Exemplo de uma arquitetura centralizada de *botnet*.

Em geral, *botnets* baseadas em arquitetura centralizada oferecem menor latência de comunicação das mensagens trocadas entre o *botmaster* e os *bots* [Tyagi and G.Aghila 2011]. Por outro lado, o principal problema é que o servidor de C&C por si próprio é um ponto central de falhas, tornando-o não resiliente contra intervenções legais que visam interromper o seu funcionamento como sequestro e redirecionamento de domínio DNS [Stone-Gross et al. 2009] ou ataques de sessão `HTTP` ou `HTTPS` [Callegati et al. 2009].

3.2.5.2. Botnets baseadas em Arquiteturas Descentralizadas

Para oferecer robustez à infraestrutura da *botnet*, *botmasters* costumam empregar arquiteturas descentralizadas ou distribuídas [Tyagi and G.Aghila 2011]. Nesse tipo de arquitetura, protocolos `P2P` são frequentemente utilizados para fornecer resiliência, flexibilidade e escalabilidade a rede [Rodríguez-Gómez et al. 2013, Grizzard et al. 2007].

Nesse tipo de arquitetura, as informações sobre os membros ou da própria *botnet* estão distribuídas ao longo de toda rede maliciosa. Assim, *botnets* baseadas em arquiteturas descentralizadas são mais difíceis de serem desarticuladas, pois mesmo a descoberta de muitos *bots* não necessariamente significa a perda da rede da *botnet* inteira, visto que não existe um um servidor central de C&C [Rodríguez-Gómez et al. 2013]. A Figura 3.4 ilustra tal arquitetura de rede.

A união de um *bot* aos demais membros da *botnet* pode ser realizada através de consulta a uma lista estática de endereços, geralmente codificada no próprio software malicioso [Grizzard et al. 2007] ou por meio da varredura de endereços IP na Internet [Dagon et al. 2007]. Após a conexão com o C&C, um *bot* baseado em arquiteturas `P2P` busca, em servidores remotos, aplicações e instruções para atacar outras redes. Grizzard et al. (2007) apontam que, durante a fase de infecção do *bot* , se o conteúdo das informações foram cifradas usando criptografia de chave pública, a descoberta de outros componentes da infraestrutura é praticamente impossível. Além disso, a flexibilidade da

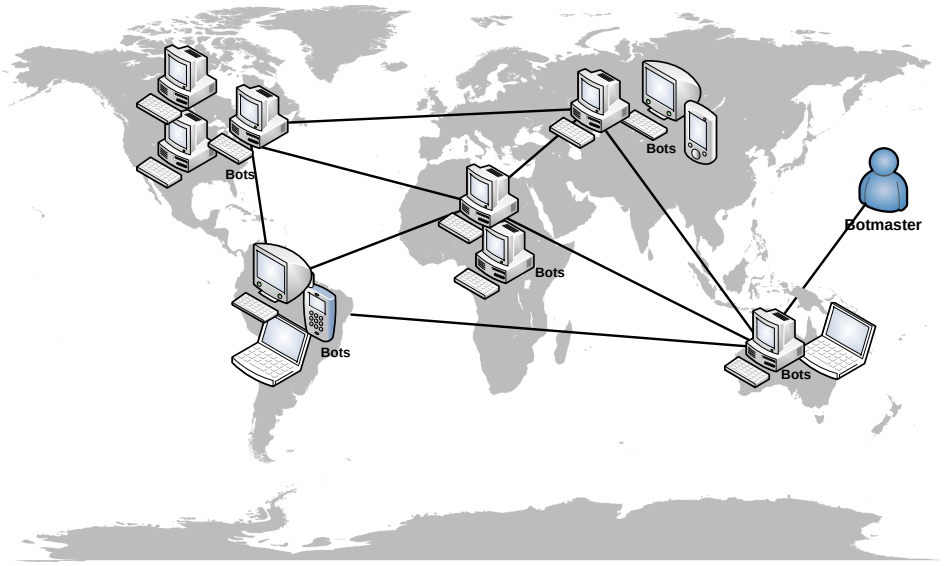

Figura 3.4. Exemplo de uma arquitetura descentralizada de *botnet*.

rede P2P dificulta a identificação dos membros e do canal de comando e controle. Em geral, as *botnets* descentralizadas operam em conjunto com outras redes P2P existentes como BitTorrent e Skype [Nappa et al. 2010], tornando difícil a distinção entre um host legítimo e um host malicioso [Yen and Reiter 2010].

As *botnets* baseadas em arquiteturas P2P podem operar tanto em modo estruturado quanto não-estruturado [Dagon et al. 2007]. No modo estruturado, existe um tipo de controle que pode ser utilizado para distribuir informações sobre a rede e armazenar arquivos. Um exemplo dessa arquitetura é o protocolo CHORD [Stoica et al. 2001]. Modelos não estruturados buscam pela descentralização completa da rede, onde cada nó participante opera tanto como cliente e servidor. A rede GNUTELA é um exemplo desse modelo [Chawathe et al. 2003].

Outra abordagem para manter o funcionamento da *botnet* é a organização dos nós em camadas e supernós [Chawathe et al. 2003]. Nessa estratégia, os clientes (*bots*) entram em contato com supernós, os quais fazem o roteamento das mensagens entre si até o destino final [Nappa et al. 2010]. A principal vantagem nessa arquitetura é que, caso um supernó seja retirado do ar, a *botnet* continua em operação devido outros supernós ainda receberem instruções do *botmaster*.

Para mitigar a proliferação de *botnets* baseadas em P2P, diversos trabalhos têm investigado técnicas para sobrescrever o índice da tabela de espalhamento distribuída com valores falsos [Wang et al. 2009]. Outra abordagem utilizada é infiltrar na *botnet* um grande número de nós falsos, visando interromper a comunicação entre os *bots* a partir da inserção do seu próprio endereço na lista de pares a servem comunicados [Holz et al. 2008b, Davis et al. 2008]. A análise do fluxo de rede também tem sido utilizada para identificar tráfego maliciosos em redes P2P [Rodríguez-Gómez et al. 2013, Peng et al. 2007].

3.2.5.3. Botnets baseadas em Arquiteturas Híbridas

Uma forma de manter o equilíbrio entre a simplicidade do C&C centralizado e a escalabilidade de uma *botnet* descentralizada é unir ambos os conceitos em uma arquitetura híbrida. A *botnet* `Waledac` [Calvet et al. 2010] é um típico exemplo que explora essa arquitetura. A comunicação dos *bots* é realizada através do protocolo `P2P` e alguns membros das *botnets* funcionam como servidores de encaminhamento (*relay*), os quais escondem o verdadeiro endereço do canal de comando e controle [Tenebro 2008]. Outros exemplos de *botnets* híbridas são `Alureon/TDL4` e `Zeus-P2P` [Rodionov and Matrosov 2011]

Além dos protocolos mais usados como `IRC` e `HTTP`, a pluralização do acesso aos dispositivos móveis permite que novas topologias de *botnets* híbridas sejam propostas e identificadas. Por exemplo, as mensagens de texto de celular (mensagens `SMS`) podem ser utilizadas para operar o canal de comando e controle, enquanto o protocolo `P2P` pode permitir que os nós sejam controlados de maneira descentralizada [Mulliner and Seifert 2010]. Aplicações maliciosas para dispositivos móveis já são observadas desde a década passada, incluindo aplicações para envio de spam, controle remoto e disseminação de vírus [Dunham 2009].

3.2.6. O que uma Botnet Faz?

Independentemente da arquitetura utilizada, uma *botnet* é projetada para executar uma grande variedade de ataques. Esta Seção descreve algumas das principais atividades executadas através de *botnets*.

Negação de serviço

Ataques de negação de serviço procuram impedir que os usuários legítimos possam se utilizar de algum serviço ou recurso na rede, tornando-os indisponíveis [Kumar and Selvakumar 2011]. Sendo atualmente executados de forma distribuída (DDoS - *Distributed Denial of Service*), este tipo de ataque necessita de uma estrutura que pode ser provida inteiramente pela arquitetura padrão de uma *botnet*, onde os *bots* são utilizados para sobrecarregar o alvo de acordo com as instruções do responsável pela coordenação geral do ataque.

Para intensificar o volume de tráfego durante ataques de DDOS, atacantes podem manipular protocolos que não armazenam o estado da conexão como `DNS` e `SNMP`. Recentemente, o procolo `SNMP` foi demonstrado como vetor de ataques de reflexão de resposta [BITAG 2012], onde endereços IP de origem são forjados durante as consultas para que as respostas sejam redirecionadas para o alvo do ataque.

O tráfego DNS também é utilizado para amplificar o tamanho das respostas [Guo et al. 2006]. Nessa abordagem, o atacante formula solicitações para vários servidores de nomes recursivos (`rDNS`) na Internet, cujas respostas ultrapassam o limite 1500 bytes da unidade de transmissão máxima (`MTU`) da rede. Portanto, ao receber a quantidade de pacotes, o servidor aumenta a carga de processamento para atender todas as solicitações recebidas. Um exemplo deste tipo de ataque é apresentado na Figura 3.5.

Peng et al. [Peng et al. 2007] mostram que uma consulta DNS com 60 bytes

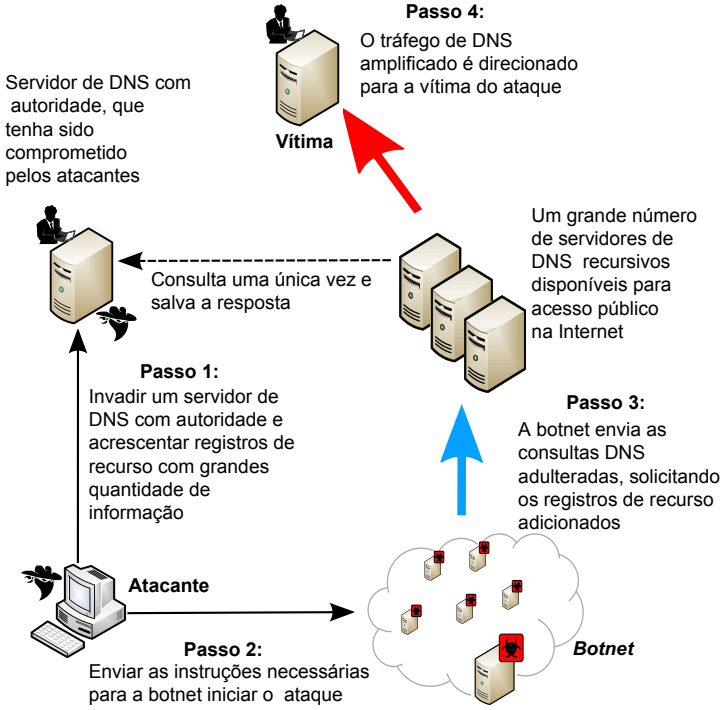

Figura 3.5. Exemplo de ataque de negação de serviço do tipo Amplificação de DNS.

de tamanho pode ser construída de tal forma a gerar uma resposta que possuirá mais de 4.380 bytes de tamanho, o que corresponde a um fator de ampliação superior a 73 vezes o tamanho da mensagem original. Para que este nível de amplificação seja alcançado, é necessário que o servidor de nomes, com a autoridade para responder a consulta, tenha sido previamente comprometido. Desta forma, o atacante pode manipular vários registros de recursos (RR) adicionais, os quais serão solicitados pelos membros da *botnet*.

Em contra partida, mesmo que os atacantes não tenham acesso ao servidor de DNS comprometido, as consultas DNS da *botnet* podem ser construídas para utilizar endereços públicos autênticos que já possuem tais registros.

Ataques de reconhecimento de rede

Para identificar um IP válido, o atacante pode utilizar ferramentas que percorrem segmentos de endereçamento IP, como `nmap`[1]. No entanto, essa abordagem gera muito ruído (evidência) e pode ser utilizada para identificar o atacante.

Outra solução para encontrar endereços válidos é utilização do tráfego DNS. Nesse tipo de ataque, o tráfego DNS indica possíveis nós ativos e configurados na rede. Essa abordagem utiliza o registro de recurso do tipo `PTR`, o qual fornece o nome de um domínio a partir de um endereço IP. O registro reverso `PTR` é útil, pois cada IP acessível na Internet deve possuir um nome reverso [Barr 1996].

Além disso, programas de código malicioso também utilizam técnicas de reconhecimento de rede para encontrar possíveis nós comprometidos. Essa estratégia tem como

[1]http://nmap.org/

objetivo comprometer outros computadores localizados na rede. Dependendo da natureza do programa malicioso, o reconhecimento pode utilizar portas de serviço bem conhecidas ou comprometer outros computadores com base na confiança entre eles.

Propagação de *spam* em massa

Botnets também são usadas com um meio eficiente para a propagação de mensagens *spam* em massa [John et al. 2009]. Antes de enviar uma mensagem não solicitada, o *bot* precisa percorrer endereços para encontrar servidores de e-mail abertos na Internet (*relay servers*). Para evadir de possíveis sistemas de detecção de intrusos (IDS) na rede, os *bots* solicitam essas informações diretamente aos servidores de nome raiz. A Figura 3.6 ilustra esse processo.

