MINISTÉRIO DA DEFESA

EXÉRCITO BRASILEIRO

DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA

INSTITUTO MILITAR DE ENGENHARIA

1º Ten RÔMULLO GIRARDI MOREIRA
Al DIEGO FURTADO GONÇALVES

ESTUDO SOBRE DETECÇÃO DE *BOTNETS*

Rio de Janeiro
2012
INSTITUTO MILITAR DE ENGENHARIA

1º Ten RÔMULLO GIRARDI MOREIRA
Al DIEGO FURTADO GONÇALVES

ESTUDO SOBRE DETECÇÃO DE *BOTNETS*

Trabalho apresentado ao Curso de Engenharia de Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção da graduação em Engenharia de Computação.

Orientador: Major Sérgio dos Santos Cardoso Silva – M.Sc.

Rio de Janeiro
2012
INSTITUTO MILITAR DE ENGENHARIA

1º Ten RÔMULLO GIRARDI MOREIRA
Al DIEGO FURTADO GONÇALVES

ESTUDO SOBRE DETECÇÃO DE *BOTNETS*

Trabalho apresentado ao Curso de Engenharia de Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção da graduação em Engenharia de Computação.

Orientador: Major Sérgio dos Santos Cardoso Silva – M.Sc. do IME

_____

SÉRGIO DOS SANTOS CARDOSO SILVA – M.Sc., do IME

_____

ANDERSON FERNANDES PEREIRA DOS SANTOS – D.Sc., do IME

_____

JULIO CESAR DUARTE – D.Sc., do IME

Rio de Janeiro
2012

RESUMO

O presente trabalho tem por finalidade realizar um estudo sobre *botnets*, delimitando-se na análise dos mecanismos de detecção das *botnets*. Para tal, a partir da pesquisa bibliográfica, a obras relativas ao tema, criaram-se subsídios para alcançar o objetivo proposto. Inicialmente, realizou-se uma revisão dos protocolos HTTP, DNS, IRC e P2P, juntamente com um estudo sobre as *botnets*. Como resultado, buscou-se a consolidação de um estudo comparativo entre as técnicas de detecção das *botnets,* tendo por base os protocolos HTTP, DNS, IRC e P2P.

Palavras chave: *botnets*, detecção.

ABSTRACT

This study aims to conduct a search about botnets, delimiting the analysis of mechanisms to detect botnets. To achieve this goal, subsidies were created after a bibliographic research about this topic. Initially, we developed a review of HTTP, DNS, IRC and P2P protocols, along with a study about botnets. As a result, we sought to consolidate a comparative study between the techniques for detecting botnets, based on HTTP, DNS, IRC and P2P protocols.

Key words: botnets, detecting.

SUMÁRIO

1 INTRODUÇÃO

Várias ameaças de segurança e crimes digitais tem surgido com o aumento do número de computadores. Um dos modernos tipos de ameaça à segurança de sistemas computacionais é a *botnet* - rede formada por computadores infectados e monitorados por um agente externo.

Esta pesquisa pauta-se no estudo sobre este tipo de rede maliciosa, delimitando-se no estudo sobre os mecanismos de detecção das *botnets*, tendo por base os protocolos HTTP, DNS, IRC e P2P.

1.1. Motivação

Segundo Rajab (2006), aproximadamente um quarto dos computadores participam de alguma *botnet*. Esta parcela pode tornar-se ainda mais representativa, tendo em vista o espantoso crescimento que esta atividade ilícita vem apresentando nos dias atuais. Sendo assim, tal percepção justifica os esforços de elucidação do tema, uma vez que torna-se fundamental entender os mecanismos de funcionamento, detecção e desativação das *botnets* para analisar o seu comportamento e conter os seus ataques.

Além disso, este estudo busca servir de subsídio para outras pesquisas sobre o assunto, como por exemplo, os trabalhos em andamento no Instituto Militar de Engenharia: PIBITI (Programa Institucional de Bolsas de Iniciação em Desenvolvimento Tecnológico e Inovação), TD (Trabalho Dirigido), IP (Iniciação à Pesquisa), PFC (Projeto de Fim de Curso) e teses de Mestrado e Doutorado.

1.2. Objetivos

O objetivo geral desta pesquisa é: realizar um estudo sobre os mecanismos de detecção das *botnets*, tendo por base os protocolos HTTP, DNS, IRC e P2P. Para atingir este objetivo, são definidos os seguintes objetivos específicos:

1) Revisar os protocolos HTTP, DNS, IRC e P2P;
2) Estudar as *botnets*: definição, componentes, tipos, objetivos e ciclo de vida;
3) Consolidar um estudo comparativo entre as técnicas de detecção das *botnets,* tendo por base os protocolos HTTP, DNS, IRC e P2P.

1.3. Metodologia

O trabalho foi dividido em etapas que percorrem desde definições, conceitos básicos e revisão sistemática de trabalhos correlatos até a identificação das técnicas de detecção das *botnets*. Sendo assim, com base na estrutura da metodologia adotada, chegou-se a seguinte definição do problema:

Com base na revisão sistemática, quais seriam as técnicas de detecção das *botnets,* tendo por base os protocolos HTTP, DNS, IRC e P2P?

Como solução, buscou-se realizar uma pesquisa bibliográfica com o intuito de sintetizar um estudo comparativo entre as técnicas de detecção das *botnets,* tendo por base os protocolos HTTP, DNS, IRC e P2P.

1.4. Organização

Este trabalho está organizado em 4 (quatro) capítulos.

No segundo capítulo, realiza-se uma revisão sobre protocolos. Para tal, estudam-se os protocolos HTTP, DNS, IRC e P2P.

No terceiro capítulo, abordam-se aspectos conceituais e básicos relacionados às *botnets*. Para tal, tratam-se definições, componentes, tipos, objetivos e ciclo de vida das *botnets*.

No quarto capítulo, abordam-se as técnicas de detecção das *botnets*.

Nessa perspectiva, reuniram-se informações que possibilitaram a constituição de um embasamento teórico consistente para subsidiar a formulação da conclusão da pesquisa. Sendo assim, ao final do trabalho, consolidou-se um estudo comparativo entre as técnicas de detecção das *botnets,* tendo por base os protocolos HTTP, DNS, IRC e P2P.

2 PROTOCOLOS

Faz-se necessário uma revisão sobre os protocolos HTTP, DNS, IRC e P2P, uma vez que estes serão os protocolos tratados na consolidação deste estudo.

2.1. Protocolo HTTP (*HyperText Transfer Protocol*)

O protocolo HTTP (*HyperText Transfer Protocol*) é “o protocolo usado para a comunicação entre um navegador e um servidor *Web* ou entre máquinas intermediárias e servidores *Web”*(COMER, 2006, p.323).

Segundo Comer (2006), a *Web* consiste em um conjunto de documentos (páginas *Web*) acessíveis pela *Internet*. Cada página recebe um nome que é seu identificador único denominado URL (*Uniform Resource Locator*).

- Estrutura do URL: protocolo://máquina:porta/caminho/recurso
- Explicação da estrutura do URL:

  - Protocolo: Geralmente, o protocolo é o HTTP;

  - Máquina: Especifica o nome do domínio ou o endereço IP do computador que opera o servidor HTTP;

  - Porta: O HTTP tem como porta padrão a porta 80;

  - Caminho: Identifica a localização de um documento no servidor HTTP;

  - Recurso: Identifica um documento no servidor HTTP.

2.1.1. Características do HTTP

O protocolo HTTP apresenta as seguintes características:

1) Opera em nível de aplicação;
2) Utiliza o protocolo de transporte TCP (*Transmission Control Protocol*) que é confiável e orientado a conexão;
3) Utiliza, por padrão, a porta 80;

4) Ao estar estabelecida a conexão TCP, é enviada uma requisição HTTP para que o servidor envie uma mensagem de resposta HTTP;
5) Cada requisição HTTP é autocontida. O servidor não guarda um histórico de transações realizadas;
6) A comunicação HTTP ocorre nos dois sentidos;
7) O navegador *Web* guarda, em cache, uma cópia de cada página *Web* que recebe do servidor. Caso o navegador necessite requisitar esta página novamente, só será necessário verificar, junto ao servidor, se a cópia armazenada está atualizada;
8) É permitido a colocação de máquinas (servidores *proxy*), entre cliente e servidor, que guardam em cache cópias de páginas *Web*. Este artifício busca acelerar o processo de requisição à páginas, além de diminuir o fluxo de acesso ao servidor (COMER, 2006).

2.1.2. Funcionamento básico do HTTP

O protocolo HTTP funciona, basicamente, de acordo com a seguinte dinâmica entre cliente e servidor, como ilustrado na Figura 1:

1) Cliente inicia uma conexão TCP com o servidor via porta 80;
2) Servidor aceita a conexão TCP do cliente (estabelecimento da conexão);
3) Cliente envia uma requisição HTTP servidor Web (servidor HTTP). O servidor envia mensagem de resposta;
4) A conexão TCP é fechada.

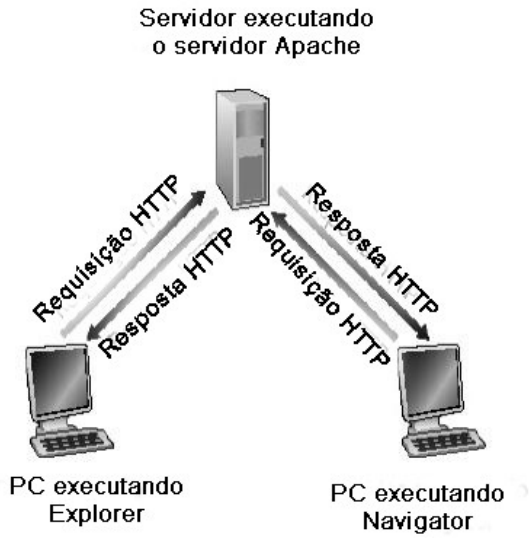

Figura 1: Funcionamento básico do HTTP
Fonte: Kurose (2006)

2.1.3. Tipos de conexão

De acordo com a versão do HTTP, existem dois tipos de conexões possíveis:

1) Versão HTTP/1.0: Utiliza a conexão não persistente. Neste tipo de conexão, há o estabelecimento de uma conexão TCP para cada objeto a ser transferido;
2) Versão HTTP/1.1: Utiliza a conexão persistente. Neste tipo de conexão, vários objetos podem ser transferidos apenas com o estabelecimento de uma conexão TCP. Para que este procedimento ocorra, segundo Comer (2006), o HTTP envia o tamanho do objeto antes de cada resposta. Caso o servidor não saiba o tamanho do objeto, ele informa ao cliente este fato, envia a resposta e fecha a conexão.

2.1.4. Requisição HTTP

A mensagem de requisição HTTP tem como elemento principal o método. Este método indica qual é o objetivo da mensagem de requisição realizada.

A seguir, apresenta-se a Tabela 1, com os principais métodos constituintes das mensagens de requisição HTTP.

Tabela 1: Métodos de requisição HTTP
Fonte: Tanembaum (2003)

Método	Descrição
GET	Solicita a leitura de uma página na Web
HEAD	Solicita a leitura de um cabeçalho de página da Web
PUT	Solicita o armazenamento de uma página da Web
POST	Acrescenta a um recurso(por exemplo,uma página da Web)
DELETE	Remove a página da Web
TRACE	Ecoa a solicitação recebida
CONNECT	Reservado para uso futuro
OPTIONS	Consulta certas opções

- Exemplo de requisição HTTP extraído de Kurose (2006):

  GET somedir/page.html HTTP/1.1
  Host: www.someschool.edu
  Connection: close
  User-agent: Mozilla/4.0
  Accept-agent: fr

2.1.5. Resposta HTTP

A mensagem de resposta HTTP tem como elemento principal o status. O status representa o resultado do processamento executado pelo servidor. O status possui o formato NXX, onde:

a) N representa a classe da resposta;
b) XX representa a categoria da resposta.

Atualmente, o protocolo possui cinco classes de respostas, segundo a Tabela 2.

Tabela 1: Classes de resposta HTTP
Fonte: Tanenbaum (2003)

Código	Significado	Exemplos
1xx	Informação	100 = server agrees to handle client's request
2xx	Sucesso	200 = request succeeded; 204 = no content present
3xx	Redirecionamento	301 = page moved; 304 = cached page still valid
4xx	Erro do cliente	403 = forbidden page; 404 = page can not found

5xx	Erro do servidor	500 = internal server error; 503 = try again later

2.2. Protocolo DNS (*Domain Name System*)

O Sistema de Nomes de Domínio (DNS) é um serviço primordial para o funcionamento da *Internet*. Ele é o responsável por converter os nomes (URLs) em endereços IPs e vice-versa. Para uma pessoa que acessa a *Internet* torna-se muito mais fácil lembrar o nome http://www.google.com.br, ao invés de 209.85.193.104 (endereço IP que identifica o servidor que hospeda o site de buscas *Google* na *Internet*).