Para identificar ataques de *spam* em massa, uma alternativa é monitorar a frequência de utilização dos registros `A`, `PTR` e `MX` [Barbosa and Souto 2009]. Esses tipos de registros indicam fortes evidências de comportamento anômalo na rede. Por exemplo, uma variação do *worm Sobig* é observada a partir da análise dos registros `A` e `MX` [Levy 2003].

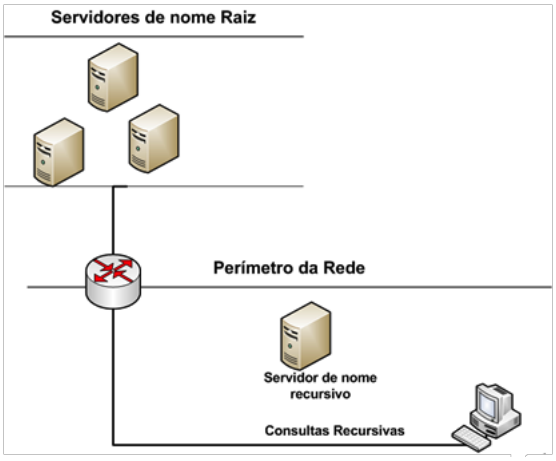

Figura 3.6. Cliente infectado busca diretamente os servidores de raiz para não ser detectador por sistemas de intruso da rede.

Botnets como um modelo de negócios

Para Li et al. [Li et al. 2009b], antigamente os desenvolvedores de *malware* eram motivados por sentimentos de auto-realização, por diversão ou ainda pelo desejo de provar suas habilidades. Este comportamento mudou para motivações de ordem financeira, onde o modelo de negócio empregado envolve a criação, exploração e manutenção de *botnets*, que tem seus recursos postos à venda para aqueles interessados nas atividades ilegais executadas através destas redes. Como estas redes tendem a crescer de forma exponencial, isto se tornou um negócio da ordem de bilhões de dólares.

De acordo com Feily et al. [Feily et al. 2009], o controle de *botnets* se transformou num negócio ilícito muito atrativo, visto que tais redes são projetadas para proteger a identidade de seus criadores. Este anonimato é obtido por elementos como: *i*) um canal de comunicação de múltiplas camadas, o que dificulta a sua rastreabilidade; *ii*) pela distribuição física dos *bots* por diversos países, o que agrega diferenças de idiomas, zonas de tempo e leis; e *iii*) pelas próprias distâncias geográficas associadas aos elementos que

constituem a *botnet*, dificultando ainda mais o combate a este tipo de ameaça cibernética.

Um relatório técnico do *Communication Systems Group* (CSG), sobre os aspectos financeiros de diversas *botnets* [Studer 2011], afirma que existem duas formas de se ganhar dinheiro neste ramo de negócios escusos: *i*) venda de serviços, onde os *botmasters* são contratados para atacar alvos definidos pelos seus clientes; ou ainda *ii*) através do aluguel da *botnet*, onde os criadores originais cedem temporariamente o controle dos computadores infectados a terceiros. Os dados financeiros apresentados neste relatório são preocupantes:

- Apenas no ano de 2008, depois de executarem 190 mil ataques para seus clientes, os proprietários das *botnets* ganharam aproximadamente 20 milhões de dólares, o que torna este negócio muito rentável.

- Por apenas 67 dólares por hora, é possível alugar uma *botnet* com mais de mil computadores comprometidos à sua disposição, o que deixa o processo de aluguel de *botnets* atrativo.

- Uma análise dos dados de quatro empresas, para exemplificar o impacto destes ataques para os seus alvos, mostrou que o custo hora de cada ataque sofrido variou de 190 mil dólares até 19 milhões de dólares, com uma perda total média variando de 380 mil dólares até 114 milhões de dólares.

Levando estas características em consideração, Li et al. [Li et al. 2009b] propõem combater o fenômeno das *botnets* através de uma abordagem econômica, onde máquinas virtuais seriam propositadamente inseridas na rede para comprometer o desempenho da *botnet* e a efetividade de seu ataque, tornando seu uso inviável economicamente.

3.2.7. Abordagens usadas para detecção de *botnets*

Em função das características fundamentais das redes maliciosas, existem duas abordagens normalmente usadas para o combate às *botnets*: *i*) utilização de *honeypots*, e *ii*) monitoramento e análise do tráfego de rede [Feily et al. 2009].

Os mecanismos de *honeypots* permitem simular comportamentos de sistemas operacionais ou aplicações com problemas de vulnerabilidades. Nesse tipo de abordagem, o *honeypot* pode interagir (responder) aos ataques de maneira passiva ou ativa [Alata et al. 2007]. *Honeypots* passivos, ou de baixa interatividade, permitem detectar tentativas de varreduras de portas e servidores de redes falsos, enquanto os *honeypots* ativos, ou de alta interatividade, fornecem acesso ao atacante em um ambiente controlado. Tal ambiente é frequentemente monitorado para extração de registros de acesso e características de comportamentos maliciosos.

Vale ressaltar que *honeypots* são ferramentas que possibilitam a coleta de amostras de *malwares*, que serão analisadas com o objetivo de identificar as características e tecnologias associadas às redes maliciosas. Por outro lado, o monitoramento e análise passiva do tráfego de rede tem a responsabilidade de prevenir ou identificar as contaminações nas redes monitoradas.

O monitoramento e análise do tráfego podem ser desenvolvidos através do emprego de duas abordagens distintas: *a*) baseada em assinaturas e *b*) análise de fluxo de dados para detecção de anomalias [Feily et al. 2009].

As abordagens baseadas em assinaturas englobam sistemas de detecção de intrusão que monitoram continuamente o tráfego da rede, comparando elementos suspeitos com uma base de assinaturas ou com um conjunto de regras. Esta comparação permite identificar a presença de algum elemento malicioso previamente mapeado [Peng et al. 2007]. Por esta razão, esta técnica é eficiente e, em geral, resulta em baixas taxas de falso positivos. Entretanto, este modelo de identificação não é capaz de lidar com *malware* cuja descrição não tenha sido inserida previamente no sistema, o que deixa esta solução dependente de bases de dados atualizadas.

A detecção de *botnets* através da análise de fluxo de dados de rede permite a identificação de informações relevantes a respeito da estrutura de controle da *botnet*, tais como características e tecnologias empregadas para sustentar seu funcionamento. Esta abordagem é vital em situações onde o processo de engenharia reversa, que é a base para a construção de assinaturas de *malware*, não é suficiente para interromper a proliferação de ataques na rede [Stone-Gross et al. 2009]. O objetivo principal é inspecionar o fluxo para identificar comportamentos suspeitos como alta latência, grande volume de dados sendo transportados, presença de tráfego associado à portas de comunicação incomuns ou, ainda, comportamentos incomuns do sistema. Essas e outras características são utilizadas como indicadores da presença de componentes contaminados dentro da rede. Por exemplo, o monitoramento do tráfego `DNS` permite identificar o momento em que a máquina contaminada tenta se comunicar com o *botmaster*.

Para detecção de *botnets* através da análise de rede, diferentes abordagens podem ser empregadas. Agrupamento de padrões de comportamento semelhantes (conhecidos) ou comportamentos suspeitos (não conhecidos) em *botnets* [Gu et al. 2008a, Zeidanloo and Manaf 2010]. Técnicas da Teoria da Informação [Husna et al. 2008, Kang and Zhang 2009] e classificação automática através de aprendizagem de máquina [Zhao et al. 2013, Lin et al. 2009].

A Seção 3 apresenta alguns trabalhos que empregam diferentes abordagens e analisam diferentes aspectos do tráfego de rede para identificação de *botnets*.

3.3. Estado da Arte

Esta Seção descreve algumas das principais abordagens utilizadas para identificação e caracterização de redes *botnets*. Para um melhor entendimento, os trabalhos apresentados são categorizados de acordo com o protocolo utilizado pela *botnet*.

3.3.1. Botnets baseadas no Protocolo IRC

O protocolo IRC foi inicialmente projetado para estabelecer a comunicação entre usuários distribuídos geograficamente em um único servidor [Oikarinen and Reed 1993]. Devido ao crescimento da rede IRC, operadores desenvolveram ferramentas (*scripts*) para auxiliar na administração de usuários e canais de bate-papo. Essa possibilidade de comunicação em tempo real tem sido bastante explorada por atacantes para controlar máquinas infec-

tadas [Schiller and Binkley 2007].

No ciclo de vida de uma *botnet* IRC, cada *bot* tenta encontrar um servidor IRC através de uma consulta DNS. Uma vez que a comunicação entre *bots* e servidor IRC está estabelecida, o *bot* envia uma senha, através da mensagem `PASS`, para iniciar o processo de autenticação. Tal mecanismo pode ser mútuo: o *bot* se autentica para o servidor e o *botmaster* se identifica para o *bot* [Lu and Ghorbani 2008]. Para iniciar um ataque de negação de serviço, o *botmaster* envia um comando para a sala (por exemplo, `!ddos start IP`), como ilustrado na Figura 3.7.

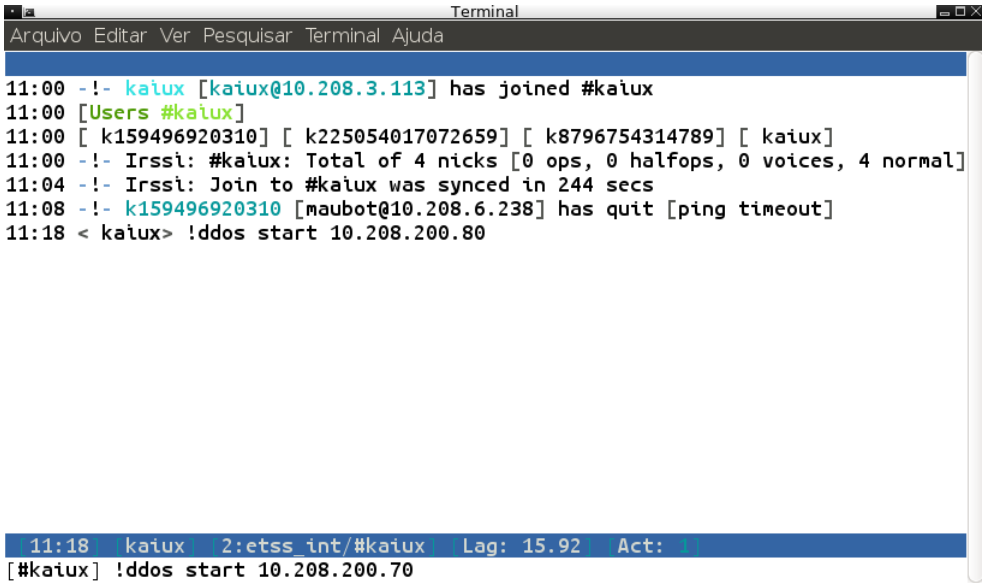

Figura 3.7. Botmaster disparando ataque do tipo SYN

A seguir são fornecidos alguns exemplos de *botnets* baseadas no protocolo IRC. Canavan apresentou uma análise da evolução histórica das *botnets* e do processo de comunicação entre o operador da *botnet* e os *bots* [Canavan 2005]. Essa análise demonstrou que o código fonte da *botnet* `GT-Bot` era utilizado para criar versões derivadas de outras *botnets* como `SpyBot`, `RBot` e `Agobot`. Em função disto, os *bots* criados possuíam semelhanças estruturais de código, o que permitiu o desenvolvimento de soluções de antivírus, baseadas em assinaturas e em padrões de comportamento, destinadas a identificar e interromper a proliferação de *botnets*. Entretanto, algumas *botnets* conseguiam enganar tais sistemas através de técnicas de polimorfismo de código como, por exemplo, a `PolyBot` [Hachem et al. 2011].

Mazzariello apresentou uma abordagem capaz de diferenciar padrões normais (emitidos por usuários legítimos) e maliciosos (emitidos por máquinas infectadas) em canais IRC [Mazzariello 2008]. O autor parte do princípio que as sequências de caracteres e estruturas gramaticais utilizadas por humanos mudam com mais frequência e não estão limitadas a um conjunto restrito de palavras ou dicionários, diferentemente de máquinas infectadas. O procedimento de detecção é baseado em um conjunto de características obtidas a partir de um canal IRC, como número de usuários, quantidade de palavras numa sentença, frequência de palavras, apelidos (*nicknames*) diferentes, quantidade de respostas semelhantes a partir de uma pergunta e número de comandos emitidos como `ping` e

join. Tais características são extraídas a partir de uma base de dados contendo amostras benignas e maliciosas, classificadas através de algoritmos de aprendizagem de máquina como SVM e J48.

Os autores em [Livadas et al. 2006] utilizaram as características do tráfego IRC como tamanho da carga útil, endereços IP de origem e destino, *flags* do cabeçalho TCP e total de bytes para detecção de anomalias. Tais características são extraídas de bases de dados rotuladas como legítimas e maliciosas. Livadas et al. compararam o desempenho de classificação usando Redes Bayesianas, Naive Bayes e árvore de decisão (J48). Os resultados mostraram que o classificador Naive Bayes detecta 97.51% de pacotes com tráfego malicioso e os algoritmos de árvore de decisão (J48) e redes bayesianas identificam 92.11% e 90% de tráfego de *botnet*, respectivamente. No entanto, para classificar comportamento malicioso na rede, a proposta utiliza características do tráfego quando o serviço de IRC é utilizado na sua configuração padrão e comunicação entre *bots* em texto claro.

Já em [Strayer et al. 2008], os autores apresentaram uma arquitetura para identificar *botnets* através da análise de características do fluxo de rede, como largura de banda, temporização dos pacotes e tempo de comunicação, em busca de indícios da presença de mensagens de C&C. A metodologia proposta está dividida em quatro etapas: (a) filtragem, (b) classificação, (c) correlação e (d) análise topológica. Na primeira etapa, os pacotes de rede são filtrados por listas de reputação, tipo de protocolo (TCP), flags do cabeçalho (SYN−RST), remoção de pacotes com mais de 300 bytes e fluxos de curta duração (tamanho superior a um pacote e com duração máxima de apenas 60 segundos). Isto remove pacotes que não são considerados suspeitos, como mensagens de IRC normais ou ataques de varredura, por exemplo.

Na etapa seguinte, os fluxos remanescentes são classificados através do algoritmo de aprendizagem de máquina Naïve Bayes, para determinar se existe alguma evidência de comunicação entre clientes e servidores IRC. A partir da identificação de comunicação IRC, os fluxos são correlacionados para rastrear hosts que possuam o mesmo tipo de comportamento, o que permite identificar outros elementos participando da mesma *botnet*. O agrupamento de fluxo é feito através de um algoritmo que calcula a distância entre eles. Na última etapa da metodologia, pares de fluxo que possuem maior afinidade são investigados através da análise topológica, a qual permite identificar membros da *botnet* e servidores que controlam o canal de C&C. Os experimentos demonstraram que esta metodologia é capaz de identificar a presença de *botnets*, mesmo em tráfegos contendo mais de um 1.3 milhões de fluxos IRC.

Ma et al. [Ma et al. 2010] apresentaram um algoritmo de detecção de botnets através da análise de sequência dos tamanhos dos pacotes IRC. Este trabalho parte do princípio que os padrões de comportamento apresentado por humanos e *bots*, quando estão interagindo com servidores de IRC, possuem características distintas. Enquanto o primeiro apresenta um padrão estocástico, o segundo demonstra possuir ciclos de periodicidade. Para identificar estas diferenças, os autores propuseram a construção de uma estrutura de dados baseada na sequência de conteúdo da conversa (CCS - *conversation content sequence*). Tal estrutura registra o tamanho dos primeiros 120 pacotes transmitidos entre o cliente e o servidor IRC. A partir daí, a metodologia de identificação é dividida

em 4 etapas.

Inicialmente, é feito um cálculo da média dos tamanhos de CCS, o qual é útil para identificar sequências de conteúdo de conversa produzidas seres humanos. Na segunda etapa, o conteúdo de CSS é convertido em uma *string S*, a qual é inserida em uma árvore de *substring* utilizando o algoritmo *Ukkonen*. Na etapa seguinte, esta árvore é usada para encontrar *substrings* que são mais frequentes e não se sobrepõem, sendo assim identificadas como instruções de ataques. Na quarta e última etapa, é feito um cálculo para determinar se *S* é ou não uma *string* de `botnet`, a partir da média dos tamanhos dos pacotes e do seu nível de periodicidade. Os autores validaram a proposta a partir de base de dados do projeto *HoneyNet* de fevereiro a abril de 2009[2] e tráfego benigno do provedor de IRC *FreeNode*[3]. Os experimentos demonstraram a eficiência deste método na identificação de *botnets*, mas os autores deixam claro que a inclusão de mensagens randômicas na comunicação de C&C pode dificultar a detecção por este método.

Os autores em [Carpine et al. 2013] apresentaram uma abordagem de detecção de *botnets* baseadas no monitoramento e construção de modelos comportamentais, para o tráfego IRC na rede e para as salas de bate-papo específicas. Este modelo foi construído com o objetivo de atingir três metas básicas: (a) ser capaz de efetuar detecção em tempo real, com um monitoramento on-line da rede; (b) possuir desempenho satisfatório, sem exigências elevadas de poder de processamento; e (c) fazer uso de modelos de classificação baseados em técnicas de aprendizagem de máquina.

Para atingir todos estes objetivos, os autores construíram um sistema que aprende de maneira gradativa o conceito atual, sem perder o que já fora aprendido. Este classificador foi baseado nos algoritmos SOINN [Shen and Hasegawa 2010] e KNN, com o segundo sendo utilizado para fornecer dados de entrada para que o primeiro aprenda novos conceitos. Esta metodologia utiliza ainda uma função de maximização, que busca encontrar o mínimo de instâncias e que ofereça o máximo de precisão para a base de dados utilizada. A detecção acontece em tempo real e possui o desempenho desejado, pois o classificador é capaz de processar 10.000 amostras por segundo, em um computador CPU *Intel Celeron* 530 com 1.5GB de memória RAM.

A partir das características identificadas nos trabalhos apresentados nesta Seção, é possível observar que a detecção de *botnets* baseadas no tráfego IRC pode ser dividida em duas etapas: identificação de tráfego IRC na rede e classificação de comportamento malicioso. A primeira tem como objetivo identificar e filtrar apenas o tráfego IRC dos demais protocolos na rede, pois algumas *botnets* não utilizam a porta padrão do serviço de IRC [Strayer et al. 2008]. Na etapa seguinte, as características da carga útil do IRC são utilizadas para fazer a distinção entre comportamentos suspeitos e benignos.

A Tabela 3.1 sumariza as principais características utilizadas em ambas às etapas.

Além das características citadas, alguns trabalhos incluem o uso de lista negra e expressões regulares para identificar apelidos de *bots*.

[2]http://www.edu.cn/HomePage/english/cernet/index.shtml

[3]http://www.freenode.org/

Tabela 3.1. Características utilizadas para classificação de botnets baseadas no tráfego IRC.

Características de Rede	Tupla Básica	IP de origem e destino
		Porta de origem e destino
		Flags do cabeçalho TCP (syn, syn-rst, ack)
		Tempo de chegada do pacote.
	Estatística do Pacote ou Fluxo	Tamanho
		Média do tamanho do pacote
		Round-Trip-Time (RTT)
		Total de pacotes e de bytes
		Total de bytes
		Total de pacotes enviados
		Média de bytes e de bits por pacote no fluxo
		Média de pacotes enviados por segundo
		Porcentagem de pacotes enviados
		Variância do RTT
		Variância de bytes por pacote no fluxo
		Intervalo de tempo de chegada de pacotes
Classificação do Comportamento	Comandos IRC	nick, user, privmsg, join, ping, pong, who, kick
	Estatísticas do Canal IRC	Total de usuários
		Média de palavras em uma sentença
		Distribuição de palavras do dicionário
		Número de respostas iguais
		Taxa de ingresso no canal
		Número de mudanças de apelidos
		Média e variância de mensagens privadas
		Média de vogais, consoantes e caracteres especiais em uma sentença e no tópico do canal
	Estatísticas do Apelido	Tamanho do nickname
		Quantidade de letras
		Quantidade de números
		Média de vogais, consoantes e caracteres especiais no apelido

3.3.2. Tráfego HTTP

Botnets baseadas no tráfego HTTP apresentam algumas vantagens em comparação com as que utilizam o tráfego IRC. Uma das mais óbvias é que o tráfego IRC não é considerado comum em redes corporativas. Portanto, a utilização desse protocolo é sempre sinalizada como suspeita ou é simplesmente bloqueada. Por outro lado, o bloqueio do tráfego HTTP é inviável, pois restringe o acesso e a navegação na Internet. Em função disto, uma *botnet* baseada no tráfego HTTP é capaz de passar por filtros tradicionais de pacotes. Tal característica é explorada por *botmasters* para hospedar mecanismos de C&C [Perdisci et al. 2010] ou para utilizar redes sociais visando propagar aplicações de código malicioso [Souza et al. 2013, Freitas et al. 2014]

Outras *botnets* utilizam aplicações baseadas no tráfego HTTP para buscar e armazenar informações privilegiadas e disseminar códigos maliciosos. Por exemplo, [Boshmaf et al. 2011] simularam o comportamento de usuários e se infiltraram na rede social *Facebook*. Os autores conseguiram coletar, durante 8 semanas, informações úteis na identificação de uma pessoa, como número do telefone, idade, sexo, endereço residencial e email. Outros exemplos demonstram a existência de *bots* publicando informações fraudulentas sobre eleição norte americana na rede social *Twitter* [Ratkiewicz et al] ou utilizando os tópicos de tendência (*trending topics*) para divulgar sites maliciosos e ataques

de *phishing* através de URLs encurtadas [Souza et al. 2013, Freitas et al. 2014].

As *botnets* baseadas no tráfego HTTP precisam frequentemente entrar em contato com o canal de comando e controle para obter instruções do *botmaster*, estratégia conhecida como `PULL` (solicitação) [Gu et al. 2008b]. Devido a isto, alguns trabalhos identificaram essas rede a partir do grau de repetição de acesso [Lee et al. 2008]. No entanto, para evadir tal mecanismo de detecção, os *botmasters* adicionaram comportamentos aleatórios nas solicitações de comandos ao C&C, resultando no aumento da quantidade de falsos alarmes na rede [Eslahi et al. 2013]. Por isso, diversos trabalhos buscam identificar *botnets* baseadas no tráfego HTTP através da construção de modelos comportamentais para o tráfego da rede.

A combinação do tráfego HTTP com outros mecanismos para garantir maior resiliência da *botnet* é proposta em [Xiang et al. 2011]. A proposta tem como objetivo permitir que dispositivos de pequeno porte (*smarthones* e PDAs, por exemplo) acessem o C&C através de uma arquitetura híbrida. Nesta arquitetura, o tráfego HTTP é o protocolo de comunicação e as instruções de ataques são obtidas através de URLs criadas dinamicamente por algoritmos.

Por exemplo, se o canal de C&C é uma URL de um usuário em uma rede social, as máquinas infectadas gerariam nomes de usuários dinamicamente (UGA - *user generation algorithm*) para obterem as instruções de ataque. Tal estratégia é semelhante as redes de domínios de fluxo rápido, baseadas em algoritmos DGA. Desta forma, a proposta de Xiang et al. também pode ser detectada a partir de consultas com erros (`404` - página não encontrada).

Perdisci et al. [Perdisci et al. 2010] apresentaram um sistema de agrupamento de *malware* em nível de rede. Os resultados mostraram que é possível extrair automaticamente assinaturas de rede a partir de máquinas infectadas utilizando o tráfego HTTP como meio de disseminação. Para extração das características da rede, é usado um sistema de detecção de intrusos no roteador de borda para monitorar solicitações HTTP de saída. Este processo de detecção monitora a quantidade consultas, o tamanho e semelhanças estruturais entre URLs, a duração e o tipo de solicitação enviada. Inicialmente, os `malwares` são agrupados com base no comportamento para encontrar semelhanças estruturais entre as sequências de solicitações HTTP.

Este processo é dividido em duas etapas. Primeiro, são agrupados os *malwares* com base em elementos estatísticos (e.g.: número de solicitações HTTP e tamanho da URL). Na segunda etapa, após este agrupamento inicial, cada grupo é dividido em subgrupos classificados por características estatísticas de tráfego e com diferentes estruturas do HTTP. Para validação da metodologia proposta, os autores compararam os seus resultados com aqueles obtidos por três produtos de antivírus comerciais: *McAfee*[4], *Avira*[5] e *TrendMicro*[6]. Ao final deste processo de avaliação, nenhuma das soluções de antivírus foi capaz de detectar os *malwares* identificados pela metodologia proposta. No entanto, como a abordagem proposta monitora apenas o tráfego HTTP, uma *botnet* baseada em HTTPS seria capaz de burlar este processo de identificação.

[4]http://www.mcafee.com

[5]http://www.avira.com

[6]http://www.trendmicro.com

Haddadi et al. [Haddadi et al. 2014] detectaram *botnets* baseadas no tráfego HTTP através da análise de fluxo de rede ao invés da carga útil do pacote. Esta abordagem permite identificar comportamento malicioso mesmo que o tráfego entre o cliente e o canal de comando e controle esteja encriptado. Cada componente do fluxo da rede é representado através de uma tupla contendo cinco elementos extraídos a partir do cabeçalho do pacote: endereços IP de origem e destino, portas de origem e destino e protocolo de comunicação. Além disso, os autores extraem 21 características (como duração do fluxo, tamanho do fluxo e o tipo de serviço) que são utilizadas por dois classificadores: Naive Bayes e C4.5 (árvore de decisão). Para validação, Haddadi et al. utilizam uma base benigna[7] e outra maliciosa[8] e comparam os desempenho dos classificadores. Os resultados obtidos atestaram que, apesar do algoritmo C4.5 exigir mais tempo computacional para classificação, os resultados foram superiores aos do Naive Bayes, atingindo uma taxa precisão de 88% identificação das *botnets* avaliadas.

Cai e Zou [Cai and Zou 2012] investigaram como as características do tráfego HTTP podem ser agrupadas para identificar *botnets*. Algumas das características estudadas incluem campos do cabeçalho HTTP como `user agent`, tipo de conteúdo e tamanho do conteúdo. Como resultado deste estudo, foi identificado que *bots* não criam as solicitações HTTP contendo as informações completas ou corretas. Além disso, para que a *botnet* seja escalável, o tamanho do conteúdo da página, que fornece as instruções do comando e controle, deve ser pequeno. A partir da análise do tipo de conteúdo foi demonstrado que os *bots* fazem download de binários maliciosos com extensões de arquivos falsos, como .mp3 e .mp4, com o objetivo de evadir filtros de rede.