Kurose (2006) afirma que o DNS é um banco de dados distribuído implementado em uma hierarquia de servidores de nome (servidores DNS), juntamente com o protocolo da camada de aplicação (protocolo DNS) que permite que *hosts* consultem o banco de dados distribuído.

2.2.1. Características do DNS

1) Opera em nível de aplicação;
2) Considera o protocolo de transporte UDP (*User Datagram Protocol*), não confiável e não orientado a conexão, para a realização de consultas. E considera o protocolo de transporte TCP (*Transmission Control Protocol*), confiável e orientado a conexão, para a realização de transferência de zona;
3) Utiliza, por padrão, a porta 53;

2.2.2. Estrutura de funcionamento do DNS

Como citado anteriormente, o DNS é um banco de dados distribuído. A estrutura do DNS é conhecida como “árvore invertida”, como ilustrado na Figura 2:

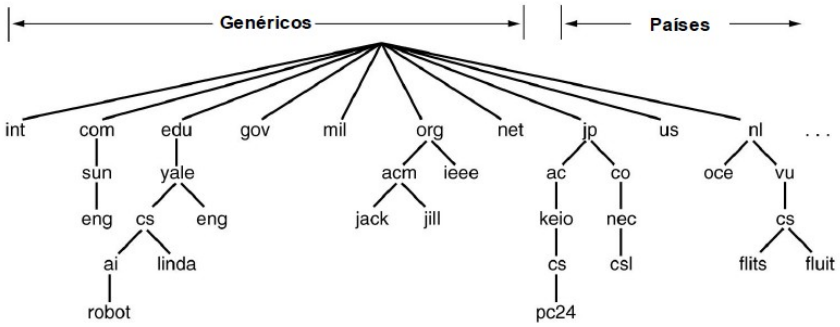

Figura 1: Parte do espaço de nomes de domínios da *Internet*
Fonte: Tanenbaum (2003)

O ponto (“.”) representa o nó raiz da hierarquia – o topo da árvore. Abaixo do nó raiz, encontramos os *Generic Top Domains Levels* (gTLDs). Os gTLDs são uma das categorias do *Top Level Domains (TLDs)* mantidos pelo IANA (*Internet Assigned Numbers Authority*) para uso na *Internet*. Exemplos de gTLDs são os domínios .com, .edu, .org, .gov, dentre outros. Os *Country Code Top Level Domains* – ccTLDs são domínios reservados para um país ou território. Cada país ou território possui uma entidade para gerenciar seu ccTLD (ICANN, 2011). O Brasil, por exemplo, possui a extensão .br para os domínios registrados em nosso país, como ilustrado na Figura 3.

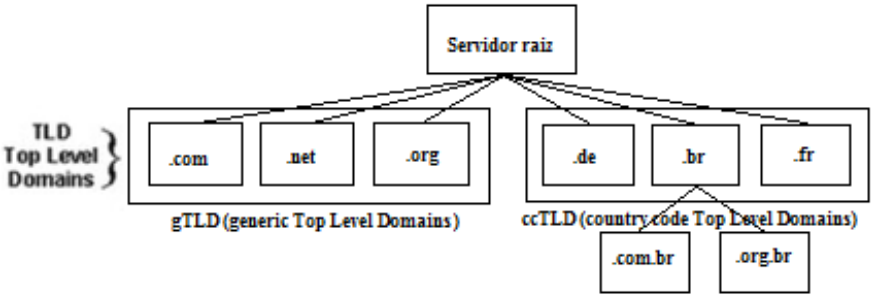

Figura 1: Parte da estrutura dos Top Level Domains (TLDs)
Fonte: Vaz (2011)

Cada nó também é a raiz de uma nova subárvore da árvore total. Cada uma das subárvores representa um domínio no *Domain Name System*. Cada domínio pode ser subdividido em divisões adicionais, chamados de subdomínios DNS.

2.2.3. Registros de recursos (*Resource records*)

A configuração de uma zona ou domínio fica armazenada em um arquivo específico no servidor DNS. Dentro deste arquivo existem *resource records* – RRs.

Os registros de recursos são entradas nos arquivos de zona do servidor DNS que refletem, por exemplo, a resolução de um endereço IP em nome, dentre outras informações. Basicamente, a resposta recebida por um cliente DNS (*resolver*) durante uma consulta resume-se ao *resource record* encontrado no servidor consultado. Tanenbaum (2003, p. 620) conclui que “a principal função do DNS é mapear nomes de domínios em registros de recursos”. Na Tabela 3, verificam-se alguns exemplos de registros de recursos.

Tabela 1: Principais registros de recursos
Fonte: Tanenbaum (2003)

	Significado	Valor
	Início de autoridade	Parâmetros para essa zona
	Endereço IP de um host	Inteiro de 32 bits
	Troca de mensagens de correio	Prioridade,domínio disposto a aceitar correio eletrônico
	Servidor de nomes	Nome de um servidor para este domínio
	Nome canônico	Nome de domínio
	Ponteiro	Nome alternativo de um endereço IP
	Descrição de host	CPU e sistema operacional em ASCII
	Texto	Texto ASCII não interpretado

2.2.4. Tipos de consulta DNS

Em DNS, a consulta pode ser encadeada/interativa ou recursiva:

1) Consulta encadeada/interativa: Servidor contatado responde com o nome de outro servidor de nomes para contato. Ou seja, por não poder responder a solicitação requisitada, ele responde com o nome de outro servidor que possa responder;
2) Consulta recursiva: Transfere a tarefa de resolução do nome para o servidor de nomes consultado.

2.2.5. Componentes do DNS

- *Resolvedores* (Clientes DNS): *Resolvedores* são clientes que acessam os servidores de nomes. Programas que são executados em um *host* e precisam de informações DNS, usam o resolvedor para solicitar informações a um servidor DNS. Outras funções do resolvedor em um *host* são interpretar respostas (que podem ser Registros de Recurso – *RRs* ou um erro) e devolver a informação para os programas que a requisitaram. Os *hosts*

também possuem um *cache* DNS como forma de evitar que consultas a servidores DNS realizadas anteriormente sejam refeitas desnecessariamente (ALBITZ, 2001).

- Servidores DNS autoritativos: Ao receber requisições de resolução de nome, um servidor DNS autoritativo pode responder um endereço caso possua, uma referência caso conheça o caminho da resolução ou uma negação caso não conheça (DE CAMPOS, 2011). Os servidores autoritativos são divididos em servidores *master* e servidores *slave*. Um servidor autoritativo *master* é o servidor principal do domínio. Ele responde as consultas DNS do qual ele possui autoridade. O servidor *slave* é o servidor que responde as consultas DNS caso o servidor *master* falhe (problemas no equipamento, problemas conexão com a internet do servidor *master*, etc.). O servidor *slave* é um servidor que armazena uma cópia dos domínios dos quais o *master* é responsável. Qualquer atualização nos arquivos de zona do servidor master, o servidor *slave* também é atualizado automaticamente através de um procedimento intitulado transferência de zona. Na transferência de zona, os arquivos que contém os RRs são copiados para o servidor *slave*, exatamente como estão no servidor *master*. Após esta transferência, diz-se que os servidores DNS estão sincronizados entre si.
- Servidores DNS locais: ao receber requisições, utiliza o método de consulta recursiva para a resolução de nomes. Os servidores DNS locais possuem cópias das suas últimas consultas realizadas, cópias estas chamadas de *DNS* cache. Assim, não se faz necessário refazer a mesma consulta DNS até um período preestabelecido na configuração do servidor consultado.
- Servidores DNS raiz (*root servers*): Os servidores DNS raiz são os servidores que estão no topo da árvore DNS. Quando a consulta recursiva chega a este servidor, ele é responsável por redirecionar esta consulta para um servidor TLD, ou então relatar que a informação procurada não existe. Estes servidores são cruciais para o bom funcionamento da *Internet*, já que muitas consultas DNS começam nestes servidores. Existem treze *root servers* espalhados pelo mundo, representados pelas doze primeiras letras do alfabeto (letra “A” até “M”). Porém, para cada um destes servidores raiz

existem vários outros duplicados. A Figura 4, a seguir, mostra a localização dos treze *root servers* distribuídos pelo globo terrestre.

Figura 1: Distribuição geográfica dos servidores raiz
Fonte: Michel (2011)

2.3. Protocolo IRC (*Internet Relay Chat)*

O IRC (*Internet Relay Chat)* é um protocolo de comunicação em tempo real muito popular na Internet. O protocolo IRC atrelou o conceito de sistema de conversação em tempo real ao conceito de canal. A ideia dos canais no IRC permitiu que as pessoas se conectassem a discussões específicas, entrando em canais específicos (HAMMAN, 2011).

O uso do protocolo IRC rapidamente espalhou-se e estruturou-se sobre redes de servidores independentes. Cada servidor é conectado a uma rede que é independente das demais redes de IRC. Cada rede possui milhares de canais, inclusive com os mesmos nomes ou tópicos. A Brasnet, por exemplo, é uma das maiores redes brasileiras de IRC.

Cada rede estrutura-se sobre um conjunto de canais com tópicos diferentes. Não é possível, por exemplo, que coexistam dois canais com o mesmo nome em uma mesma rede. Da mesma forma, não é possível que dois usuários possuam o mesmo apelido na rede, ao mesmo tempo.

O IRC utiliza o identificador URI (*Uniform Resource Identifier*).

- Estrutura do URI: irc://<máquina>[:<porta>]/[?<canal>[...]]
- Explicação da estrutura do URI:

  - A existência de colchetes nos itens especifica que estes itens são de caráter opcional na URI;

  - Máquina: Especifica o nome do domínio ou o endereço IP do computador que opera o servidor IRC;

  - Porta: O IRC tem como porta padrão a porta 6667;

  - Canal: Especifica o número do canal a ser utilizado;

2.3.1. Características do protocolo IRC

O IRC possui as seguintes características fundamentais:

1) Opera em nível de aplicação;
2) Considera, geralmente, o protocolo de transporte TCP (*Transmission Control Protocol*), confiável e orientado a conexão;
3) Utiliza, por padrão, a porta 6667;
4) Para se conversar em tempo real no IRC, é preciso além de se conectar a um servidor, entrar em um canal. Além de existirem canais pré-existentes no momento em que alguém se conecta ao IRC (já criados por outras pessoas), qualquer usuário pode criar um canal (ou “sala”) novo sob um novo título. Para criar um canal, basta que o usuário digite o nome do mesmo e peça para entrar. Se o canal escolhido não tiver “dono”, ele será o operador. Caso o mesmo já exista ou possua um “dono”, ele entrará como usuário comum. Toda a estrutura do IRC dá-se através de seus canais, que funcionam como “salas” simbólicas, onde se pode entrar e sair. Cada sala pode ter inúmeros participantes, entretanto permite que exista um limite de usuários para o canal se o “operador” da sala assim o desejar. No entanto, em geral, essa limitação não é utilizada, pois todos querem o maior número possível de pessoas em seu canal.
5) Ao se conectar em um servidor IRC, cada usuário precisa escolher um apelido (*nickname* ou simplesmente *nick*), que seja diferente dos *nicks* que já estão sendo utilizados por outros usuários. Este apelido é a única forma de

identificação que o usuário possui em princípio. Ao entrar no IRC o usuário recebe uma “máscara” que não permite que ele seja identificado (variando de acordo com o servidor e rede utilizados. Alguns servidores não disponibilizam este serviço.). Essa máscara é modificada a cada nova conexão. Alguns programas, que são utilizados para acesso de servidores de IRC, como o mIRC, por exemplo, permitem que o usuário disponibilize outras informações, se desejar.

6) Em um canal de IRC apenas é possível conversar utilizando-se os caracteres ASCII18, ou seja, os caracteres do teclado. É possível enviar arquivos a outros usuários conectados, por meio de conexões diretas, mas a conversa nos canais é puramente textual.
7) No IRC, o espaço público coexiste com o privado. Enquanto na arena de um canal é possível conversar simultaneamente com várias pessoas e todas podem ver aquilo que é digitado por todas, é possível também conversar por mensagens privadas, que um usuário envia a outro e que os demais não podem ver. Para se enviar mensagens privadas, não é preciso que se esteja dentro de um canal.
8) Todas as mensagens trocadas no IRC possuem o caráter de tempo real. São mensagens que, ao contrário do *e-mail*, são escritas para serem lidas quase que imediatamente. É uma comunicação instantânea, apesar de algumas vezes ocorrerem atrasos no envio e recebimento dessas mensagens (conhecido como *lag*).
9) A ferramenta do IRC, como a maioria dos *chats*[1], possibilita que a comunicação seja uma via de mão dupla, um sistema aberto, onde vários agentes ativos podem efetuar trocas comunicativas. É um sistema de diálogo, que possibilita interação mútua.
10) O IRC oferece, aos operadores, maneiras de controlar o comportamento dos usuários, podendo um usuário ser desconectado do canal (*kick*, ou chute), em geral utilizado como advertência, ou banido do canal ou da rede (*ban* ou *kill*) e, neste caso, não poderá mais reconectar-se ao sistema durante um período de tempo. Cada servidor possui regras genéricas de comportamento, que são

1Chat: Neologismo para designar aplicações de conversação em tempo real.

apresentadas ao usuário, quando este se conecta. Cada canal possui regras próprias e níveis de tolerância diferenciados.

2.3.2. Tipos de comunicação IRC

Existem três tipos de comunicação IRC: *Unicast*, *Broadcast* e *Multicast*.

2.3.2.1. Unicast

Em uma comunicação *Unicast* em IRC, uma cópia separada dos dados (ou mensagem) é enviada de sua origem para cada computador cliente que os tenha requerido. Nenhum outro computador na rede precisa processar o tráfego gerado. No entanto, em uma rede com muitos computadores o *Unicast* não é muito eficiente, uma vez que o computador de origem terá que transmitir múltiplas cópias dos dados (haverá sobrecarga da máquina). O *Unicast* tem melhor desempenho em redes de pequeno porte.

2.3.2.2. Broadcast

Nesse tipo de comunicação em IRC, os dados são enviados apenas uma vez, mas para toda a rede. Mesmo os *hosts,* que não realizaram a requisição, receberão os dados. Eles somente não realizam o processamento destes dados.

2.3.2.3. Multicast

Este tipo de comunicação IRC é um padrão de comunicação um para muitos. Ele pode ser visto como uma transmissão para um subconjunto de nós da rede. A existência do canal IRC é dinâmica e a troca de mensagens realizada em um canal só envolve as máquinas relacionadas a este canal. Além disso, a mensagem só será enviada uma vez para cada ligação local.

2.3.3. Possibilidade de execução de scripts

Linguagens de *script* são linguagens de programação executadas do interior de programas e/ou de outras linguagens de programação, não se restringindo a esses ambientes. As linguagens de *script* servem para estender a funcionalidade de um programa e/ou controlá-lo. Os programas escritos em linguagens de *script* são, normalmente, referidos como *scripts*.

O mIRC e suas variações em forma de scripts são os clientes de IRC mais populares em sistemas operativos. Existem clientes chamados de *bots.* Um *bot* é um tipo de *script* que é, normalmente, utilizado para executar tarefas num canal de IRC. Ele é capaz de exibir mensagens aos utilizadores que entram no canal e realizar uma série de rotinas, automaticamente, após algum comando realizado pelo utilizador. Os *bots* servem como pontos de contato de informações permanentes, como se verifica na Figura 5.

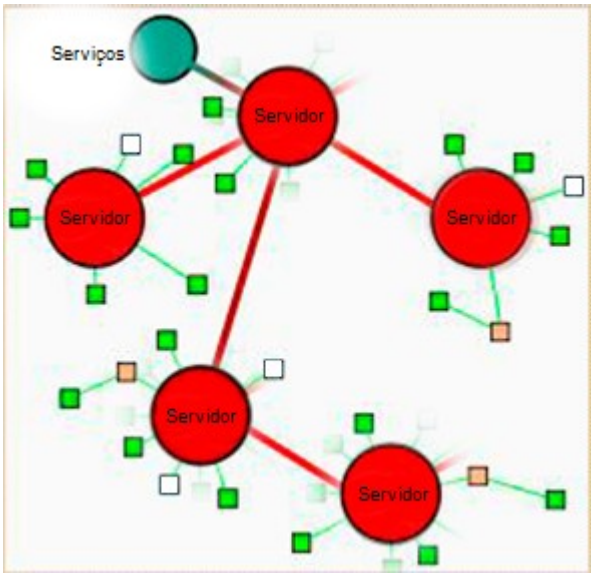

Figura 1: Esquema de uma rede IRC, com clientes normais (cinza) e *bots* (branco).
Fonte: Hamman (2011)

2.3.4. Lista de comandos do IRC

A Tabela 4 mostra os principais tipos de comando utilizados no protocolo IRC:

Tabela 1: Lista de tipos de comando do IRC

Comando	Descrição
Nickserv	Realiza o registro de nomes no servidor
Nickserv set	Realiza a ativação de nomes no servidor
Chanserv	Realiza o registro de canais no servidor
Chanserv set	Realiza a ativação de canais no servidor
Memoserv	Realiza a manipulação de mensagens
Botserv	Realiza o registro de *bots*
Botserv set	Realiza a ativação de *bots*

2.4. P2P (Peer-to-Peer)

Os serviços distribuídos encontrados na *Internet*, na maioria dos casos utilizam a arquitetura cliente/servidor, onde os clientes e os servidores têm papeis fixos, necessitando de alta capacidade de processamento e de banda, tornando esta uma aplicação cara e que precisa de atenção sempre quanto a equipamentos (VOSS JR, 2004). A Figura 6 mostra esta tecnologia, há um processo servidor que provê informações ou recursos para processos clientes ligados na mesma rede, fazendo com que a informação seja centralizada em um único ponto.

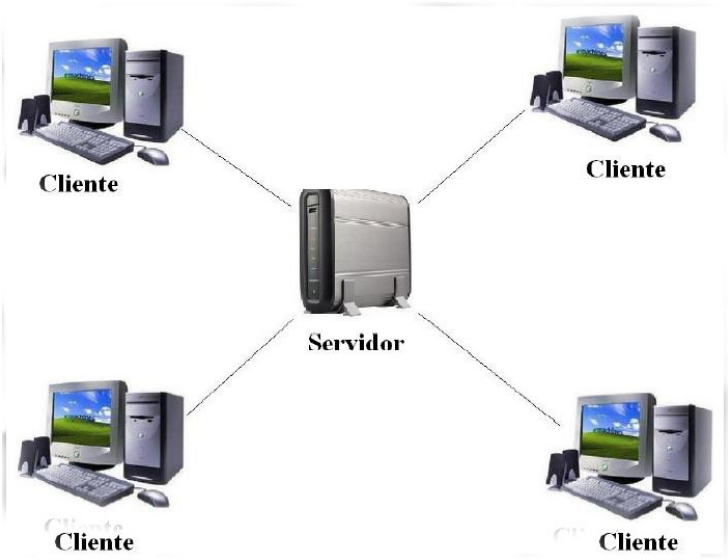

Figura 1: Arquitetura cliente / servidor

Um tipo de arquitetura que tem se desenvolvido no mercado da *Internet*, para solucionar problemas tanto de conexão quanto de disponibilização de recursos, é a P2P (*Peer-to-Peer*). Ela provê uma conexão direta entre dois pontos ou nós da rede (também

chamados de *peers*), eliminando, total ou parcialmente, a necessidade de centralização de dados em um servidor, dando assim maior agilidade aos serviços desejados.

2.4.1. Características da arquitetura P2P

Conforme (BALAKRISHNAN, 2003), as principais características da arquitetura P2P são:

1) Devido ao aumento constante de estações conectadas a *Internet*, as redes P2P tem crescido e tornado sua quantidade de recursos compartilhados cada vez maior e diversificada.
2) Caso ocorra problema em algum *peer*, o sistema não irá parar totalmente, pois os demais podem permanecer atuantes, utilizando recursos e/ou conteúdos existentes;
3) Caso um *peer* não esteja utilizando algum recurso, ele pode compartilhar seu recurso disponível com outros *peers,* aumentando assim a capacidade de processamento da rede.
4) Com o aumento da rede, o tempo de resposta pode variar consideravelmente;
5) Como um *peer* possui a facilidade de entrar e sair da rede, um conteúdo compartilhado pode facilmente deixar de existir na rede;

2.4.2. Classificação das redes P2P

Redes P2P podem ser nativas ou híbridas (YANG, 2001). Sistemas híbridos não são puramente descentralizados, ou seja, utilizam uma estação como concentrador central, para armazenar informações e efetuar processamentos (semelhante a funcionalidade do servidor na arquitetura cliente/servidor). Sistemas Nativos são puramente descentralizados, ou seja, não necessitam de nenhum tipo de concentrador de informações.

Desta forma, as arquiteturas P2P são comumente divididas em três modelos:

1) Com serviço de localização centralizada;
2) Baseada em inundação;

3) Baseada em redes de superposição.

2.4.2.1. Com serviço de localização centralizada

Este modelo é uma arquitetura P2P híbrida, pois ao acessar a rede, a estação conecta-se a um ou mais servidores e envia informações de compartilhamento aos mesmos para que neles sejam gerados índices sobre os recursos disponíveis na estação. Estes índices estarão disponíveis aos demais *peers* conectados ao mesmo servidor.

As informações geradas pelos servidores são distribuídas aos *peers* e quando esses efetuarem uma busca será informado o endereço IP da máquina que aloca os dados. A estação cliente (*peer)* que efetuou a busca pode então conectar-se diretamente à estação proprietária das informações (APPLIED META COMPUTING, 2000).

A Figura 7 exemplifica o funcionamento de um serviço de localização centralizada, onde:

1) O novo cliente se conecta ao servidor e envia uma lista dos seus dados compartilhados;
2) O cliente já registrado no servidor solicita a pesquisa de dados;
3) O servidor retorna uma lista com o endereço de todos os *peers* que contém a informação desejada;
4) O cliente contata o nó que contém o arquivo desejado e o requisita;
5) O *peer* contatado transfere o arquivo ao requisitante.

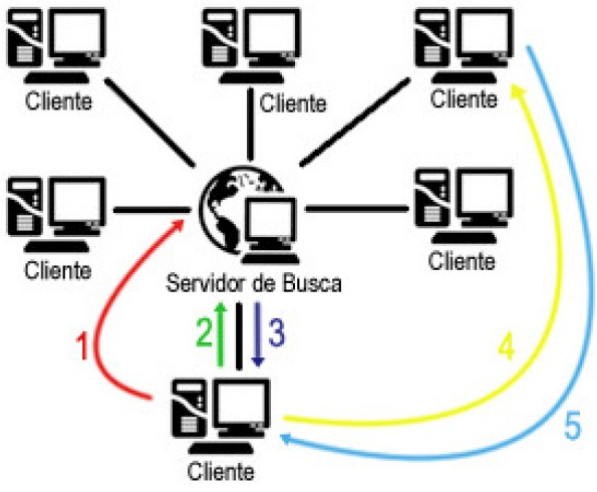

Figura 1: Exemplo de serviço de localização centralizada

Fonte: APPLIED META COMPUTING (2000)

Um exemplo de rede P2P com serviço de localização centralizada é a Napster.

Segundo Oram (2001), a arquitetura básica do Napster requer que clientes conectados a rede enviem uma lista dos arquivos compartilhados ao servidor central, o qual cria um índice dos arquivos. Quando o cliente busca algum arquivo, se conecta ao servidor, e este atualiza a lista de índices e lhe encaminha o endereço do cliente que contém o arquivo buscado, iniciando-se a troca, como se verifica na Figura 8.

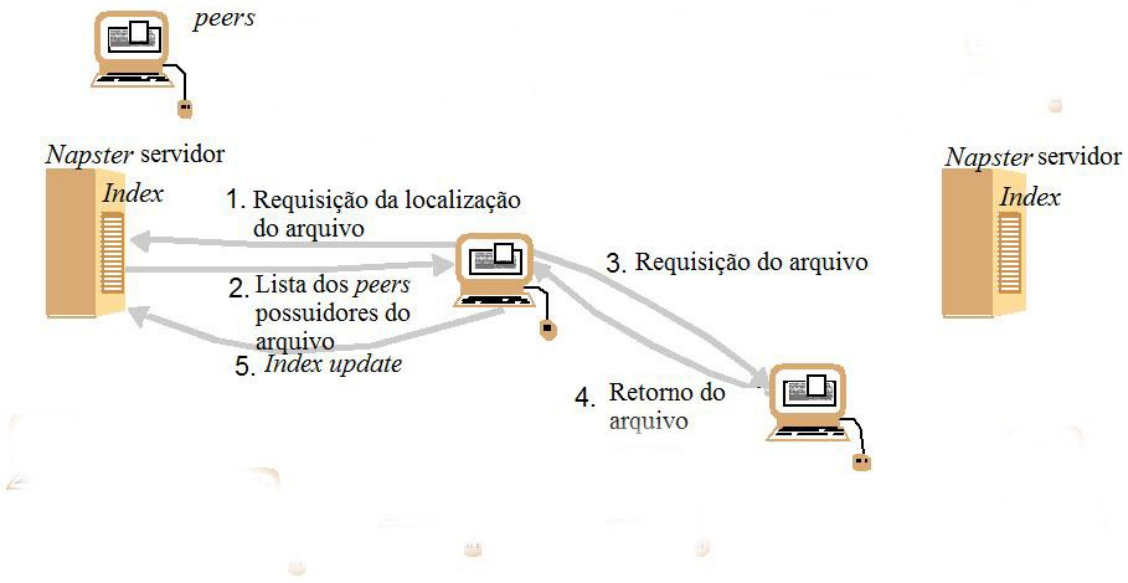

Figura 1: Dinâmica de funcionamento do *Napster*

2.4.2.2. Baseada em inundação

Uma rede baseada em inundação (*Flooding Style Network*) segue uma arquitetura nativa, ou seja, P2P pura. Desta forma, este tipo de rede não possui servidores centrais.