De maneira semelhante aos trabalhos de Perdisci et al. e Haddadi et al., a proposta de Cai e Zou mostra que a frequência de consultas entre máquinas infectadas e o servidor de C&C pode ser utilizada na detecção de *bots* na rede. No entanto, como apontado por Haddadi et al., caso o *botmaster* utilize algum tipo de criptografia entre o *bot* e o canal de comando e controle, esta proposta não seria mais viável como ferramenta de identificação.

A Tabela 3.2 sumariza as características utilizadas do tráfego de rede e os métodos estatísticos para detecção de *botnets* baseadas no tráfego HTTP.

3.3.3. Tráfego DNS

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.

[7] http://www.alexa.com

[8] https://zeustracker.abuse.ch

Tabela 3.2. Características utilizadas para classificação de botnets baseadas no tráfego HTTP.

Características de Rede	Métodos HTTP GET, POST, HEAD
	Nome do USER Agent
	Porta de destinos (80, 8080, 800*, 443)
	Campos do cabeçalho HTTP como tamanho do conteúdo (content-length)
	Tipo do conteúdo (content-type)
Classificação do Comportamento	Tamanho médio da URL
	Total de requisições
	Total de GET, POST e HEAD
	Média de argumentos passados na URL
	Média de bytes enviados no POST
	Média do tamanho das respostas
	Similaridade entre os valores dos argumentos passados
	Frequência de solicitações GET/POST
	Total de pacotes e bytes
	Total de pacotes ebits por segundo
	Total de bytes por pacote

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, como o ataque Kaminsky [Musashi et al. 2011]. Na segunda categoria, atacantes utilizam o tráfego DNS para identificar possíveis alvos configurados em redes remotas, encontrar servidores de correio eletrônico abertos ou para encontrar o endereço do servidor do canal de comando e controle durante a fase de rallying.

Por esses motivos, o tráfego DNS tem sido investigado para identificar tanto anomalias de rede quanto para entender o ciclo de vida de *botnets*. Por exemplo, no trabalho de Choi e Lee [Choi and Lee 2012], *botnets* são detectadas a partir do agrupamento de atividades similares no tráfego DNS, tais como consultas repentinas, quantidade de consultas únicas, nome de domínio requisitado por múltiplos endereços IP de origem em um determinado intervalo de tempo e consultas para domínios dinâmicos DNS (`DDNS`). O processo de detecção utiliza similaridades entre consultas, listas negras para diferenciar entre domínios legítimos e maliciosos e aplicações de terceiros (máquinas de busca e reputação web, por exemplo).

Para encontrar o canal de C&C, os *bots* podem utilizar consultas DNS estáticas ou dinâmicas. Na consulta estática, o host comprometido utiliza o endereço IP do servidor de C&C codificado no próprio software *bot* [Jakobsson and Ramzan 2008]. No entanto, apesar da estratégia ser simples, tal abordagem apresenta diversas fraquezas [Tiirmaa-Klaar et al. 2013]. Por exemplo, técnicas de engenharia reversa poderiam revelar o endereço IP da rede permitindo que a *botnet* fosse retirada de funcionamento. Outro problema ocorre na migração de uma *botnet* para outro canal de comando e controle, pois os membros da *botnet* devem atualizar o software bot e tal comportamento em massa pode indicar atividades suspeitas na rede [Choi et al. 2009].

Uma alternativa para contornar os problemas das listas estáticas é o uso da resolução dinâmica de nomes de domínios, a qual pode ser dividia em duas estratégias. Na primeira, um nome de domínio é gerado automaticamente por um algoritmo, chamado DGA (*Do-*

main Generation Algorithm) [Yadav et al. 2012]. Tais nomes de domínios também são conhecidos como domínios de fluxo rápido ou *domain flux*. Para ilustrar como nomes de domínios são gerados, a Listagem 3.1 apresenta um algoritmo que gera um nome 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.br` e `odnslofgimonbruy.net`.

Listagem 3.1.]Exemplo de algorítimo que gera nome de domínio. [Autor desconhecido]

```
1  def generate_domain(year, month, day):
2      """Generates a domain by the current date"""
3      domain = ""
4      for i in range(16):
5          year = ((year ^ 8 * year) >> 11) ^ ((year & 0xFFFFFFF0) <<
               17)
6          month = ((month ^ 4 * month) >> 25) ^ 16 * (month & 0
               xFFFFFFF8)
7          day = ((day ^ (day << 13)) >> 19) ^ ((day & 0xFFFFFFFE) <<
               12)
8          domain += chr(((year ^ month ^ day) % 25) + 97)
9          print domain
10
11     return domain
12
13 print generate_domain(2014, 9, 25)
```

Na segunda estratégia para encontrar o endereço do C&C, o nome de domínio pode ser estático e as mensagens direcionadas ao servidor de comando são processadas por *bots* , que operam como intermediadores (*proxy*) entre a vítima e o verdadeiro servidor. Vale ressaltar que essa abordagem é frequentemente utilizada para esconder páginas de conteúdo malicioso ou ilícito [Holz et al. 2008a].

A Figura 3.8 ilustra o mecanismo conhecido como redes de serviço de fluxo rápido (FFSN - *fast-flux service network*). Antes de acessar à página maliciosamente forjada pelo *botmaster* , a vítima deve consultar o endereço do servidor que hospeda tal conteúdo. O objetivo das redes de fluxo rápido é manter o maior número de endereços IP associados a um nome de domínio. Por exemplo, uma vítima consulta o domínio `www.example.com` e obtém como resposta o endereço IP `192.168.0.8`. A cada consulta direcionada ao nome de domínio malicioso, novos endereços IPs, que fazem parte da *botnet* , são retornados. Estes endereços IPs são trocados com frequência, usando uma combinação de técnicas de *Round-Robin* e o tempo de vida (TTL) consideravelmente curto. Portanto, a vítima ao acessar o conteúdo malicioso, na verdade, está se comunicando com *bots* ao invés do servidor central (*mothership*).

Salusky e Danford [Salusky and Danford 2008] apresentaram um dos primeiros trabalhos dedicados à identificação e caracterização do tráfego de redes de fluxo rápido, incluindo as principais diferenças entre as redes *single-flux* e *double-flux*. Redes *single-flux* mudam apenas o registro de recurso (RR) do tipo `A`, enquanto as redes *double-flux* alteram os RRs `A` e `NS`, o qual contém o endereço IP do servidor de nomes que responde pelo domínio malicioso.

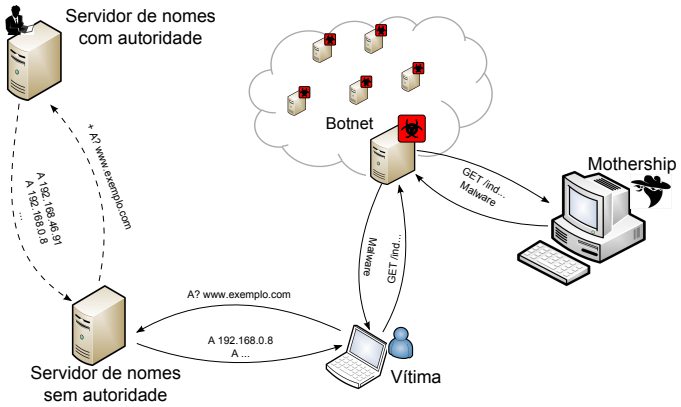

Figura 3.8. Exemplo de rede de fluxo rápido.

A proposta de Salusky e Danford para mitigar redirecionamentos de fluxo de rede é baseada no monitoramento do tempo de vida (`TTL`) das consultas DNS. Um domínio benigno possui um número estável de endereços IPs, enquanto um domínio de fluxo rápido emprega o `TTL` curto para que o nome de domínio seja atendido por um número maior de endereços IP distintos de máquinas infectadas.

No entanto, Holz et. al. [Holz et al. 2008a] 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. Os autores apresentaram uma heurística que utiliza um conjunto de recursos de rede para distinguir entre tráfego de rede de fluxo rápido, redes de distribuição de conteúdo (CDN) e redes que usam o tráfego DNS para distribuição de carga em diferentes servidores. Para isso, foi empregado o número de endereços IP dos registros de recurso `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 redes CDN e FFSN, Holtz et al. dividem a quantidade total de endereços IP encontrados no registro `A`, pela quantidade de endereços retornados em uma única consulta do tipo `A`. Caso o valor encontrado seja próximo a 1, o domínio é benigno, caso contrário o domínio apresenta comportamento de CDN ou FFSN. Para classificação de domínios de fluxo rápido, os autores compararam as características de base de dados de domínios benignos (e.g.: `alexa.com` e Projeto Dmoz[9]) com base de dados de domínios maliciosos, a qual foi criada a partir de `URLs` presentes em emails categorizados como *spam*. Em comparação com as CDNs, os resultados mostram que máquinas com software de *fast-flux* possuem baixa disponibilidade de acesso, visto que podem ser desligadas a qualquer momento. Além disso, foi possível identificar que 30% dos domínios encontrados em SPAM são hospedados em redes de fluxo rápido.

Nazario e Holz [Nazario and 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., a proposta de Nazario e Holz extrai informações da zona autorizativa do domínio (`SOA`) e a distribuição dos endereços IP encontrados.

[9]Projeto Dmoz disponível em:http://www.dmoz.org/

Caso um domínio seja confirmado como suspeito, esse endereço é adicionado em uma base de dados que, posteriormente será utilizada como referência de reputação de IPs. Os resultados mostraram que 76% do total de 928 endereços de domínios `.com` são redes de fluxo rápido e que 83% dos endereços IP são utilizados em múltiplos domínios, demonstrando, consequentemente, uma sobreposição de endereços IP por nomes de domínios.

De uma maneira geral, as soluções propostas dependem de fontes externas para identificar tráfego suspeito, tal como listas negras, engenharia reversa de código e base de dados de *spam* [Holz et al. 2008a, Nazario and Holz 2008, Passerini et al. 2008]. Por esses motivos, Perdisci et al. propuseram um modelo passivo para detecção e monitoramento de redes de fluxo rápido a partir da análise de comportamento histórico do tráfego DNS [Perdisci et al. 2009]. Os autores observaram as características do domínio como o tempo de vida (TTL); quantidade de endereços IP que atendem um domínio; a proporção de ASN por domínio; tempo da última consulta do 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. Perdisci et al. classificaram os domínios maliciosos através do algoritmo de árvore de decisão (C4.5). Os resultados mostraram que o modelo proposto acerta em 99.7% e erra 0.002%. No entanto, apesar do percentual de precisão ser elevado, o modelo utiliza o comportamento histórico do DNS e uma janela de monitoramento extensa; 24 horas.

Huang et al. [Huang et al. 2010] apresentaram um mecanismo de detecção de *fast-flux* em tempo real, baseado em localização geográfica dos hosts. De uma maneira geral, enquanto os trabalhos anteriores utilizavam a relação temporal (`TTL`), Huang et al. utilizaram a relação espacial (latitude, longitude) entre os hosts para detecção de domínios maliciosos. Os autores partem do princípio que tais redes possuem máquinas infectadas em todo globo. Por tanto, o fuso horário é utilizado para calcular a dispersão entre os hosts da rede. Isto é, redes benignas possuem concentração de hosts na mesma zona de fuso horário, enquanto as redes maliciosas a distribuição é mais dispersa.

Os resultados, em comparação com [Holz et al. 2008a], mostraram que a solução espacial é mais rápida na detecção de *botnets* de fluxo rápido (0,5 segundos). A solução proposta consegue identificar 98.16% dos domínios maliciosos, enquanto Holz et. al. 90.80%. Huang et al. utilizaram uma base de dados de terceiros para obter a localização geográfica de endereços IP. Isso significa que, caso um endereço IP não esteja cadastrado na base de dados, o sistema proposto não atingirá o seu objetivo.

Mecanismos como *single-flux* e *double-flux* mostram que *botmasters* buscam estratégias para dificultar o interrompimento da *botnet* . Ao combinar redes de fluxo rápido com algoritmos que geram nomes de domínios dinamicamente, o problema torna-se ainda mais crítico. Antonakakis et al. [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*, *spamtraps* 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.

Os autores em [Shin and Gu 2010] 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 outros tipos de identificação são necessários.

Stone-Gross et al. [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.

Já Yadav et al. [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 Tabela 3.3 sumariza as principais características utilizadas pelos trabalhos citados nesta Seção para detecção de redes e domínios de fluxo rápido. São apresentadas características comuns aos dois tipos de anomalia (*fast-flux* e *domain-flux*), onde as redes de fluxo rápido são identificadas através da correlação dos recursos de rede, enquanto os

Tabela 3.3. Características 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

domínios de fluxo rápido são investigados pelas características do nome do domínio.

Vale ressaltar que além das características ilustradas na Tabela 3.3, 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.

3.3.4. Protocolos P2P

A computação ponto-a-ponto (`P2P`) tem promovido uma grande modificação nos padrões de uso da Internet nos últimos anos. Sua grande vantagem, em relação à computação cliente/servidor, é possibilitar a colaboração direta entre os usuários, sem depender de servidores administrados por terceiros [Kamienski et al. 2005]. Contudo, a falta de controle sobre a troca de conteúdo compartilhado nessas redes, tornou as aplicações `P2P` a principal fonte de compartilhamento de conteúdo ilegal na Internet.