Uma máquina acessa toda a rede P2P simplesmente conhecendo outro *peer.* Cada requisição de compartilhamento de recurso circula livremente entre os *peers*, até que algum *peer* que contenha tal recurso retorne o seu IP. Usa-se como controle dos mecanismos de inundação o TTL (*Time-To-Live*, que significa o tempo de vida de tráfego das informações na rede) da rede, o qual garante que uma requisição não trafegue indefinidamente na rede (APPLIED META COMPUTING, 2000).

A Figura 9 exemplifica uma rede baseada em inundação onde:

1) A estação (*peer* 2) requisita um recurso na rede;

2) Esta requisição percorre os *peers* ainda não pesquisados (1), até que algum resultado seja encontrado e o *peer* proprietário seja marcado como satisfatório;
3) Aqueles dos quais os resultados não foram satisfatórios, serão diferenciados e não serão novamente questionados (3);
4) Aqueles quais os resultados são satisfatórios serão positivados (4);
5) Dentre todos aqueles positivados (4), deve existir um ou mais, com melhor resultado. Este(s) será(ão) o(s) que retornará(ao) o recurso requisitado (5).
6) É criada uma ligação direta (caso não exista) entre o *peer* requisitante e o *peer(s)* requisitados (6).

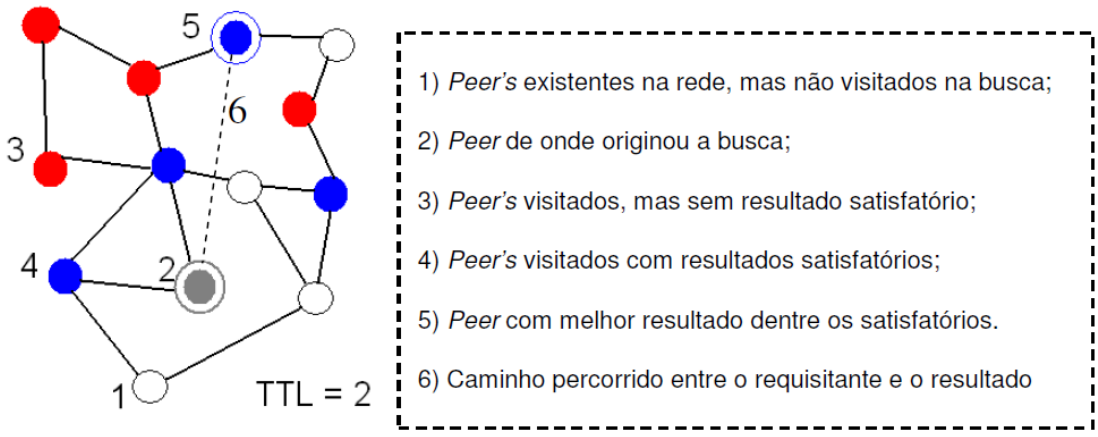

Figura 1: Exemplo de uma rede baseada em inundação

Um exemplo de rede P2P baseada em inundação é a FREENET.

Utilizando uma arquitetura totalmente descentralizada, a FREENET objetiva o uso de sistemas remotos anônimos, ou seja, a execução pela rede de uma série de instruções e processamentos remotamente. Para isto, um usuário pode requisitar à rede um local para armazenar um arquivo, e outro usuário pode responder a requisição, sem ao menos saber quem a fez. Outro ponto é que, se um usuário tem armazenado um arquivo em seu sistema, ele é incapaz de saber quem o armazenou e que está compartilhando este arquivo.

2.4.2.3. Baseada em redes de superposição.

Esta abordagem utiliza algoritmos aleatórios para construir a rede de

sobreposição. Cada nó possui uma lista de vizinhos construída aleatoriamente, e os itens de dados são colocados aleatoriamente em nós.

Um dos objetivos deste tipo de sistema é construir uma rede de sobreposição semelhante a um grafo aleatório. Cada nó mantém uma lista de vizinhos que representam os nós escolhidos aleatoriamente. Esta lista forma uma visão parcial. A visão parcial é formada por entradas, e cada entrada:

- **Identifica outro nó da rede; e**
- **Tem uma idade associada.**

Os nós trocam entradas de sua visão parcial regularmente. São usados dois *threads*[2]:

- ***Thread* ativo: seleciona um nó (*peer*) da sua visão parcial corrente. Se o *thread* estiver em *push mode*[3], ele constrói um *buffer*[4] com 2 (duas) entradas tiradas da visão parcial mais uma entrada que identifica ele próprio. Se o *thread* também estiver em *pull mode*[5], ele vai esperar por uma resposta do *peer* selecionado;**
- ***Thread* passivo: ao receber um buffer de um nó, o *thread* irá construir um buffer da mesma forma que o *thread* ativo, e irá enviá-lo como resposta.**

Ao receber o *buffer*, os dois *threads* constroem uma nova visão parcial com as entradas. Há dois modos de construir a visão parcial:

- **Os dois nós descartam as entradas que tinham enviado (trocam parte de suas visões);**
- **Os dois nós descartam o maior número possível de entradas antigas.**

Observações importantes surgem deste tipo de abordagem:

1) **Protocolos que usam somente *pull mode* ou somente *push mode* podem gerar redes de sobreposição desconectadas;**

2 *Threads*: formas de um processo dividir a si mesmo em duas ou mais tarefas que podem ser executadas concorrentemente.
3 *Push mode*: um *peer* envia informação aos outros sem eles pedirem.
4 *Buffer:* dispositivo de armazenamento de dados temporários.
5 *Pull mode*: um *peer* pergunta aos outros se eles tem novas informações

2) **Um nó pode sair da rede sem informar qualquer outro nó, desde que os nós troquem visões parciais regularmente. Como as entradas antigas são descartadas, as entradas referentes a nós que saíram da rede serão removidas automaticamente.**

3) Adotar a estratégia descrita em 2) pode gerar outro problema. Dado que o grau interno de um nó é o número de nós cujas visões parciais referenciam este nó, quanto maior o grau interno de um nó, maior é a probabilidade de que alguém decida contatar este nó. O descarte sistemático de entradas antigas aumenta o grau interno de alguns nós (TANENBAUM, 2006).

Este tipo de rede P2P não-estruturada pode apresentar o problema de escalabilidade: aumento da dificuldade em localizar itens de dados à medida que a rede cresce.

Uma alternativa para este problema é o modelo de *superpeers*. Este modelo de rede delega funções especiais a alguns *peers* da rede (chamados de *superpeers*), tornando-os servidores para outros *peers.* Apenas os s*uperpeers* têm acesso efetivo a rede P2P, ou seja, somente eles possuem conexão e operação direta com outros *superpeers*. Os demais *peers* (convencionais) apenas requisitam e respondem aos respectivos *superpeers*. De acordo com Tanenbaum (2006), todo *peer* está conectado a um *superpeer*, e toda comunicação do *peer* é realizada através do *superpeer*.

Outra característica deste modelo é que ao entrar na rede e conforme o critério utilizado para ordená-los (geralmente o IP da estação) os *peers* são reagrupados, de maneira que fiquem ordenados sequencialmente. A Figura 10 ilustra o modelo de tal rede, onde pode ser visualizado *peers* convencionais, ligados a *superpeers*, como se fossem pequenas comunidades. Entre estas comunidades existe uma ligação, garantindo, desta maneira, a comunicação total da rede.

Este modelo de rede P2P agrega vantagens dos modelos de localização centralizada e baseado em inundação, pois mantém sempre uma tabela atualizada nos *superpeers* com o conteúdo disponível na rede, de forma a agilizar as buscas. Entretanto, pode ocorrer a sobrecarga dos *superpeers*, uma vez que estes efetuam maior número de operações. A Figura 10 exemplifica este tipo de modelo de rede P2P.

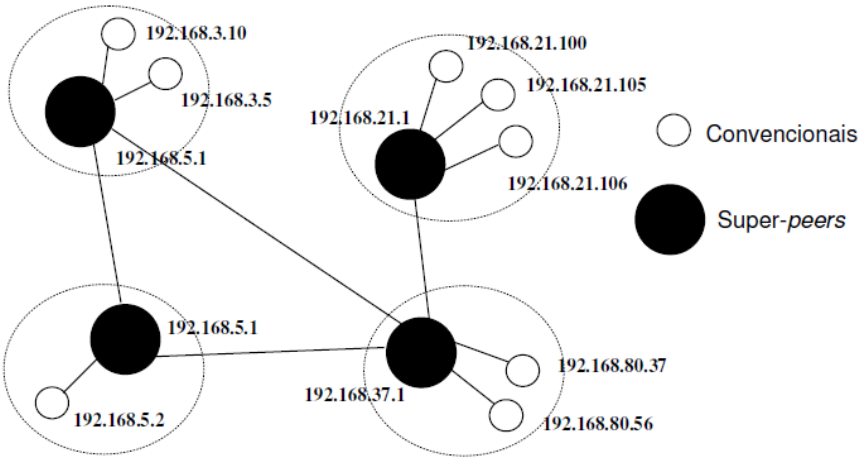

Figura 1: Exemplo de uma rede baseada em redes de superposição

Sendo assim, o primeiro objetivo específico da pesquisa - Revisar os protocolos HTTP, DNS, IRC e P2P - foi cumprido neste capítulo, onde foi realizada uma revisão sobre estes protocolos, o que proporcionará o entendimento pleno do funcionamento das técnicas de detecção de *botnets* fundamentadas nestes protocolos.

3 BOTNETS

Faz-se necessário conhecer alguns conceitos iniciais sobre *botnets*, dentre os quais destacam-se: definições, componentes, tipos, objetivos e ciclo de vida.

3.1. Definições

Segundo Sacchetin (2008), *botnets* são redes de computadores que foram infectadas e estão sob o comando do controlador da rede.

De acordo com Zhu (2008), *botnet* é uma coleção de *softwares* robôs que funcionam em computadores *host* [6],de forma autônoma e automaticamente, controlados remotamente por um ou vários invasores.

O nome "*bot*" significa robô e "*net*" é a abreviação de network, ou seja, é um robô que executará suas ações na rede por meio de comandos enviados remotamente pelo controlador da rede.

3.2. Componentes das *botnets*

- *Bot* – *Software* instalado na máquina da vítima capaz de realizar um conjunto de ações, normalmente maliciosas.

6 Host: máquina ou computador conectado a uma rede que hospeda alguma aplicação.

- Máquina vítima – Máquina infectada pelo *bot*;
- *Botmaster* – Usuário que detém o controle da rede.
- Centro de comando e controle – Meio pelo qual o *botmaster* envia os comandos para os *bots*.

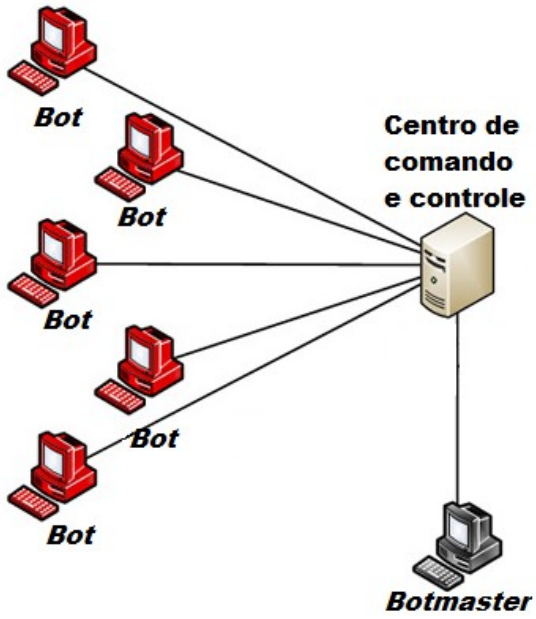

Figura 1: Componentes de uma *botnet*

Assim, *botnet* é uma rede de *bots* gerenciada pelo *botmaster* através do centro de comando e controle, como se verifica na Figura 11.

3.3. Tipos de *botnets*

Neste trabalho, serão abordados dois tipos de *botnets*, descritos segundo ENISA (2011):

- *Botnet* centralizada: Segue o modelo de rede cliente-servidor e é o tipo de *botnet* mais utilizado. Todos os *bots* estabelecem um canal de comunicação com um ponto de conexão, como ilustrado na Figura 12. Este ponto de conexão é o centro de comando e controle, sob o controle do *botmaster*.