Para detectar estes tipos de *botnets*, a maioria dos trabalhos propostos são base-

ados em técnicas de análise do tráfego P2P. Características como o par endereço IP e porta, protocolos utilizados, *strings* especificas contidas no pacote e a taxa de sucesso de comunicação do nó são comumente empregadas na análise do tráfego.

Para entender o funcionamento das redes P2P, Grizzard et al. (2007) investigaram o comportamento de computadores infectados com *worm Trojan.Peacomm* que utiliza o tráfego P2P para comunicação. Após a infecção de uma vítima, o *malware* utiliza uma URL para baixar os binários da injeção secundária, conforme descrito no ciclo de vida das *botnets* 3.2.4. Tal binário permite ao *bot* encontrar outros pares da rede P2P e oferecer serviços como *relay SMTP* ou coleta de informações sigilosas como o número de cartão de crédito e credenciais de banco.

Douceur [Douceur 2002] apresentou um ataque, denominado de Sybil, que explora a falta de uma autoridade de certificação em redes P2P para assumir múltiplas identidades e assim controlar a rede. Este tipo de ataque se baseia no fato de que é praticamente impossível, em sistemas computacionais distribuídos, que nós que não se conhecem apresentem identidades distintas convincentes. Um ataque Sybil é aquele em que um atacante subverte o sistema de reputação de uma rede P2P, criando muitas identidades e usando-as para obter vantagens. Baseado neste conceito, Davis et al. (2008) investigaram a viabilidade de um ataque *Sybil* para interromper a proliferação da *botnet Storm*. Ataques do tipo *sybil* simulam falsos pares (nós) na rede para sobrescrever o índice da tabela distribuída dos membros verdadeiros. Ao enviar dados falsos aos membros da *botnet*, um *bot* legítimo não consegue estabelecer a comunicação com outros *bots* da rede, pois as informações repassadas para envenenamento sobrescrevem os endereços dos *bots* válidos.

Nagaraja et al. [Nagaraja et al. 2010] desenvolveram o BotGrep, uma ferramenta para detectar *botnets* P2P baseadas nas trocas de mensagens entre pares de nós da rede, também conhecido como grafos de comunicação. Esta abordagem se baseia em grafos gerados a partir da estrutura de C&C da *botnet* P2P. O algoritmo BotGrep particiona de forma iterativa o grafo de comunicação, estreitando-o para destacar apenas os componente da *botnet*. Esta análise pressupõe que os hosts pertencente as *botnets* P2P tendem a ser mais conectados do que outros hospedeiros, o que permite sua separação de outros hosts benignos. Os resultados experimentais indicaram que esta técnica pode localizar a maioria dos *bots* com baixa taxa de falsos positivos.

Outro modelo de comunicação ponto-a-ponto é o serviço de mensagens curtas (SMS) utilizado por aparelhos celulares [Hua and Sakurai 2013]. Hua e Sakurai apresentam uma arquitetura de botnet baseada no protocolo *Bluetooth* e em mensagens de texto curtas utilizadas para contactar o servidor de comando e controle. Os autores demonstraram que tal *bonet* é possível simulando três cenários de comportamentos distintos, onde 100 *bots* estão distribuídos aleatoriamente em uma área de $100m^2$. O principal problema dessa arquitetura é que a mesma não é escalável, pois está limitada ao raio de alcance da tecnologia *Bluetooth*.

3.4. Ferramentas e Técnicas para Detecção de Botnets

Esta Seção apresenta as ferramentas e técnicas frequentemente utilizadas para a detecção de *botnets*. A literatura sobre o tema é extensa e, por isso, este trabalho são descritas

apenas algumas ideias e ferramentas comumente usadas para identificar comportamentos suspeitos na rede. Contudo, o material apresentado é suficiente para oferecer um visão geral a respeito deste assunto.

O processo de coleta do tráfego de rede deve seguir um roteiro predeterminado, respeitando os principais objetivos da pesquisa, para que dados irrelevantes ou desnecessários não sejam processados nessa etapa. Vale ressaltar que questões legais sobre monitoramento e armazenamento do tráfego devem ser consideradas. Portanto este trabalho assume que qualquer processo de coleta e armazenamento do tráfego, executado com base nas ferramentas e técnicas aqui apresentadas, possui algum amparo ético e legal.

Primeiro, será demonstrado como o tráfego de rede pode ser coletado através de ferramentas de captura (*sniffers*). Em seguida, são apresentados exemplos de bibliotecas de programação de rede que auxiliam no desenvolvimento de ferramentas específicas para coleta do tráfego. Finalmente será apresentado um estudo de caso que demonstra como a análise dos dados coletados pode ser utiliza para o processo de identificação de *botnets*.

3.4.1. Coleta e Armazenamento do Tráfego de Rede

Para coletar o tráfego de rede é preciso identificar possíveis pontos de coleta na rede. Em redes de computadores que utilizam comutadores (*switch*), o tráfego de rede deve ser replicado em um porta secundária (*span port*), a qual terá um *sniffer* capaz de processar e armazenar os pacotes de rede.

Entre as aplicações de *sniffers* mais conhecidas, podem ser citadas as ferramentas `TCPDump`[10], `Wireshark`[11] e `Snort`[12]. Para ilustrar como o tráfego pode ser coletado por tais ferramentas, considere o exemplo da Listagem 3.2. Por definição do protocolo `IRC`, os clientes da rede devem estabelecer uma conexão com o servidor através da porta `6667` [Oikarinen and Reed 1993]. Entretanto, como descrito na seção 3.3.1, os *botmasters* podem utilizar faixas de endereçamento de portas para evadir sistemas de intrusão. No exemplo em questão, o filtro do `TCPDump` coleta o tráfego `IRC` nas portas `TCP 6660-6669` e `7000` através da interface de rede `eth0`. Portanto, essa regra permite identificar e capturar o tráfego de qualquer aplicação que utilize estas portas.

Listagem 3.2. Exemplo de coleta de tráfego `IRC` através do `TCPDump`

```
1 tcpdump -n -l -i eth0 tcp portrange 6660-6669 or tcp port 7000
```

Semelhante às operações ilustradas pelo `TCPDump`, a ferramenta `Wireshark` oferece ao usuário da aplicação mecanismos de monitoramento do tráfego de rede através de uma interface gráfica mais amigável. Por exemplo, para filtrar as requisições `GET` de uma *botnet* baseada no tráfego `HTTP` (e.g.: `Zeus`) um usuário deve adicionar a seguinte opção `http.request.method == "GET"` no campo de filtros da ferramenta `Wireshark`, como ilustrado na Figura 3.9. Vale ressaltar que tal filtro também pode ser utilizado para criar estatísticas dos sites que tiveram o maior número de requisições originadas pelos *bots*.

[10]http://www.tcpdump.org
[11]https://www.wireshark.org
[12]https://www.snort.org

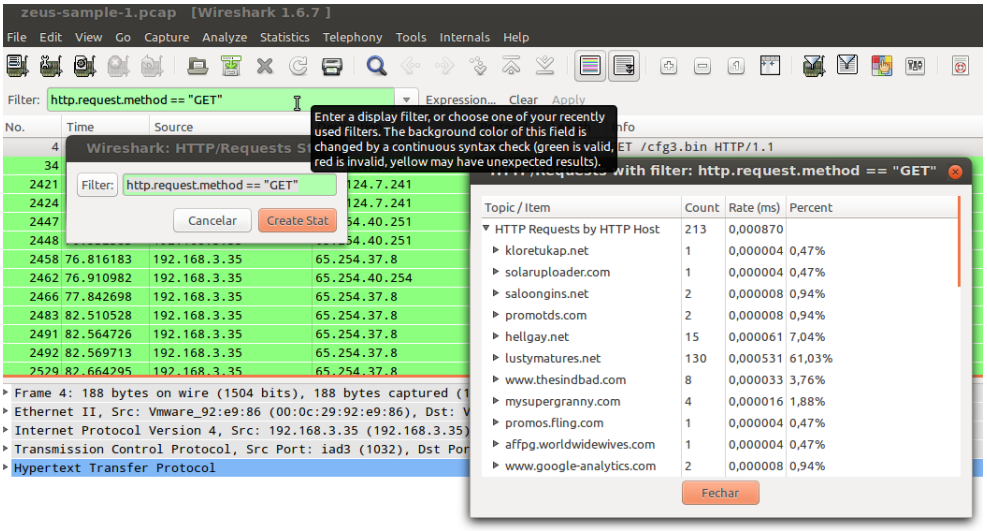

Figura 3.9. Exemplo do total de solicitações do método `GET` enviados por um membro da botnet `Zeus`.

Embora a versão em modo gráfico da ferramenta `Wireshark` permita analisar o tráfego de maneira mais simples, tal solução é inviável em ambientes onde o acesso é remoto e lento. Para superar este problema, o usuário pode utilizar uma outra versão desta ferramenta que opera em modo linha de comando, conhecida como `tshark`, a qual inclui todas opções que a versão em modo gráfico possui, porém em modo texto. Para ilustrar o emprego do `tshark` na coleta do tráfego, a Listagem 3.3 apresenta um exemplo da coleta do tráfego `DNS` durante 5 minutos na interface de rede `eth0` de um servidor Web. Esse filtro realiza uma comparação estatística dos registros de recursos (RR) `A` (`dns.qry.type==1`), `PTR` (`dns.qry.type==12`) e `MX` (`dns.qry.type==15`) considerando uma janela de tempo de 1 minuto (`-qz io,stat,60`). Após 5 minutos ou 300 segundos de atividade (`-a duration:300`), a execução do *sniffer* é interrompida automaticamente.

Listagem 3.3. Exemplo de coleta de tráfego `DNS` através do `tshark`

```
1  tshark -n -l -qz "io,stat,60,dns.qry.type==1,dns.qry.type==12,
       dns.qry.type==15" -a duration:300 -R dns -i eth0
```

Vale ressaltar que os métodos ilustrados acima são utilizados para monitorar o tráfego em tempo real. Portanto, antes de coletar e armazenar o tráfego, é importante planejar uma estratégia que defina os protocolos a serem filtrados, o tempo de coleta e o tamanho dos arquivos armazenados. A Listagem 3.4 ilustra dois exemplos de coleta de tráfego: `HTTP` e `DNS`. Nas linhas de 1-4, o tráfego `HTTP` (`tcp port 80`) é coletado a partir da interface de rede (`eth0`). As opções `-b duration:300` e `-w http.pcap` permitem ao `tshark` armazenar este tráfego em arquivos distintos, formados pelo prefixo *http*, quando o limite de tempo (`300`) for alcançado. De maneira semelhante, as linhas de 6-9 ilustram o processo de coleta do tráfego `DNS` (`udp port 53`), onde as opções do temporizador `-G 300` e o formato do nome do arquivo `-w "dns_%Y%m%d.%H%M%S.pcap"` permitem ao `TCPDump` criar múltiplos arquivos com prefixo *dns*.

Listagem 3.4. Exemplo de coleta de tráfego HTTP e DNS através do TCPDump

```
1 tshark  -n -l -i eth0 \
2         -b duration:300 \
3         -w http.pcap \
4         tcp port 80
5
6 tcpdump -n -l -i eth0 \
7         -G 300 \
8         -w "dns_%Y%m%d.%H%M%S.pcap" \
9         udp port 53
```

Apesar de poderosas, nem todos os elementos de redes (roteadores e *switches*) oferecem suporte a execução do Wireshark e TCPDump. Para essa situações, existem duas alternativas: *i*) o fabricante do equipamento deve fornecer uma ferramenta proprietária para suprir as operações de coleta e gerenciamento do tráfego da rede; e *ii*) o equipamento deve oferecer suporte a algum padrão de coleta de tráfego IP. Por exemplo, o protocolo Netflow [Claise 2004] é um recurso, que foi introduzido em roteadores Cisco, cuja função é coletar o tráfego de redes IP, tanto na saída quanto na entrada de uma interface.

Um fluxo de dados Netflow consiste numa tupla contendo o endereço IP de origem e destino, portas de origem e destino, protocolo, campos do cabeçalho do protocolo TCP como SYN, RST e FIN, tipo de serviço, total de pacotes, total de bytes, hora inicial e final que o fluxo foi visto na rede. O formato Netflow é frequentemente utilizado devido à maneira em que os dados do fluxo são organizados. Os fluxos são coletados por meio de algum tipo de técnica de amostragem, uma vez que a coleta de todas as informações do fluxo podem elevar o consumo de CPU de um roteador [Schiller and Binkley 2007].

Para leitura e processamento do formato específico do Netflow é necessário utilizar um conjunto de ferramentas do projeto NFDUMP[13]. A Figura 3.10 apresenta o resultado da execução do comando nfdump em um tráfego de rede previamente armazenado no formato NetFlow. É possível observar a hora inicial que o pacote foi visto, o tempo de duração do total, endereço IP de origem e destino, campos do cabeçalho, tipo de serviço (Tos), total de pacotes, total de bytes, total de pacotes por segundo (pps) e total de bytes por segundo (bps).

```
Date flow start         Duration Proto    Src IP Addr:Port         Dst IP Addr:Port   Flags Tos  Packets    Bytes      pps      bps
2005-08-30 06:53:53.370   63.545 TCP   113.138.32.152:25    ->   222.33.70.124:3575  .AP.SF   0       62     3512        0      442
2005-08-30 06:53:53.370   63.545 TCP    222.33.70.124:3575  ->  113.138.32.152:25    .AP.SF   0       58     3300        0      415
Time window: Aug 30 2005 06:53:53 - Aug 30 2005 06:54:56
```

Figura 3.10. Exemplo do resultado do comando nfdump com a opção da saída estendida ativada.