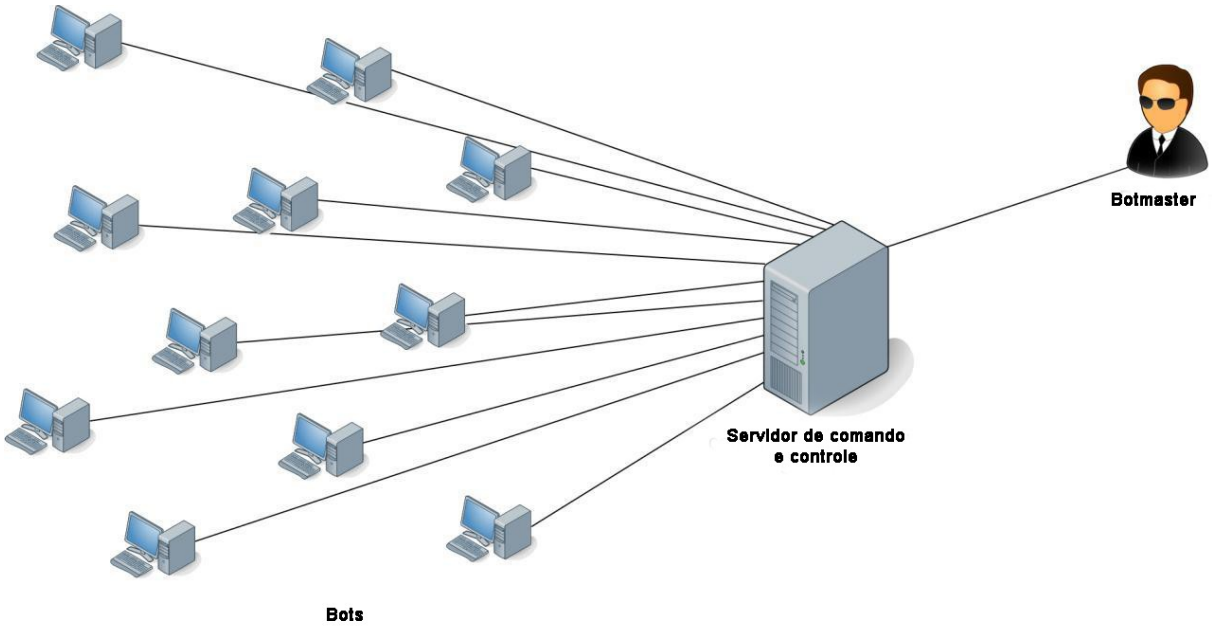

Figura 2: *Botnet* centralizada
Fonte: ENISA (2011)

- *Botnet* descentralizada: Segue o modelo de rede *peer-to-peer* (P2P). Não existe um servidor central que centraliza o controle da *botnet*, como ilustrado na Figura 13.

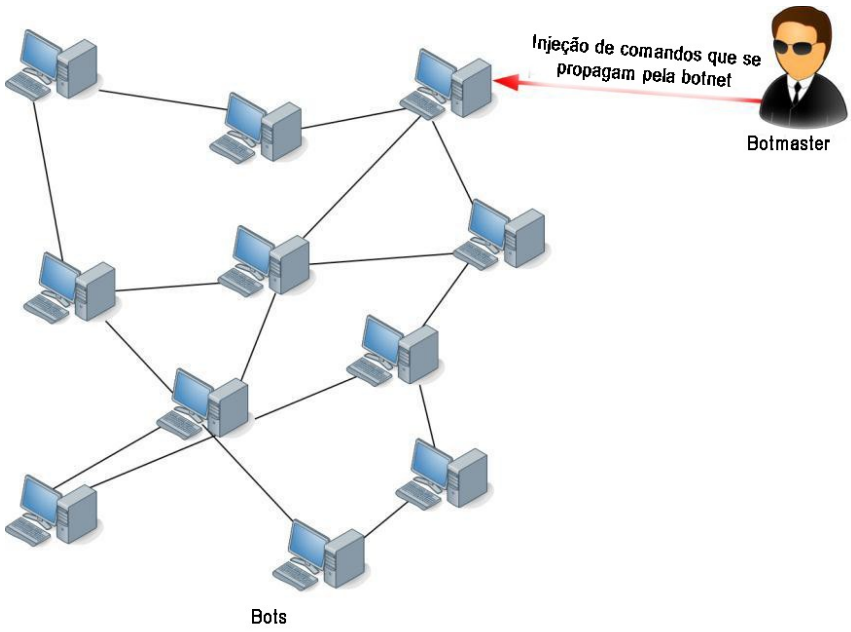

Figura 3: *Botnet* descentralizada
Fonte: ENISA (2011)

3.4. Objetivos das *botnets*

*Botnet*s em geral têm os seguintes objetivos:

- Coleta de informações: Alguns *bots* têm habilidade para capturar teclas digitadas no teclado, pontos da tela ao serem clicados pelo botão do mouse, arquivos, tráfego de rede e dados armazenados. Esta habilidade pode ser utilizada para capturar informações sobre cartões bancários, estratégias comerciais de empresas de grande vulto, documentos de alto nível de sigilo, entre outros;
- Ataque distribuído de negação de serviço (DDoS): *Botnets* são usadas para lançamento de ataques de negação de serviços que podem parar um serviço ou um servidor alvo;
- Encaminhamento de *spam*: *e-mails* não solicitados (*spam*) podem ser efetivamente distribuídos com maior cobertura através das *botnets*;
- Repositório *malware*[7]: os controladores de *botnets* precisam de recursos para manter as ferramentas disponíveis. Para este fim, algumas máquinas são usadas como repositórios (*bots* clientes). Colocar estas ferramentas em várias máquinas garante disponibilidade à *botnet*;
- Conteúdo ilegal: *botnets* podem ser utilizadas para armazenamento ilegal de conteúdo como, por exemplo, arquivos, documentos, números de cartões de crédito e pornografia;
- Anonimato: quando várias máquinas em todo o mundo são utilizadas como pontes para acessar um host comprometido, é muito difícil realizar um rastreamento e identificar o invasor real ou *botmaster.* (SACCHETIN, 2008).

7*Malware*: proveniente do inglês ***mal****icious soft****ware***; é um *software* destinado a se infiltrar em um sistema de computador alheio de forma ilícita, com o intuito de causar algum dano ou roubo de informações.

3.5. Ciclo de vida das *botnets*

O ciclo de vida das *botnets* segue 5 (cinco) fases, como se verifica na Figura 14:

1) Busca/Infecção inicial: Através de varreduras de redes detectam-se vítimas em potencial. Estas máquinas-vítimas podem ser infectadas através de: emprego de *exploits*[8], *download* de *malwares*, anexos de correio eletrônico, cavalos de tróia, entre outros;
2) Infecção secundária: A máquina infectada busca contato com o repositório de binários. A partir desta etapa, a máquina-vítima torna-se um verdadeiro *bot*;
3) Conexão ou *Rally*: O *bot* realiza contato com o centro de comando e controle da *botnet* e está pronto para realizar atividades maliciosas;
4) Atividade maliciosa: O *bot* realiza alguma atividade maliciosa a comando do *botmaster*;
5) Manutenção e atualização: São realizadas atividades de manutenção e atualização da *botnet*, principalmente vinculadas à segurança e ao funcionamento da rede, como por exemplo: atualização do código binário dos *bots*, mudança de servidor de comando e controle, mudança de protocolos, entre outras.

8*Exploit*: programa de computador, uma porção de dados ou uma sequência de comandos que se aproveita das vulnerabilidades de um sistema computacional.

Figura 1: Ciclo de vida das *botnets*

Sendo assim, o segundo objetivo específico da pesquisa - Estudar as *botnets*: definição, componentes, tipos, objetivos e ciclo de vida – foi cumprido neste capítulo, onde foi desenvolvido um estudo sobre estes aspectos conceituais relacionados às *botnets.* Este estudo viabilizará a compreensão da relação entre as técnicas de detecção e os componentes, tipos, objetivos e ciclo de vida das *botnets*.

4 DETECÇÃO DE *BOTNETS*

As técnicas de detecção de *botnets* são divididas em duas abordagens, de acordo com o contexto em que se enquadram:

- Análise de hospedeiro (*host*): ocorre em um contexto isolado, onde é feita a análise da máquina separadamente, com técnicas que buscam detectar se este hospedeiro participa de uma *botnet*.
- Análise de rede (*network*): ocorre em um contexto generalizado, onde é feita a análise como um todo do tráfego da rede. É realizada a partir de dois tipos de técnicas:
    - Técnicas passivas: Segundo ENISA (2011), este grupo é constituído por técnicas em que os dados são analisados somente através da observação. Com o monitoramento, as atividades maliciosas podem ser rastreadas sem interferir no ambiente da rede. Sendo assim, a utilização destas técnicas é de difícil percepção por parte dos *botmasters*, entretanto elas podem limitar a obtenção de dados recolhidos para a análise.
    - Técnicas ativas: Este grupo de técnicas de detecção contém abordagens que envolvem interação com a rede monitorada. Assim, esta interferência direta na dinâmica da rede possibilita uma obtenção de dados mais completa e consistente. Entretanto, a sua atuação pode ser percebida pelo *botmaster*, podendo acarretar reações como: ataques DDoS contra os elementos de monitoramento, mudanças na estrutura da *botnet* que visam dificultar o monitoramento, entre outras. (ENISA, 2011).

Neste trabalho, focar-se-á nas técnicas de detecção enquadradas na abordagem da análise de rede (*network*), tendo em vista que estas técnicas são as mais utilizadas.

4.1. Análise de rede (network)

4.1.1. Técnicas passivas

Como passivas, serão abordadas as seguintes técnicas: Inspeção de pacotes, Análise de registros de fluxo, Análise de log e *Honeypots*.

4.1.1.1. *Inspeção de pacotes*

Esta técnica consiste basicamente na inspeção dos dados de pacotes de rede. Segundo ENISA (2011), a idéia fundamental é fiscalizar os campos de protocolo e o *payload* (carga útil) de um pacote com o intuito de identificar padrões pré-definidos de conteúdo anormal ou suspeito. Estes padrões são também chamados de assinaturas de detecção.

Esta abordagem apresenta baixo rendimento em redes com alta carga de tráfego uma vez que ela necessita realizar a inspeção de todo ou grande parte do conteúdo de todos os pacotes.

Exemplos de aplicação da técnica de inspeção de pacotes:

- Uma aplicação desta técnica são os sistemas de detecção de intrusão (IDS) que rastreiam assinaturas de detecção. O propósito destes IDSs é proteger o ambiente no qual ele é implantado e emitir um aviso se um ataque é reconhecido. Um IDS pode realizar ações adicionais além da emissão do alerta de identificação de um ataque, como por exemplo:
    - Rejeitar o pacote suspeito ou fechar a conexão relacionada.
    - Transmitir o conteúdo do pacote suspeito a um sistema de análise.
    - Usar informações extraídas sobre a ataque para contribuir com a atualização de listas negras ou de regras de *firewall*[9].
- Segundo Binkley (2006), em uma análise prática de desempenho de um algoritmo de detecção de *botnets* baseadas em IRC a partir da técnica de inspeção de pacotes aplicada a rede de um campus, verificou-se que o uso

9*Firewall:* dispositivo de uma rede de computadores que tem por objetivo aplicar uma política de segurança a um determinado ponto da rede.

de medidas de evasão, como a criptografia e a não codificação padrão do protocolo de comunicação, impedem a detecção de pacotes maliciosos.

Com relação à avaliação da técnica de inspeção de pacotes tem-se:

1) Complexidade: Esta técnica possui um baixo nível de complexidade uma vez que se baseia na análise dos dados de um pacote, não sendo necessários infra-estrutura de rede e equipamentos especiais.
2) Evasão do *botmaster*: A evasão do *botmaster* desta técnica é muito simples pois ele pode evitar a detecção através de medidas, como por exemplo:
    - Uso de criptografia, impossibilitando a análise direta dos dados dos pacotes de rede.
    - Divisão dos pacotes para a transmissão de tal forma que as assinaturas de detecção em sua carga útil fiquem desmembradas em pacotes distintos e impossibilitem a sua identificação através desta técnica.
3) Falso-positivo: Esta técnica tende a possuir uma taxa de falso-positivo relativamente baixa uma vez que pacotes benignos geralmente possuem sua assinatura diferente das assinaturas de detecção que caracterizam pacotes maliciosos. Entretanto, a adoção de critérios rigorosos de filtragem pode levar à classificação de pacotes benignos como maliciosos.
4) Falso-negativo: A inspeção de pacotes apresenta uma alta taxa de falso-negativo. Conforme descrito no experimento de Binkley (2006), o uso de medidas de evasão, como a criptografia e a não codificação padrão do protocolo de comunicação, impedem a detecção de pacotes maliciosos.