Apesar dos exemplos ilustrados nesta Seção serem favoráveis para o monitoramento e análise de tráfego, existem situações onde o controle da porta espelhada do roteador ou do *switch* não é possível. Para superar esta limitação, algumas técnicas de redirecionamento de tráfego na camada 2 podem ser aplicadas como *arp spoofing*, ataques de inundação da tabela MAC e ataques de saltos de redes virtual VLAN (*vlan hopping*).

[13]http://nfdump.sourceforge.net

O ataque de *arp spoofing* consiste na substituição das informações armazenadas nas tabelas `arp` dos computadores da rede com os dados forjados pelo atacante através de mensagens de *broadcast* [Schiller and Binkley 2007]. Nesse ataque, o computador alvo recebe uma mensagem `arp` informando que o endereço IP do roteador (legítimo) mudou para um novo endereço (malicioso). Desta forma, os hosts que tiveram esse dado alterado, mandarão suas informações para o endereço IP do atacante, o qual será capaz de coletar o tráfego.

No caso de um *switch*, os atacantes se aproveitam do fato de que, para manter a comunicação ponto a ponto, este equipamento deve manter uma tabela que contenha todos os endereços `MAC` das máquinas e suas respectivas portas [Bhaiji 2009]. Assim, se um grande número de endereços `MAC` forem injetados em uma única porta, a tabela mantida pelo *switch* será inundada. Nessa situação, o *switch* não saberá qual endereço `MAC` pertence a vítima, permitindo que os hosts ligados a esse equipamento recebam todos os pacotes transmitidos na rede.

Finalmente, o ataque de saltos de `VLAN` acontece quando um atacante se conecta a uma `VLAN` para ter acesso ao tráfego de outras `VLANs` [Schiller and Binkley 2007]. Para este ataque funcionar, o usuário pode simular o comportamento de uma porta *trunk* em seu computador. Uma porta *trunk* corresponde é uma porta que é atribuída para transportar o tráfego de todas as `VLANs` que são acessíveis através de um *switch* específico. O protocolo DTP (*Dynamic Trunk Protocol*) permite a interligação e a configuração automática entre `VLANs` através da porta *trunk* [Cisco 2014]. Desta forma, o atacante consegue ter acesso a todas as `VLANs` da rede para coleta de tráfego.

3.4.2. Bibliotecas de Programação de Rede

As ferramentas ilustradas na Seção anterior apenas coletam o tráfego de rede, cabendo ao pesquisador, desenvolver soluções próprias para analisar e interpretar os resultados obtidos, tais como *parsers* e *scripts*. Para agilizar tal processo de investigação, pesquisadores podem utilizar bibliotecas de programação de rede para automatizar a leitura e análise do tráfego.

Entre as bibliotecas de programação de rede mais populares, encontra-se a `libpcap` [Jacobson et al. 2004]. Desenvolvida em 2003, a `libpcap` oferece uma interface de programação que permite capturar pacotes que passam através das interfaces de rede. Esta biblioteca pode ser instalada em sistemas baseados em `UNIX` e `Windows`. Ao iniciar uma sessão de captura, filtros semelhantes ao `TCPDump` podem ser especificados para capturar pacotes relevantes para análise.

De modo resumido, para coletar tráfego de rede a partir da `libpcap`, os seguintes passos podem ser seguidos:

- **Leitura do tráfego**: deve ser definido qual a fonte de coleta do tráfego. A `libpcap` oferece basicamente dois métodos: leitura em tempo real e leitura *off-line*. A leitura em tempo real coleta o tráfego através das interfaces de rede do computador, enquanto o último pode ler tráfego a partir de arquivos salvos no formato da `libpcap`.
- **Filtros**: são sequências de caracteres que seguem o formato *BSD Packet Filter*

[McCanne and Jacobson 1993]. Por exemplo, a sequência `tcp port 80` pode ser utilizada para filtrar o tráfego `HTTP` na porta 80.

- **Encapsulamento de dados**: alguns sistemas operacionais e formatos de arquivo rede organizam os dados da camada de enlace de maneira distinta. Por exemplo, o modelo `Ethernet` (IEEE 802.11) difere do modelo `RAW IP`, visto que o último não possui informações da camada de enlace de rede.
- **Leitura em camadas**: após a coleta de um pacote, é possível obter informações das camadas de rede individualmente. No entanto, cabe ao desenvolvedor implementar os métodos necessários para acessar tais dados.

A Listagem 3.5 apresenta um exemplo de código fonte na linguagem `C` capaz de monitorar o tráfego a partir de uma interface rede. Por questões de espaço, os trechos de códigos que validam retornos de funções e ponteiros estão omitidos. Os comandos apresentados nas linhas 21 a 26 são responsáveis pela captura do tráfego.

Listagem 3.5. Código básico para coleta de pacotes usando a `libpcap`.

```
1  #include <stdio.h>
2  #include <pcap.h>
3  #include <sys/socket.h>
4  #include <netinet/in.h>
5  #include <arpa/inet.h>
6
7  void dumpPkt(u_char *args, const struct pcap_pkthdr *header, const u_char *packet);
8
9  int main(int argc, char **argv)
10 {
11         char *device = argv[1];              // interface de rede
12         char errBuf[PCAP_ERRBUF_SIZE];       // armazena erros
13         pcap_t *handler = NULL;              // tratador
14         struct pcap_pkthdr header;           // info. pacote capturado
15         struct bpf_program fp;               // filtro
16         char filter_rule[] = "tcp port 80"; // regras do filtro - apenas trafego http
17         bpf_u_int32 mask;                    // mascara ip da rede
18         bpf_u_int32 netip;                   // ip da rede
19         const u_char *pacote;                // pacote em si
20
21         handler = pcap_open_live(device, 65535, 1, 0, errBuf);
22         pcap_lookupnet(device, &netip, &mask, errBuf);
23         pcap_compile(handler, &fp, filter_rule, 0, netip);
24         pcap_setfilter(handler, &fp);
25         pcap_loop(handler, -1 , dumpPkt, NULL);
26         pcap_close(handler);
27         return 0;
28 }
```

Vale destacar a instrução descrita na linha 25, a qual apresenta o método de *callback* `dumpPtk` que sempre é executado quando um pacote de rede for lido pela função `pcap_loop`. O `dumpPtk` recebe um ponteiro (`*packet`) que pode ser utilizado para acessar às camadas do pacote capturado. Esta função de *callback* deve conter rotinas que façam o tratamento de dados por camadas tais como enlace, rede e transporte. Desta forma, o acesso à carga útil (*payload*) do protocolo `HTTP` pode ser obtido através das operações de ponteiros como `payload = (u_char *)(packet + SIZE_ETHERNET + size_ip + size_tcp)`. No caso do tráfego `HTTP` , antes de extrair os dados para análise, é importante que o desenvolvedor reorganize os fluxos `TCP` conforme foram enviados na rede.

Embora a `libpcap` seja utilizada por diversas aplicações e seja portada como extensão para outras linguagens como `Perl`, `Ruby`, `Python` e `Java`, o acesso a carga útil do pacote não é trivial. Nesse sentido, algumas soluções mais recentes buscam agilizar este acesso como a `libtrace` e `scapy`.

A biblioteca `libtrace`, desenvolvida em [Alcock et al. 2012], possui inúmeras vantagens em relação à `libpcap`, incluindo capacidade de leitura e escrita de arquivos compactados, menor consumo de memória e acesso direto às camadas de rede através de métodos prontos. O acesso nativo de leitura aos formatos compactados permite que análise do tráfego de rede seja mais rápida, através de paralelismo computacional. Para extrair as informações do cabeçalho `TCP`, o desenvolvedor pode utilizar a seguinte função `trace_get_tcp(pacote)`, a qual retorna um ponteiro para tal posição do cabeçalho. No entanto, mesmo oferecendo algumas facilidades na manipulação de camadas, ainda é necessário que o desenvolvedor implemente suas rotinas para tratar a carga útil do pacote.

A `scapy` tem como objetivo abordar os problemas demonstrados acima de maneira mais simples [Biondi 2010]. Desenvolvida em `Python`, a biblioteca permite ao usuário enviar, coletar, dissecar e forjar pacotes de rede. No cenário de coleta de tráfego, o desenvolvedor pode definir uma função de *callback*, a qual funciona de maneira semelhante a `libtrace`.

Para ilustrar o processo de coleta usando a `scapy`, a Listagem 3.6 apresenta um código que captura o tráfego `TCP` destinado à porta 80. A função `http_callback` (linhas 4-16) recebe os pacotes fornecidos pela função `sniff` (linha 18) e é capaz de imprimir o endereço IP de destino do servidor `HTTP` (linha 13) e o cabeçalho da solicitação (linha 14).

Listagem 3.6. Exemplo de *sniffer* `HTTP` usando a biblioteca `scapy`.

```
1  #!/usr/bin/env python
2  from scapy.all import *
3
4  def http_callback(pkt):
5          httpHead = False
6          try:
7                  httpHead = pkt[Raw].load
8
9                  if httpHead:
10                         dstIP = pkt[IP].dst
11
12                         print "-----------------"
13                         print dstIP
14                         print httpHead
15         except:
16                 pass
17
18 sniff(prn=http_callback, filter="tcp dst port 80", store=0)
```

Essa Seção demonstrou como as ferramentas e bibliotecas de programação de rede podem ser utilizadas no processo de coleta de tráfego. A seguir, será demonstrado um estudo de caso de detecção de tráfego malicioso.

3.4.3. Estudo de Caso

Esta Seção apresenta um estudo de caso de comportamentos suspeitos na rede. Para investigação, um ambiente virtualizado composto por quatro com o sistema operacional `Debian GNU/Linux` na versão estável. A arquitetura possui um servidor e três clientes, onde cada máquina virtual possui 1GB de memória RAM e softwares de linguagem de programação como `C`, `Perl` e `Python`. Além disso, softwares extras foram adicionados no servidor para auxiliar na análise de tráfego incluindo os descritos na seção 3.4.1 e ferramentas como `RRDTool`[14], `MRTG`[15] e `NFSen`[16].

Inicialmente, o servidor é habilitado com o software `nfdump` e `MRTG`, o qual será utilizado para capturar fluxos de tráfego. Em seguida, os clientes iniciarão um ataque de negação de serviço a partir da ferramenta `T50`[17]. O objetivo é lançar ataques do tipo `SYN` contra o serviço web do servidor e coletar fluxos de dados para análise posterior.

Cada máquina executou, em dois momento distintos, o comando `t50` fornecendo os argumentos para estabelecimento de conexão (`-S`) na porta de destino 80 (`--dport 80`) do endereço IP do servidor `10.208.8.81`. Além disso, foram adicionados os comandos para inundação (`--flood`), que continua enviando pacotes à vítima até o cancelamento do `t50` e a opção `--turbo`, que aumenta o número de `threads` durante o ataque. É possível observar dois picos de ingresso de tráfego na Figura 3.11 durante o ataque. Apesar de gerar certo volume e deixar o acesso mais lento, tal ataque não interrompeu o serviço web da vítima. No entanto, em casos reais, como a `botnet Torpig`, o volume gerado por milhares de máquinas pode interromper a operação de grandes provedores de Internet [Stone-Gross et al. 2009].

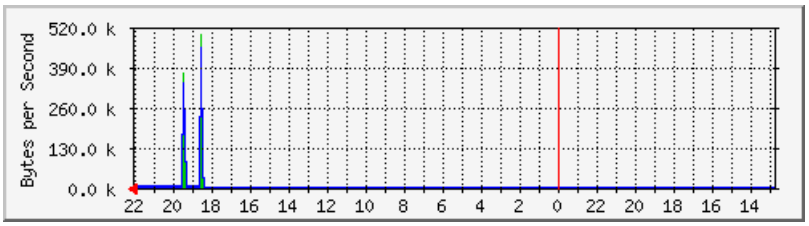

Figura 3.11. Exemplo de tráfego coletado durante um ataque de negação de serviço do tipo `SYN`.

Outra maneira de observar o ataque descrito na Figura 3.11 é a utilização dos métodos estatísticos que o `tshark` oferece. A Listagem 3.7 apresenta um resumo comparativo entre comportamentos de tráfego legítimo e `DDoS`. As linhas 6 e 18 representam o filtro de pacotes do tipo `SYN`, enquanto as linhas 7 e 19 apenas as consultas do tipo `ACK`, ou seja, a confirmação que o pacote foi recebido.

O comportamento do tráfego legítimo foi obtido através de consultas randômicas, utilizando o comando `wget`. É possível observar que existe uma proporção entre as consultas (*frames*) do tipo `SYN` (*Column #0*) e do tipo `ACK` (*Column #1*) nesse tráfego. Por outro lado, a quantidade de consultas do tipo `SYN` no tráfego `DDoS` é desproporcional

[14] http://oss.oetiker.ch/rrdtool/
[15] http://oss.oetiker.ch/mrtg/index.en.html
[16] http://nfsen.sourceforge.net/
[17] http://sourceforge.net/projects/t50/

em relação às consultas do tipo ACK. Vale ressaltar que em ataques de DDoS do tipo SYN, o endereço IP de origem é forjado. Portanto, a vítima desse ataque não recebe a confirmação, pois o endereço IP verdadeiro não enviou o pacote [Eddy 2007].