4.1.1.2. *Análise de registros de fluxo*

Esta técnica se baseia na inspeção de pacotes de rede em um nível mais abstrato. Em vez de inspecionar os pacotes individualmente, como acontece na técnica de inspeção de pacotes, os fluxos de comunicação são considerados de forma agregada. Segundo ENISA (2011), um registro de fluxo consiste em várias propriedades que descrevem os dados obtidos a partir de um conjunto de pacotes de rede. Atributos típicos são: a origem e o destino, os números de porta

relacionados, o protocolo utilizado, a duração da sessão, o tamanho cumulativo e o número de pacotes transmitidos. Como a verificação da carga útil dos pacotes é ignorada nesta abordagem, pode-se analisar uma maior carga de tráfego do que com a técnica de inspeção de pacotes. Uma estratégia para lidar com redes de alta taxa de tráfego é trabalhar com amostragem, ou seja, analisar apenas uma parcela dos pacotes que trafegam na rede. Taxas de amostragem comumente usadas são 0,1-1%, o que pode gerar perda de informação ou obtenção de conclusões distorcidas.

O objetivo da análise dos registros de fluxo é identificar padrões de tráfego na rede com o intuito de distinguir o tráfego benigno do tráfego malicioso e criar um esquema para detecção de potenciais comunicações maliciosas. Ela permite, por exemplo, a identificação dos *hosts* que interagem com servidores de comando e controle de *botnets* conhecidos, como por exemplo, de acordo com Higgins (2007):

- Storm, *botnet* P2P com, em média, 230.000 membros por dia.
- Rbot, *botnet* IRC com, em média, 40.000 membros por dia.
- Bobax, *botnet* HTTP com, em média, 24.000 membros por dia.

Exemplo de aplicação da técnica de análise de registros de fluxo:

- Segundo Yen (2008), com o uso da ferramenta TAMD (Agregação de Tráfego de Detecção de Malware), foi possível detectar 87,5% das *bots* de uma *botnet* inserida em uma rede de universidades com mais de 33.000 endereços IP. Esta ferramenta realiza a correspondência de padrões sobre os fluxos de rede. Três características são consideradas: os fluxos que se comunicam com um destino comum, os fluxos com tamanho de carga semelhante e os fluxos que pertencem aos *hosts* com um mesmo sistema operacional.

Com relação à avaliação da técnica de análise de registros de fluxo tem-se:

1) Complexidade: Esta técnica possui um baixo nível de complexidade uma vez que se baseia na análise de um conjunto de pacotes, não sendo necessários infra-estrutura de rede e equipamentos especiais.
2) Evasão do *botmaster*: A evasão do *botmaster* desta técnica é relativamente complexa pois como são analisados os campos de cabeçalho dos pacotes e

não a carga útil, o uso de criptografia não influencia na evasão do *botmaster*. Assim, as táticas de evasão utilizadas empregam a variação periódica dos dados geralmente analisados nos registros de fluxo (a origem e o destino, os números de porta relacionados, o protocolo utilizado, a duração da sessão, o tamanho cumulativo e o número de pacotes transmitidos) com o intuito de dificultar a caracterização de padrões de atuação.

3) Falso-positivo: Esta técnica tende a possuir uma taxa de falso-positivo relativamente baixa uma vez que pacotes benignos geralmente possuem seu conjunto de atributos diferente do conjunto de atributos que caracterizam pacotes maliciosos. Entretanto, a adoção de critérios rigorosos de análise pode gerar pacotes falsamente suspeitos.
4) Falso-negativo: A inspeção de pacotes apresenta uma baixa taxa de falso-negativo. Conforme o estudo de Yen (2008), com o uso da ferramenta TAMD (Agregação de Tráfego de Detecção de Malware), foi possível detectar 87,5% das *bots* de uma *botnet* inserida em uma rede de universidades com mais de 33.000 endereços IP.

4.1.1.3. Análise de arquivos de log

Segundo Fonseca (2005), os arquivos de log são arquivos onde são armazenados registros sobre data, hora, *host* e mensagem emitida de aplicações processadas em um sistema computacional. Os logs podem ser configurados para registrar desde somente os eventos críticos até praticamente todos os eventos do sistema. É nos logs que se encontram informações sobre o funcionamento dos programas servindo como ferramenta para a correção de erros e verificação de rotinas. Logs são muito importantes para a administração segura de sistemas, pois registram informações sobre o seu funcionamento e sobre eventos por eles detectados. Muitas vezes, os logs são o único recurso que um administrador possui para descobrir as causas de um problema ou comportamento anômalo. No contexto da detecção de *botnets*, de acordo com ENISA (2011), nos arquivos de log estão armazenados os registros dos dados recebidos através de uma rede e as respostas correspondentes. Sendo assim, a análise de arquivos de log consiste em uma

técnica de detecção que busca rastrear atividades de *bots* através da inspeção destes registros.

Exemplos de aplicação da técnica de análise de arquivos de log:

- De acordo com Leder (2009), a botnet do Conficker usa solicitações HTTP a *websites* de grande fluxo de acesso (como Yahoo.com, CNN.com, entre outros) para sincronizar os relógios internos dos *bots* e verificar a conectividade da Internet. À primeira vista, entradas de log criadas através deste comportamento serão semelhantes a solicitações HTTP comuns, que poderiam originar de um usuário benigno interagindo com o *website*. Isto ilustra a dificuldade na diferenciação do comportamento entre solicitações comuns e maliciosas. No caso da *botnet* Conficker, uma estratégia para rastrear os seus *bots* é verificar acessos que apenas consultaram a página inicial, mas não o conteúdo vinculado a esta página. Além disso, se um mecanismo idêntico é utilizado por dois ou mais *botnets* distintos, apenas a presença do *bot* pode ser detectada, não o *botnet* que está associado.
- Linari (2010) aborda o exemplo dos ataques DDoS (ataques distribuídos de negação de serviço) a um servidor *web*, onde a análise de arquivos de log pode render uma lista dos endereços IP participantes. Em alguns casos, não é possível afirmar quais destes endereços estão atrelados a uma atividade maliciosa. Assim, esta técnica só pode ser aplicada quando características de comportamento malicioso podem ser claramente identificadas. Tentativas tem sido feitas para identificar a caracterização de logs de eventos anormais e anomalias, tais como o utilização atípica de um serviço ou sequências de pedidos e consultas sem objetivo definido. Por exemplo, um aumento significativo de solicitações de um serviço de rede pode expor atividades de uma *botnet*.

Com relação à avaliação da técnica de análise de arquivos de log tem-se:

1) Complexidade: Esta técnica possui um baixo nível de complexidade uma vez que se baseia basicamente na análise dos registros dos arquivos de log armazenados nos elementos da rede, não sendo necessários infra-estrutura e equipamentos especiais.

2) Evasão do *botmaster*: A evasão do *botmaster* desta técnica é relativamente complexa pois como a criação de logs é uma atividade vinculada a máquina vítima e não à atividade maliciosa, o botmaster não tem interferência direta sobre a criação ou não destes registros. Assim, as táticas de evasão utilizadas empregam duas abordagens:
   - A variação periódica dos dados geralmente analisados nos arquivos de log (a origem e o destino, os números de porta relacionados, o protocolo utilizado, entre outros) com o intuito de dificultar a caracterização de padrões de atuação.
   - A invasão das máquinas vítimas com o intuito de apagar registros de log que possibilitam o rastreamento das atividades de uma *botnet*.
3) Falso-positivo: Esta técnica tende a possuir uma taxa de falso-positivo relativamente baixa uma vez que registros de *softwares* benignos geralmente possuem conteúdo distinto em relação aos registros que caracterizam *software* maliciosos. Entretanto, esta classificação de *softwares* maliciosos só é possível em casos onde a identificação de padrões dos registros é claramente verificada.
4) Falso-negativo: A inspeção de pacotes apresenta uma alta taxa de falso-negativo. Conforme verificado nos exemplos de Leder (2009) e Linari (2010), pode ser difícil a diferenciação de atividades maliciosas das atividades normais considerando apenas a análise de arquivos de log.

4.1.1.4. Honeypots

Honeypot (pote de mel, na tradução literal) é um recurso vulnerável intencionalmente implantado em uma rede que tem por objetivo ser atacado e comprometido por uma entidade maliciosa. Segundo ENISA (2011), a principal razão para pesquisa e desenvolvimento de *honeypots* é obter informações sobre as práticas e estratégias utilizadas pelos criadores de *malware*. Em geral, dois tipos de informações podem ser obtidas através de *honeypots*:

- Tipos de vetor utilizados em ataques a sistemas operacionais e softwares, bem como os códigos correspondentes.
- Ações executadas em uma máquina vítima.

Conforme Cui (2006), um *honeypot* só pode analisar o tráfego de entrada do endereço IP atribuído a ele. Isto significa que conclusões acerca de tamanho e atividade de *botnets* só podem ser obtidas por meio da agregação de *honeypots* em uma *honeyfarm*. Neste contexto de detecção de *botnets*, *honeypots* podem ser utilizados para:

- Analisar ataques provenientes de *botnets*, proporcionando a ratificação de padrões de atuação de *botnets* conhecidas ou a identificação de padrões de atuação de *botnets* novas.
- Detectar eventos e surtos de malware
- Identificar e enumerar *bots*

De acordo com Spitzer (2002), *honeypots* podem ser caracterizados pelo seu grau de interação com o atacante:

- *Honeypots* de alta interatividade: Permitem todas as ações do atacante sobre o sistema monitorado. Muitas vezes, mais de um *honeypot* de alta interatividade é implantado simultaneamente com o objetivo de abranger sistemas operacionais diferentes ou uma variedade de serviços distintos. Pode-se ainda simular uma sub-rede, que é referida como uma *honeynet*. Um ponto crítico dos *honeypots* de alta interatividade é a vulnerabilidade uma vez que a exposição do sistema operacional completo pode proporcionar o alcance de estados inseguros. Sendo assim, é comum a utilização de elementos de filtragem no tráfego proveniente de *honeypots*, conforme se verifica na Figura 15.
- *Honeypots* de baixa interatividade: Apenas simulam serviços de rede ou funcionalidades de um sistema operacional. Como desvantagem pode-se citar a limitação deste tipo de abordagem uma vez que apenas parte de um sistema é simulada, proporcionando uma visão limitada das técnicas do atacante. Como vantagem pode-se indicar a estabilidade pois a exposição de apenas um conjunto de serviços ao invés do sistema operacional completo garante que não há a necessidade de restauração de estados inseguros. Isso ajuda na manutenção do *honeypot* e na escalabilidade, na medida em que múltiplas vulnerabilidades podem ser emuladas em paralelo.

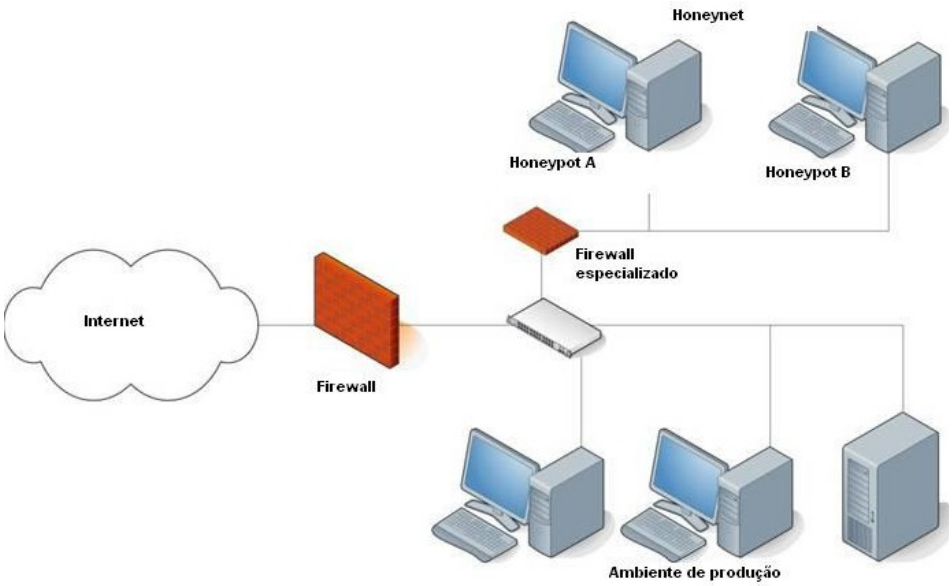

Figura 4: *Honeypot* de alta interatividade
Fonte: ENISA (2011)

Exemplos de aplicação da técnica de *honeypots*:

- Goebel (2007) implantou o *software Nepenthes honeypot* em uma universidade para observar 16.000 endereços IP por oito semanas, entre dezembro de 2006 e janeiro de 2007. Durante este período, foram reunidos 13400000 binários de *malwares*. Através dessas amostras, eles identificaram um total de 40 servidores de Comando e Controle.
- Small (2008) apresentou uma *honeypot* HTTP com o capacidade de criar respostas dinâmicas para consultas maliciosas. Seu principal objetivo era a detecção de *bots* que empregavam algoritmos de busca para encontrar *hosts* vulneráveis. Durante os 72 dias de análise, foram reconhecidos mais de 386 mil ataques contra o *honeypot*. Uma análise dos URLs incluídos nos ataques coletados mostrou 5.648 repositórios de scripts distintos. Estes dados fornecem uma estimativa do número de *bots* usados para os referidos ataques.