Para interromper os ataques de negação de serviço, existem algumas abordagens que podem ser utilizadas como filtragem, redução do temporizador que sinaliza um pacote SYN recebido, o emprego de cache de pacotes SYN, filtragem por estado de pacote e proteção através de proxy na rede. A descrição de tais técnicas podem ser observadas na RFC 4987 [Eddy 2007].

Listagem 3.7. Comparação entre comportamento de acesso legítimo e um DDoS

```
1
2  # Tráfego Legítimo
3  ===================================================================
4  IO Statistics
5  Interval: 60.000 secs
6  Column #0: tcp.flags.syn==1 && tcp.flags.ack==0
7  Column #1: tcp.flags == 0x0010
8                  |   Column #0    |   Column #1
9  Time            |frames|  bytes  |frames|  bytes
10 000.000-060.000      23      1702     92      6072
11 060.000-120.000      30      2220    120      7920
12 120.000-180.000      35      2590    137      9042
13
14 # Tráfego Malicioso (DDoS)
15 ===================================================================
16 IO Statistics
17 Interval: 60.000 secs
18 Column #0: tcp.flags.syn==1 && tcp.flags.ack==0
19 Column #1: tcp.flags == 0x0010
20                 |   Column #0    |   Column #1
21 Time            |frames|  bytes  |frames|  bytes
22 000.000-060.000   27422   1645320      0         0
23 060.000-120.000    8504    510240      0         0
24 120.000-180.000    1746    104802      6       396
```

3.5. Considerações Finais

Diferente dos outros tipo de programas maliciosos, cujo único objetivo é inutilizar os sistemas computacionais afetados, as *botnets* fazem parte de uma categoria muito mais complexa, onde o objetivo é, na maioria dos casos, atingir algum ganho financeiro. E seus desenvolvedores e mantenedores tem se esforçado muito para manter suas criações indetectáveis, garantindo a continuidade de suas atividades maliciosas.

Em suas primeiras encarnações, o ganho financeiro derivado das atividades de *botnets* era restrito à interceptação de dados, como números de cartões de crédito e informações bancárias, que poderiam ser utilizados em operações comerciais fraudulentas. Deste lá, este quadro evoluiu para o momento a partir do qual os proprietários das *botnets* passaram a vender ou alugar seus serviços, o que criou todo um mercado negro de recursos computacionais sequestrados, que são continuamente empregados como base de ações maliciosas. Seus ataques agora não são mais indiscriminados e aleatórios. Seus alvos são bem definidos e tem um preço que, infelizmente, alguém está disposto a pagar.

Assim, para proteger o seu "modo de vida", os desenvolvedores de *botnets* não têm medido esforços para ludibriar todas as tentativas de interromper suas atividades. Técnicas de polimorfismo e encriptação de mensagens, para ludibriar sistemas baseados em assinaturas; uso de protocolos de comunicação considerados benignos, para enganar os tradicionais filtros de pacotes; e a dissimulação de suas atividades, para impedir a aparição de anomalias que dariam pistas de sua presença dentro dos sistemas contaminados, são alguns dos exemplos. Isoladas ou em conjunto, todas estas técnicas tem sido amplamente empregadas para evitar cada novo processo de detecção que foram sendo desenvolvidos durante os últimos anos. Além disso, a inclusão dos dispositivos móveis na infraestrutura das *botnets* cria toda uma nova gama de desafios, exigindo ainda mais criatividade dos pesquisadores dedicados ao combate da ameaça que as *botnets* representam.

Dada a complexidade deste problema, dificilmente será encontrada uma solução única que seja capaz de lidar com todas as técnicas empregadas para proteger as ações das *botnets*. Entretanto, a combinação de soluções de detecção e a pesquisa e o desenvolvimento de novas ferramentas de combate a esta ameaça promete não dar trégua nesta contínua batalha contra estes programas maliciosos e seus desenvolvedores.

Este Capítulo apresentou informações relevantes para o entendimento do problema das *botnets*, incluindo um conjunto de definições básicas, a evolução histórica, o ciclo de vida, as principais arquiteturas utilizadas na construção, os principais tipos de ataques suportados, as abordagens básicas. Também apresentou uma visão geral do estado da arte sobre a detecção de *botnets*, bem como um apanhado de ferramentas comumente empregadas para coletar os indícios da presença de uma *botnet* em sistemas computacionais. Os autores esperam que este conteúdo sirva de ponto de partida para futuros trabalhos e possa despertar o interesse de pesquisadores para esta área.

Referências

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

[Alata et al. 2007] Alata, E., Nicomette, V., Kaâniche, M., Dacier, M., and Herrb, M. (2007). Lessons learned from the deployment of a high-interaction honeypot. *CoRR*, abs/0704.0858.

[Alcock et al. 2012] Alcock, S., Lorier, P., and Nelson, R. (2012). Libtrace: a packet capture and analysis library. *SIGCOMM Comput. Commun. Rev.*, 42(2):42–48.

[Antonakakis et al. 2010] Antonakakis, M., Perdisci, R., Dagon, D., Lee, W., and Feamster, N. (2010). Building a dynamic reputation system for dns. In *Proceedings of the 19th USENIX conference on Security*, USENIX Security'10, pages 18–18, Berkeley, CA, USA. USENIX Association.

[Barbosa and Souto 2009] Barbosa, K. R. S. and Souto, E. (2009). Análise passiva do tráfego dns da internet brasileira. In *IX Simpósio Brasileiro em Segurança da Informação e Sistemas Computacionais (SBSeg 2009)*, pages 203–216, Campinas.

[Barr 1996] Barr, D. (1996). RFC 1912: Common DNS operational and configuration errors. http://www.ietf.org/rfc/rfc1912.txt.

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

[Berghel 2001] Berghel, H. (2001). The code red worm. *Commun. ACM*, 44(12):15–19.

[Bhaiji 2009] Bhaiji, Y. (2009). Understanding, preventing, and defending against layer 2 attacks.

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

[Biondi 2010] Biondi, P. (2010). Welcome to scapyś documentation! Acessado em 12/09/2013.

[BITAG 2012] BITAG (2012). Snmp reflected amplification ddos attack mitigation. Technical report, Broadband Internet Technical Advisory Group.

[Boshmaf et al. 2011] Boshmaf, Y., Muslukhov, I., Beznosov, K., and Ripeanu, M. (2011). The socialbot network: When bots socialize for fame and money. In *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, pages 93–102, New York, NY, USA. ACM.

[Cai and Zou 2012] Cai, T. and Zou, F. (2012). Detecting http botnet with clustering network traffic. In *Wireless Communications, Networking and Mobile Computing (WiCOM), 2012 8th International Conference on*, pages 1–7.

[Callegati et al. 2009] Callegati, F., Cerroni, W., and Ramilli, M. (2009). Man-in-the-middle attack to the https protocol. *Security Privacy, IEEE*, 7(1):78–81.

[Calvet et al. 2010] Calvet, J., Davis, C. R., Fernandez, J. M., Marion, J.-Y., St-Onge, P.-L., Guizani, W., Bureau, P.-M., and Somayaji, A. (2010). The case for in-the-lab botnet experimentation: Creating and taking down a 3000-node botnet. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, pages 141–150, New York, NY, USA. ACM.

[Canavan 2005] Canavan, J. (2005). The evolution of malicious irc bots.

[Carpine et al. 2013] Carpine, F., Mazzariello, C., and Sansone, C. (2013). Online irc botnet detection using a soinn classifier. In *Communications Workshops (ICC), 2013 IEEE International Conference on*, pages 1351–1356.

[Chawathe et al. 2003] Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., and Shenker, S. (2003). Making gnutella-like p2p systems scalable. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '03, pages 407–418, New York, NY, USA. ACM.

[Choi and Lee 2012] Choi, H. and Lee, H. (2012). Identifying botnets by capturing group activities in dns traffic. *Computer Networks*, 56(1):20–33.

[Choi et al. 2009] Choi, H., Lee, H., and Kim, H. (2009). Botgad: Detecting botnets by capturing group activities in network traffic. In *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE*, COMSWARE '09, pages 2:1–2:8, New York, NY, USA. ACM.

[CISCO 2013] CISCO (2013). Cisco annual security report. Technical report.

[CISCO 2014] CISCO (2014). Cisco annual security report. Technical report.

[Cisco 2014] Cisco (2014). Layer 2 lan port configuration. Acesssado em 10/09/2014.

[Claise 2004] Claise, B. (2004). RFC 3954 - Cisco Systems NetFlow Services Export Version 9.

[Cohen 1987] Cohen, F. (1987). Computer viruses: Theory and experiments. *Computers & Security*, 6(1):22 – 35.

[Colajanni et al. 2008] Colajanni, M., Gozzi, D., and Marchetti, M. (2008). Collaborative architecture for malware detection and analysis. In Jajodia, S., Samarati, P., and Cimato, S., editors, *Proceedings of The Ifip Tc 11 23rd International Information Security Conference*, volume 278 of *IFIP - The International Federation for Information Processing*, pages 79–93. Springer US.

[Dagon et al. 2007] Dagon, D., Gu, G., Lee, C., and Lee, W. (2007). A taxonomy of botnet structures. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, pages 325 –339.

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

[Davis et al. 2008] Davis, C., Fernandez, J., Neville, S., and McHugh, J. (2008). Sybil attacks as a mitigation strategy against the storm botnet. In *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, pages 32–40.

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

[Douceur 2002] Douceur, J. (2002). The sybil attack. In Druschel, P., Kaashoek, F., and Rowstron, A., editors, *Peer-to-Peer Systems*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer Berlin Heidelberg.

[Dunham 2009] Dunham, K. (2009). *Mobile Malware Attacks and Defense*. Syngress Publishing.

[Eddy 2007] Eddy, W. (2007). TCP SYN Flooding Attacks and Common Mitigations. RFC 4987 (Informational).

[Egele et al. 2008] Egele, M., Scholte, T., Kirda, E., and Kruegel, C. (2008). A survey on automated dynamic malware-analysis techniques and tools. *ACM Comput. Surv.*, 44(2):6:1–6:42.

[Eslahi et al. 2013] Eslahi, M., Hashim, H., and Tahir, N. (2013). An efficient false alarm reduction approach in http-based botnet detection. In *Computers Informatics (ISCI), 2013 IEEE Symposium on*, pages 201–205.

[Feily et al. 2009] Feily, M., Shahrestani, A., and Ramadass, S. (2009). A survey of botnet and botnet detection. In *Emerging Security Information, Systems and Technologies, 2009. SECURWARE '09. Third International Conference on*, pages 268 –273.

[Ferrer et al. 2010] Ferrer, Z., Ferrer, M. C., and Researchers, C. I. S. (2010). In-depth analysis of hydraq. *The face of cyberwar enemies unfolds. ca isbu-isi white paper*, 37.

[Fossi et al. 2011] Fossi, M., Egan, G., Haley, K., Turner, D., Johnson, E., Johnson, E., and Adams, T. (2011). Symantec global internet security threat report. trends for 2010. Technical Report 17, Symantec.

[Fossi et al. 2010] Fossi, M., Turner, D., Johnson, E., Johnson, E., and Adams, T. (2010). Symantec global internet security threat report. Technical Report 17, Symantec.

[Freiling et al. 2005] Freiling, F., Holz, T., and Wicherski, G. (2005). Botnet tracking: Exploring a root-cause methodology to prevent distributed denial-of-service attacks. In Vimercati, S., Syverson, P., and Gollmann, D., editors, *Computer Security - ESORICS 2005*, volume 3679 of *Lecture Notes in Computer Science*, pages 319–335. Springer Berlin Heidelberg.

[Freitas et al. 2014] Freitas, C., Benevenuto, F., and Veloso, A. (2014). Socialbots: Implicações na segurança e na credibilidade de serviços baseados no twitter. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos- SBRC 2014*, pages 603–616.

[Furnell and Ward 2004] Furnell, S. and Ward, J. (2004). Malware evolution: Malware comes of age: The arrival of the true computer parasite. *Netw. Secur.*, 2004(10):11–15.

[Grizzard et al. 2007] Grizzard, J. B., Sharma, V., Nunnery, C., Kang, B. B., and Dagon, D. (2007). Peer-to-peer botnets: Overview and case study. In *Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets*, HotBots'07, pages 1–1, Berkeley, CA, USA. USENIX Association.

[Gu et al. 2008a] Gu, G., Perdisci, R., Zhang, J., and Lee, W. (2008a). Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *Proceedings of the 17th Conference on Security Symposium*, SS'08, pages 139–154, Berkeley, CA, USA. USENIX Association.

[Gu et al. 2008b] Gu, G., Zhang, J., and Lee, W. (2008b). Botsniffer: Detecting botnet command and control channels in network traffic. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium.*

[Guo et al. 2006] Guo, F., Chen, J., and cker Chiueh, T. (2006). Spoof detection for preventing dos attacks against dns servers. In *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*, pages 37–37.