Com relação à avaliação da técnica de *honeypots* tem-se:

1) Complexidade: Esta técnica possui um nível de complexidade médio uma vez que são necessários infra-estrutura e recursos de rede, apesar destes recursos não exigirem tratamento complexo tendo em vista que estes são implantados propositalmente vulneráveis com o objetivo de se apresentarem como alvos potenciais para as atividades maliciosas.
2) Evasão do *botmaster*: A evasão do *botmaster* desta técnica é difícil pois a configuração de um *honeypot* é geralmente igual a de uma vítima comum. Assim sendo, torna-se complexa a tarefa de identificar um *honeypot* e se evadir deste.
3) Falso-positivo: Esta técnica tende a possuir uma taxa de falso-positivo relativamente baixa uma vez que o tráfego de entrada benigno geralmente possui características distintas de um ataque. Entretanto, uma análise pontual e rigorosa pode classificar uma fluxo de entrada benigno falsamente como um ataque.
4) Falso-negativo: Os *honeypots* apresentam uma baixa taxa de falso-negativo. Conforme verificado nos exemplos de Goebel (2007) e Small (2008), esta técnica possibilita a obtenção de uma análise consistente do tráfego de entrada no *honeypot*, sendo possível a verificação dos detalhes de execução destas entradas. Assim, torna-se possível a identificação dos ataques realizados a uma *honeypot*.

4.1.2. Técnicas ativas

Como ativas, serão abordadas as seguintes técnicas: *Sinkholing*, *Snooping cache* DNS, Rastreamento de redes *fast-flux*, Medição IRC, Enumeração P2P e Infiltração.

4.1.2.1. Sinkholing

Sinkholing é uma técnica de detecção de *botnets* baseada no protocolo DNS.

Segundo ENISA (2011), quando um hospedeiro participa de uma *botnet*, deve ser estabelecida a comunicação entre ele e o servidor de comando e controle ou

outros hospedeiros infectados, dependendo do tipo de *botnet*. Há duas formas de estabelecer este contato:

- Por endereços IP fixos.
- Pela resolução de nomes de domínio proporcionada pelo protocolo DNS, onde um nome de domínio é resolvido em um endereço IP, conforme explicado na revisão sobre o protocolo DNS, no capítulo 2 deste trabalho.

O estabelecimento de contato por meio da resolução de nomes de domínio oferece maior flexibilidade em vários aspectos para as *botnets*. Dentre eles, destacam-se:

- Um nome de domínio pode estar atrelado a vários endereços IP, o que possibilita a estruturação de um sistema redundante, robusto e com maior grau de adaptabilidade.
- É possível atualizar a lista de endereços IP atrelados a um nome de domínio. Assim, esta resolução de nomes pode ser alterada dinamicamente.
- A remoção de nomes de domínio maliciosos, muitas vezes, esbarra na burocracia e nos esforços administrativos necessários para este cancelamento de registro junto aos provedores de serviço de *Internet*.

Sendo assim, a detecção de *botnets*, vinculada ao protocolo DNS, está diretamente relacionada ao monitoramento de nomes de domínio maliciosos. Neste contexto, aborda-se a técnica *sinkholing*. O princípio básico desta técnica, conforme se verifica na Figura 16, consiste em alterar os endereços IP dos nomes de domínio maliciosos existentes em servidores DNS, redirecionando esta comunicação para servidores de entidades confiáveis, como por exemplo, servidores de centros de pesquisa no assunto. Desta forma, torna-se possível a obtenção de dados sobre a dinâmica de funcionamento de uma *botnet* vinculada a este nome de domínio malicioso que está sendo monitorado.

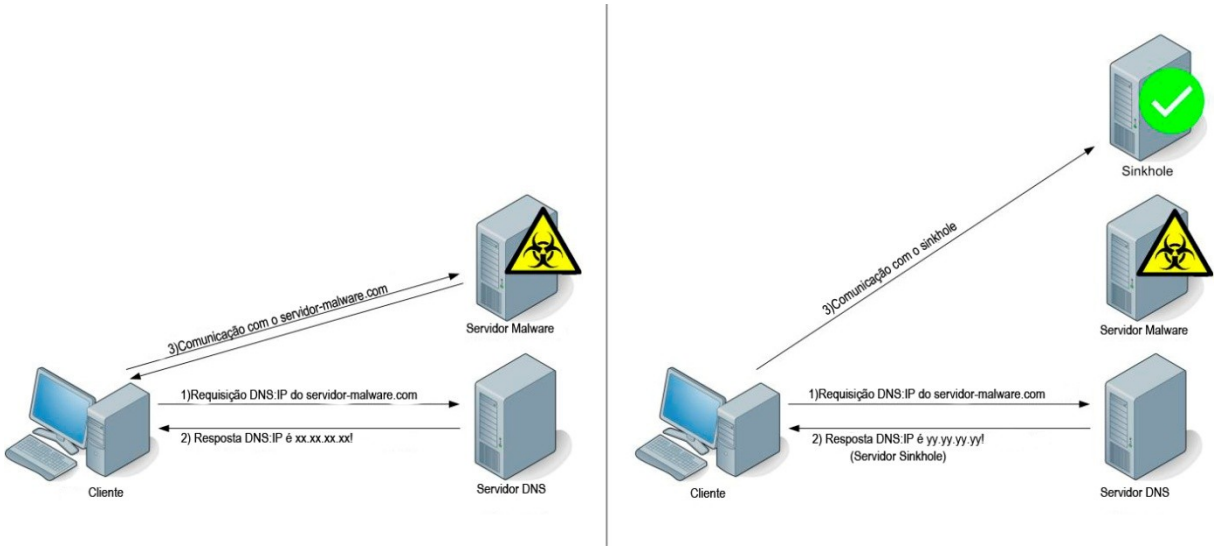

Figura 5: Sinkholing
Fonte: ENISA (2011)

Exemplos de aplicação da técnica de *sinkholing*:

- Choi (2007) apresentou uma abordagem de detecção de anomalias baseada em servidores de comando e controle. Ele observou que botnets tendem a exibir um comportamento coordenado. Por exemplo, cada vez que um servidor C&C tem uma falha de ligação ou é migrado para um novo nome de domínio malicioso, os *bots* da *botnet*, quase que simultaneamente, executam consultas ao servidor C&C com os dados antigos. Além disso, ele verificou que o uso de serviços de DNS dinâmico (DDNS), pode servir como um indicador de um servidor de comando e controle.
- Villamarin-Salomon (2008) também apresentou uma anomalia baseada na detecção de servidores C&C. Em seu trabalho, eles concluíram que quantidades anormais de respostas NXDOMAIN recorrentes são um indicador para uma *botnet*. A resposta NX-DOMAIN é gerada por um servidor DNS se um nome de domínio não pode ser resolvido. No contexto das *botnets*, este é frequentemente o resultado de uma queda ou migração de um servidor de comando e controle.

Com relação à avaliação da técnica de *sinkholing* tem-se:

1) Complexidade: Esta técnica possui um nível de complexidade médio uma vez que são necessários infra-estrutura e recursos de rede, além da alteração de registros de recursos em servidores DNS.

2) Evasão do *botmaster*: A evasão do *botmaster* desta técnica é fácil pois a alteração do nome de domínio malicioso atrelado à *botnet* anula a atuação do *sinkhole*. Além disso, o monitoramento do *sinkhole* demonstra uma forte relação de dependência com as informações disponíveis para o hospedeiro de destino, isto é, o monitoramento pode ter uma pequena taxa de obtenção de dados, caso a quantidade de informação útil extraível seja reduzida pelo uso de criptografia ou outros recursos semelhantes.

3) Falso-positivo: Esta técnica apresenta uma baixa taxa de falso positivo, uma vez que nomes de domínio e endereços IP atrelados a *botnets* tem, geralmente, propósitos puramente maliciosos.

4) Falso-negativo: Esta técnica tende a possuir uma taxa de falso-negativo muito baixa uma vez que ela monitora meios maliciosos por definição, considerando que geralmente só os *bots* resolvem o nome de domínio malicioso e acessam o *sinkhole*.

4.1.2.2. Snooping cache DNS

A técnica de detecção *Snooping cache* DNS é baseada na propriedade de *cache* implementada por muitos servidores DNS. Conforme verificado na revisão sobre o protocolo DNS, no capítulo 2 do presente trabalho, se um servidor DNS é consultado para um nome de domínio que ele não é capaz de resolver, ele emite uma consulta para o servidor DNS autoritário responsável, para que a resolução de nomes seja concluída. Ao fim do processo de resolução de nomes, os registros obtidos ficam armazenados em um *cache* local. Assim, a utilização de *caching* busca principalmente aumentar o desempenho de um servidor de nomes e diminuir sua carga de tráfego (KUROSE, 2006).

Segundo ENISA (2011), o *caching* também pode ser usado para fins de medição, como na técnica de *Snooping cache* DNS. Conforme se verifica na Figura

17, a idéia central é verificar indiretamente se um nome de domínio malicioso foi consultado através de um servidor DNS específico, testando se uma resposta é armazenada em *cache*.

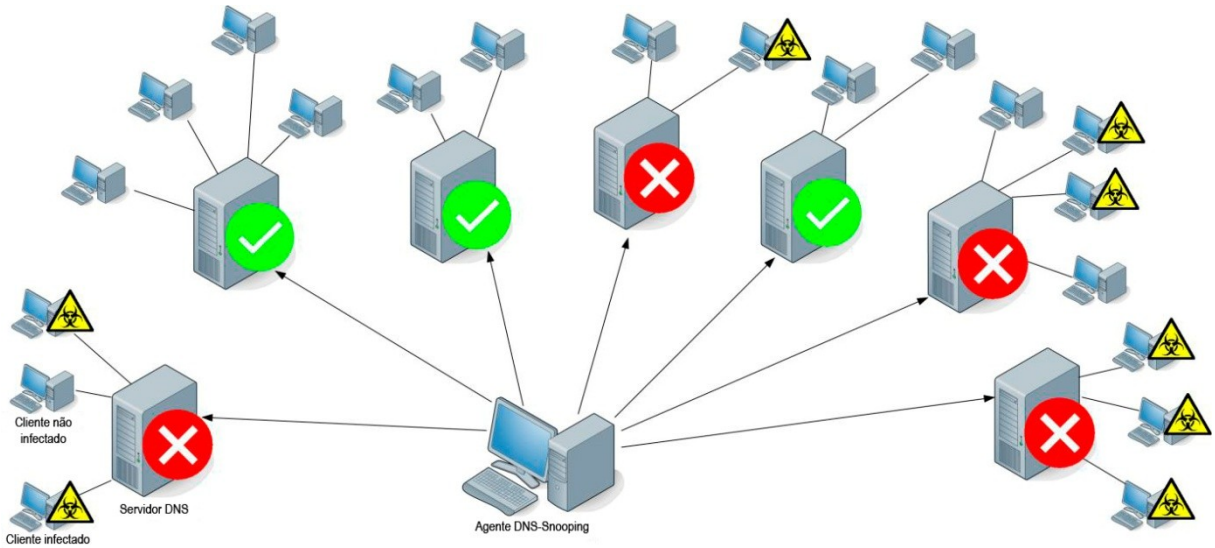

Figura 6: *Snooping cache* DNS
Fonte: ENISA (2011)

A implementação da idéia central desta técnica baseia-se no monitoramento periódico de servidores DNS. São emitidas 2 (duas) consultas consecutivas periodicamente para estes servidores: uma normal e uma não-recursiva. No processamento da primeira consulta, o servidor realiza normalmente a resolução de nomes através do serviço DNS, armazenando em cache a resolução obtida. Ao ser realizada a segunda consulta (não-recursiva), a resolução será feita mediante acesso ao cache. Ao se comparar o tempo de armazenamento em cache da resolução com o intervalo entre consultas, é possível descobrir se esta resolução foi requisitada por outra consulta neste período entre consultas (o que configura um acerto de cache). Ao se realizar, periodicamente, consultas do agente *DNS-snooping* aos servidores DNS monitorados para a resolução de nomes de domínio suspeitos ou maliciosos, é possível a identificação de servidores que apresentam uma alta taxa de acertos, ou seja, servidores que resolvem nomes de domínio maliciosos, frequentemente, em nome de seus usuários. Sendo assim, este fato pode trazer informações importantes no que se refere à detecção de *botnets*, uma vez que *bots* podem ser identificadas através de seus servidores DNS.

Exemplo de aplicação da técnica de *snooping cache* DNS:

- Rajab (2008) realizou uma análise sobre a técnica de *snooping cache*. Primeiramente, ele criou um modelo para a análise do comportamento normal de servidores DNS em resposta à consultas por máquinas arbitrárias, a fim de extrapolar acessos à *cache*. Com isso, ele obteve uma forma de estimar a presença de entradas de nomes de domínio em *cache* através da medição do tempo necessário para a resolução DNS. Para os experimentos seguintes, ele utilizou uma lista de 768.000 resolvedores DNS para monitorar os nomes de domínio em *cache* relacionados com *botnets* e estimar o tamanho real destas redes maliciosas.