[Hachem et al. 2011] Hachem, N., Ben Mustapha, Y., Granadillo, G., and Debar, H. (2011). Botnets: Lifecycle and taxonomy. In *Network and Information Systems Security (SAR-SSI), 2011 Conference on*, pages 1–8.

[Haddadi et al. 2014] Haddadi, F., Morgan, J., Filho, E., and Zincir-Heywood, A. (2014). Botnet behaviour analysis using ip flows: With http filters using classifiers. In *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on*, pages 7–12.

[Holz et al. 2008a] Holz, T., Gorecki, C., Rieck, K., and Freiling, F. C. (2008a). Measuring and detecting fast-flux service networks. In *NDSS*.

[Holz et al. 2008b] Holz, T., Steiner, M., Dahl, F., Biersack, E., and Freiling, F. (2008b). Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, LEET'08, pages 9:1–9:9, Berkeley, CA, USA. USENIX Association.

[Hua and Sakurai 2013] Hua, J. and Sakurai, K. (2013). Botnet command and control based on short message service and human mobility. *Computer Networks*, 57(2):579 – 597. Botnet Activity: Analysis, Detection and Shutdown.

[Huang et al. 2010] Huang, S.-Y., Mao, C.-H., and Lee, H.-M. (2010). 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, pages 101–111, New York, NY, USA. ACM.

[Husna et al. 2008] Husna, H., Phithakkitnukoon, S., and Dantu, R. (2008). Traffic shaping of spam botnets. In *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*, pages 786–787.

[Ishibashi et al. 2005] Ishibashi, K., Toyono, T., Toyama, K., Ishino, M., Ohshima, H., and Mizukoshi, I. (2005). 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, pages 159–164, New York, NY, USA. ACM.

[Jacobson et al. 2004] Jacobson, V., Leres, C., and McCanne, S. (2004). pcap - packet capture library.
urlhttp://www.tcpdump.org/pcap3_man.html (acessado em 01/08/2014).

[Jakobsson and Ramzan 2008] Jakobsson, M. and Ramzan, Z. (2008). *Crimeware: Understanding New Attacks and Defenses (Symantec Press)*. Addison-Wesley Professional, 1 edition.

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

[Kamienski et al. 2005] Kamienski, C., Souto, E., Rocha, J., Domingues, M., Callado, A., and Sadok, D. (2005). Colaboração na internet ea tecnologia peer-to-peer. In *XXV Congresso da Sociedade Brasileira de Computação*, pages 1407–1454.

[Kang and Zhang 2009] Kang, J. and Zhang, J.-Y. (2009). Application entropy theory to detect new peer-to-peer botnet with multi-chart cusum. In *Electronic Commerce and Security, 2009. ISECS '09. Second International Symposium on*, volume 1, pages 470–474.

[Kumar and Selvakumar 2011] Kumar, P. A. R. and Selvakumar, S. (2011). Distributed denial of service attack detection using an ensemble of neural classifier. *Computer Communications*, 34(11):1328 – 1341.

[Lee et al. 2008] Lee, J.-S., Jeong, H., Park, J.-H., Kim, M., and Noh, B.-N. (2008). The activity analysis of malicious http-based botnets using degree of periodic repeatability. In *Security Technology, 2008. SECTECH '08. International Conference on*, pages 83–86.

[Levy 2003] Levy, E. (2003). The making of a spam zombie army. dissecting the sobig worms. *Security Privacy, IEEE*, 1(4):58–59.

[Li et al. 2009a] Li, C., Jiang, W., and Zou, X. (2009a). Botnet: Survey and case study. In *Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference on*, pages 1184–1187.

[Li et al. 2009b] Li, Z., Liao, Q., and Striegel, A. (2009b). Botnet economics: Uncertainty matters. In Johnson, M., editor, *Managing Information Risk and the Economics of Security*, pages 245–267. Springer US.

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

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

[Livadas et al. 2006] Livadas, C., Walsh, R., Lapsley, D., and Strayer, W. (2006). Using machine learning technliques to identify botnet traffic. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 967–974.

[Lu and Ghorbani 2008] Lu, W. and Ghorbani, A. A. (2008). Botnets detection based on irc-community. In *Proceedings of the Global Communications Conference, 2008. GLOBECOM 2008, New Orleans, LA, USA, 30 November - 4 December 2008*, pages 2067–2071.

[Ma et al. 2010] Ma, X., Guan, X., Tao, J., Zheng, Q., Guo, Y., Liu, L., and Zhao, S. (2010). A novel irc botnet detection method based on packet size sequence. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–5.

[Mazzariello 2008] Mazzariello, C. (2008). Irc traffic analysis for botnet detection. In *Information Assurance and Security, 2008. ISIAS '08. Fourth International Conference on*, pages 318–323.

[McCanne and Jacobson 1993] McCanne, S. and Jacobson, V. (1993). The bsd packet filter: A new architecture for user-level packet capture. In *Proceedings of the USENIX Winter 1993 Conference Proceedings on USENIX Winter 1993 Conference Proceedings*, USENIX'93, pages 2–2, Berkeley, CA, USA. USENIX Association.

[McCarty 2003a] McCarty, B. (2003a). Automated identity theft. *Security Privacy, IEEE*, 1(5):89–92.

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

[McFee 2003] McFee (2003). Virus profile: W32/sdbot.worm. Acessado em 12/09/2014.

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

[Moore et al. 2002] Moore, D., Shannon, C., and claffy, k. (2002). Code-red: A case study on the spread and victims of an internet worm. In *Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurment*, IMW '02, pages 273–284, New York, NY, USA. ACM.

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

[Mulliner and Seifert 2010] Mulliner, C. and Seifert, J.-P. (2010). Rise of the ibots: Owning a telco network. In *Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on*, pages 71–80.

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

[Nagaraja et al. 2010] Nagaraja, S., Mittal, P., Hong, C.-Y., Caesar, M., and Borisov, N. (2010). Botgrep: finding p2p bots with structured graph analysis. In *Proceedings of the 19th USENIX conference on Security*, USENIX Security'10, pages 7–7, Berkeley, CA, USA. USENIX Association.

[Nappa et al. 2010] Nappa, A., Fattori, A., Balduzzi, M., DellAmico, M., and Cavallaro, L. (2010). Take a deep breath: A stealthy, resilient and cost-effective botnet using skype. In Kreibich, C. and Jahnke, M., editors, *Detection of Intrusions and Malware, and Vulnerability Assessment*, volume 6201 of *Lecture Notes in Computer Science*, pages 81–100. Springer Berlin Heidelberg.

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

[Oikarinen and Reed 1993] Oikarinen, J. and Reed, D. (1993). Internet relay chat protocol.

[Passerini et al. 2008] Passerini, E., Paleari, R., Martignoni, L., and Bruschi, D. (2008). Fluxor: Detecting and monitoring fast-flux service networks. In Zamboni, D., editor, *Detection of Intrusions and Malware, and Vulnerability Assessment*, volume 5137 of *Lecture Notes in Computer Science*, pages 186–206. Springer Berlin Heidelberg.

[Peng et al. 2007] Peng, T., Leckie, C., and Ramamohanarao, K. (2007). Survey of network-based defense mechanisms countering the dos and ddos problems. *ACM Comput. Surv.*, 39(1).

[Perdisci et al. 2009] Perdisci, R., Corona, I., Dagon, D., and Lee, W. (2009). Detecting malicious flux service networks through passive analysis of recursive dns traces. *Computer Security Applications Conference, Annual*, 0:311–320.

[Perdisci et al. 2010] Perdisci, R., Lee, W., and Feamster, N. (2010). Behavioral clustering of http-based malware and signature generation using malicious network traces. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, NSDI'10, pages 26–26, Berkeley, CA, USA. USENIX Association.

[ProofPoint 2014] ProofPoint (2014). Proofpoint uncovers internet of things (iot) cyberattack. Acessado: 10/04/2014.

[Rebane 2011] Rebane, J. C. (2011). *The Stuxnet Computer Worm and Industrial Control System Security*. Nova Science Publishers, Inc., Commack, NY, USA.

[Rodionov and Matrosov 2011] Rodionov, E. and Matrosov, A. (2011). The evolution of tdl: Conquering x64. *ESET, June*.

[Rodríguez-Gómez et al. 2013] Rodríguez-Gómez, R. A., Maciá-Fernández, G., and García-Teodoro, P. (2013). Survey and taxonomy of botnet research through life-cycle. *ACM Comput. Surv.*, 45(4):45:1–45:33.

[Royal 2008] Royal, P. (2008). Analysis of the kraken botnet. *Damballa*, 9.

[Salusky and Danford 2008] Salusky, W. and Danford, R. (2008). Know your enemy: Fast-flux service networks. an ever changing enemy. *The Honeynet Project*.

[Schiller and Binkley 2007] Schiller, C. and Binkley, J. (2007). *Botnets: The Killer Web Applications*. Syngress Publishing.

[Shen and Hasegawa 2010] Shen, F. and Hasegawa, O. (2010). Self-organizing incremental neural network and its application. In *Proceedings of the 20th International Conference on Artificial Neural Networks: Part III*, ICANN'10, pages 535–540, Berlin, Heidelberg. Springer-Verlag.

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

[Shin et al. 2011] Shin, S., Lin, R., and Gu, G. (2011). Cross-analysis of botnet victims: New insights and implications. In Sommer, R., Balzarotti, D., and Maier, G., editors, *Recent Advances in Intrusion Detection*, volume 6961 of *Lecture Notes in Computer Science*, pages 242–261. Springer Berlin Heidelberg.

[Shoch and Hupp 1982] Shoch, J. F. and Hupp, J. A. (1982). The “worm” programs—early experience with a distributed computation. *Commun. ACM*, 25(3):172–180.

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

[Souza et al. 2013] Souza, A., Barbosa, K. R. S., and Feitosa, E. (2013). Identificando spam no twitter através da anáise empírica dos trending topics brasil. In *SBSeg 2013 - WTICG (2013)*, pages 394–403.

[Stoica et al. 2001] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '01, pages 149–160, New York, NY, USA. ACM.

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

[Strayer et al. 2008] Strayer, W., Lapsely, D., Walsh, R., and Livadas, C. (2008). Botnet detection based on network behavior. In Lee, W., Wang, C., and Dagon, D., editors, *Botnet Detection*, volume 36 of *Advances in Information Security*, pages 1–24. Springer US.

[Studer 2011] Studer, R. (2011). Economic and technical analysis of botnets and denial-of-service attacks. *Communication systems IV*, page 19.

[Tenebro 2008] Tenebro, G. (2008). W32.waledac - threat analysis. Technical report, Symantec.

[Thonnard and Dacier 2011] Thonnard, O. and Dacier, M. (2011). A strategic analysis of spam botnets operations. In *Proceedings of the 8th Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference*, CEAS '11, pages 162–171, New York, NY, USA. ACM.

[Tiirmaa-Klaar et al. 2013] Tiirmaa-Klaar, H., Gassen, J., Gerhards-Padilla, E., and Martini, P. (2013). Botnets: How to fight the ever-growing threat on a technical level. In *Botnets*, SpringerBriefs in Cybersecurity, pages 41–97. Springer London.

[Turner et al. 2007] Turner, D., Entwisle, S., Denesiuk, M., Fossi, M., Blackbird, J., and McKinney, D. (2007). Internet security threat report. 2007 trends. Technical Report 17, Symantec.

[Tyagi and G.Aghila 2011] Tyagi, A. K. and G.Aghila (2011). A wide scale survey on botnet. *International Journal of Computer Applications*, 34(9):10–23. Published by Foundation of Computer Science, New York, USA.

[Virvilis et al. 2013] Virvilis, N., Gritzalis, D., and Apostolopoulos, T. (2013). Trusted computing vs. advanced persistent threats: Can a defender win this game? In *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*, pages 396–403.

[Wang et al. 2009] Wang, P., Wu, L., Aslam, B., and Zou, C. (2009). A systematic study on peer-to-peer botnets. In *Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on*, pages 1–8.

[Xiang et al. 2011] Xiang, C., Binxing, F., Lihua, Y., Xiaoyi, L., and Tianning, Z. (2011). Andbot: towards advanced mobile botnets. In *Proceedings of the 4th USENIX conference on Large-scale exploits and emergent threats*, LEET'11, pages 11–11, Berkeley, CA, USA. USENIX Association.

[Yadav et al. 2012] Yadav, S., Reddy, A. K. K., Reddy, A. L. N., and Ranjan, S. (2012). Detecting algorithmically generated domain-flux attacks with dns traffic analysis. *IEEE/ACM Trans. Netw.*, 20(5):1663–1677.

[Yen and Reiter 2010] Yen, T.-F. and Reiter, M. (2010). Are your hosts trading or plotting? telling p2p file-sharing and bots apart. In *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, pages 241–252.

[Zeidanloo and Manaf 2010] Zeidanloo, H. R. and Manaf, A. B. A. (2010). Botnet detection by monitoring similar communication patterns. In *International Journal of Computer Science and Information Security (IJCSIS)*, volume Vol. 7.

[Zhao et al. 2013] Zhao, D., Traore, I., Sayed, B., Lu, W., Saad, S., Ghorbani, A., and Garant, D. (2013). Botnet detection based on traffic behavior analysis and flow intervals. *Computers & Security*, 39, Part A(0):2 – 16. 27th {IFIP} International Information Security Conference.