Com relação à avaliação da técnica de *snooping cache* DNS tem-se:

1) Complexidade: Esta técnica possui um nível de complexidade médio uma vez que são necessários infra-estrutura e recursos de rede.
2) Evasão do *botmaster*: A evasão do *botmaster* desta técnica é relativamente complexa pois o funcionamento da *cache* é uma característica particular de cada servidor DNS, não tendo o *botmaster* acesso direto a sua configuração.
3) Falso-positivo: Esta técnica apresenta uma baixa taxa de falso positivo, uma vez que nomes de domínio e endereços IP atrelados a *botnets* tem, geralmente, propósitos puramente maliciosos.
4) Falso-negativo: Esta técnica apresenta uma taxa de falso negativo relativamente baixa, uma vez que um *bot* para contatar um servidor de comando e controle terá que resolver um nome de domínio malicioso em um servidor DNS. Entretanto, esta técnica apresenta como restrição o número de nomes de domínio a serem monitorados, uma vez que um agente *DNS-snooping* pode estar monitorando vários servidores DNS, cada um com suas capacidades e limitações de resolução de nomes específicas.

4.1.2.3. Rastreamento de redes *fast-flux*

Algumas *botnets* são baseadas em redes *fast-flux*. O serviço de rede *fast-flux* é uma invenção dos desenvolvedores de botnets que busca aumentar o anonimato

destas redes de cunho malicioso. O princípio deste serviço se baseia no funcionamento do protocolo DNS. Como verificado na revisão deste protocolo no capítulo 2 do presente trabalho, um nome de domínio é resolvido em um endereço IP, pela estrutura de servidores DNS, para que um servidor de destino possa ser alcançado por um hospedeiro através da *Internet* (COMER, 2006).

O funcionamento básico de uma rede *fast-flux* ocorre como descrito na Figura 18. Segundo ENISA (2011), quando um nome de domínio malicioso tem que ser resolvido, uma consulta é enviada ao servidor DNS local, que, em seguida, repassa a consulta, através do sistema de DNS, a um servidor controlado por um *botmaster*. A resposta desta consulta estará atrelada a um grande número de endereços IP associados a *bots*. Estes *bots* exercem a função de servidores *proxy*, uma vez que eles realizam o encaminhamento ao usuário de informações obtidas em um servidor malicioso, que se esconde por trás desta camada *proxy*.

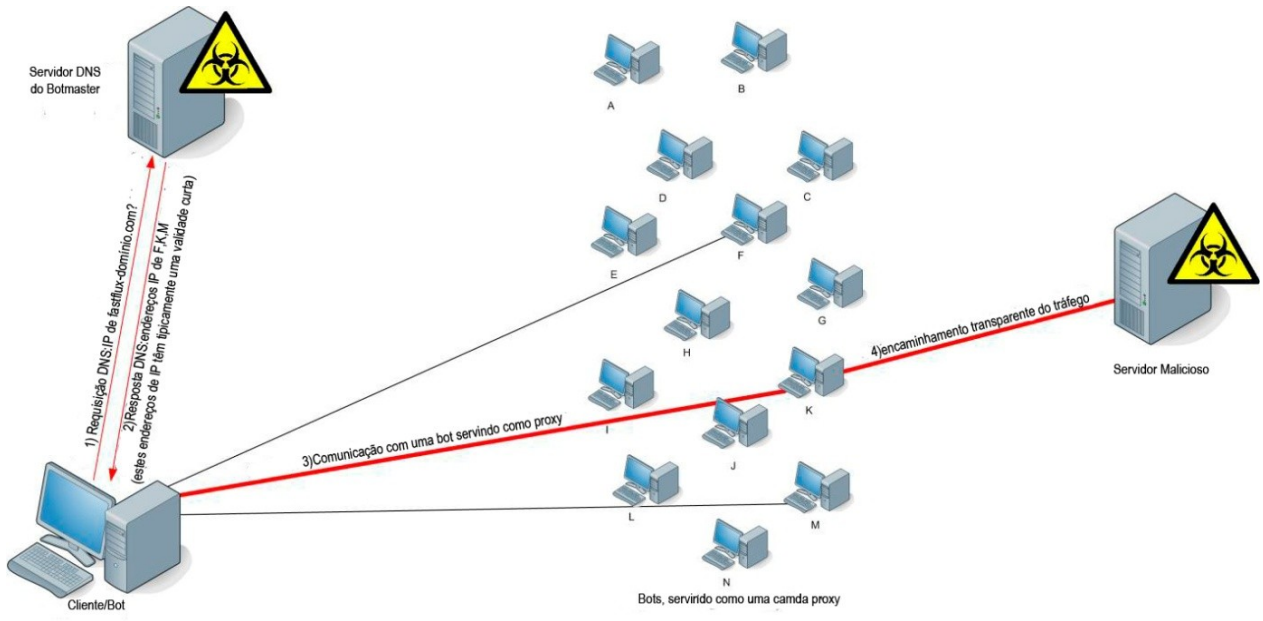

Figura 7: Redes *fast-flux*
Fonte: ENISA (2011)

O uso de redes *fast-flux* pelas *botnets* busca introduzir dissimulação de suas atividades e aumentar a confiabilidade de sua rede e estrutura de comando. Ao associar o grande volume de endereços IP dos bots da camada proxy com apenas alguns nomes de domínio maliciosos, a rede torna-se muito mais resistente contra contramedidas. Entretanto, algumas características deste tipo de rede geram padrões que podem torná-las rastreáveis.

Registros de domínios ligados à redes *fast-flux* geralmente tem uma validade curta de apenas alguns minutos. Esta validade é indicada pela informação *Time-To-Live* (TTL), valor incluído na resposta gerada pelo servidor DNS original no final da cadeia de consulta DNS, que está sob o controle do *botmaster*. Conforme se verifica na Figura 19, após este período de validade dos registros indicado pelo TTL, uma nova consulta normalmente irá resultar em um conjunto diferente de endereços IP associados com pouca ou nenhuma semelhança topológica entre si. Estes endereços IP normalmente são originários de várias redes e provedores de serviço de *Internet* diferentes. A observação deste tipo de comportamento geralmente indica a existência de botnets estruturadas em redes fast-flux, uma vez que serviços não-maliciosos com um TTL baixo, como por exemplo *google.com* e *facebook.com*, retornam endereços IP que tem forte semelhança, indicando que são originários da mesma rede e que são relacionados uns com os outros.

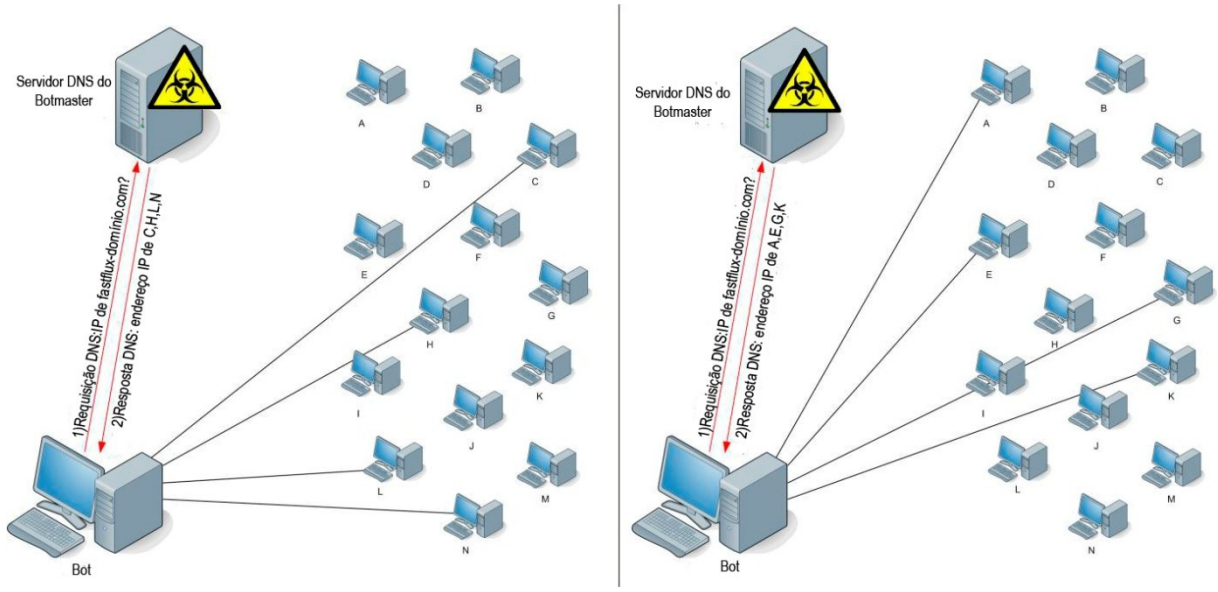

Figura 8: Rastreamento de redes *fast-flux*
Fonte: ENISA (2011)

Exemplo de aplicação da técnica de monitoramento de redes *fast-flux*:

- Holz (2008) apresentou resultados de monitoramento de *botnets* baseadas em redes *fast-flux*. Ele utilizou a *Arbor Networks Threat Analysis System Level* (ATLAS), um sistema de rede baseado em *honeypots* que possui a funcionalidade de rastreamento de redes *fast-flux*. O sistema levanta nomes de domínio possivelmente ligados a *botnets* baseadas em redes *fast-flux* levando em consideração características vinculadas à resolução

de nomes como: TTL, número e distância dos endereços IP de atendimento e seus números de AS. Durante o período de janeiro a maio de 2008, 928 domínios *fast-flux* diferentes, com um total de mais de 15 milhões de endereços IP associados, foram identificados. Mais de um terço dos domínios obtidos permaneceu ativo por menos de uma semana, com um pico de apenas um dia ou menos. Verificou-se também que em 80% dos casos, a ativação do nome de domínio ocorreu mais de um mês depois de seu registro.

Com relação à avaliação da técnica de monitoramento de redes *fast-flux* tem-se:

1) Complexidade: Esta técnica possui um baixo nível de complexidade uma vez que se baseia basicamente na análise de consultas à servidores DNS, não sendo necessários infra-estrutura e equipamentos especiais.
2) Evasão do *botmaster*: A evasão do *botmaster* desta técnica é difícil pois as informações que caracterizam uma *botnet* baseada em uma rede *fast-flux* são consequências diretas da estruturação e configuração desta rede.
3) Falso-positivo: Esta técnica tende a possuir uma taxa de falso-positivo baixa uma vez que as informações que caracterizam *botnets* baseadas em redes *fast-flux* são peculiares a este tipo de rede maliciosa.
4) Falso-negativo: Esta técnica tende a possuir uma taxa de falso-negativo baixa uma vez que as informações que caracterizam *botnets* baseadas em redes *fast-flux* são bem definidas conforme Holz (2008).

4.1.2.4. Medição IRC

O canal de comunicação usado, *pelo botmaster*, para emitir comandos em uma *botnet* pode ser implementado usando uma variedade de protocolos (por exemplo, HTTP, P2P, etc.). Atualmente, aproximadamente 31% das *botnets* usa o protocolo Internet Relay Chat (IRC) de acordo com SYMANTEC (2010). Segundo Rajab (2006), o IRC é um protocolo de bate-papo leve e robusto que permite várias formas de comunicação (*unicast*, *multicast* e *broadcast*, conforme visto no capítulo 2 deste trabalho) e possibilita a disseminação de dados entre um grande número de *hosts*.

Desta forma, este protocolo simplifica a implementação de uma *botnet* e fornece um alto grau de controle sobre os *bots*.

Conforme ENISA (2011), uma *botnet* baseada em um servidor de comando e controle IRC utiliza canais IRC para que *bots* assinalem a sua presença e recebam seus comandos. Normalmente, os *bots* recém-criados tem como primeira ação a conexão com um canal IRC malicioso para o recebimento de instruções. Para a detecção de *bots* a partir de um modelo de comunicação baseado em IRC, são necessários dois passos principais:

- 1º passo - Obter as informações necessárias para entrar e participar de um canal do servidor de comando e controle IRC da *botnet*: As informações básicas de conexão são - endereço IP e o número da porta do servidor de IRC, bem como o canal utilizado para controlar a *botnet.* Estas informações podem ser extraídas a partir de amostras do *malware* da *botnet* em questão, obtidas através de um honeypot por exemplo. Dependendo da complexidade do mecanismo de comunicação da *botnet*, é necessário a obtenção de dados de autenticação e criptografia.
- 2º passo - Realizar a medição IRC: Informações podem ser coletadas a partir da análise de canais de comando e controle IRC. A qualidade e a quantidade de informação obtida depende das precauções tomadas pelo *botmaster*. Se o canal é hospedado de maneira pública no servidor IRC, é possível o rastreamento de nomes de usuários no canal sem esforços adicionais. Além disso, mensagens de status IRC podem dar informações adicionais sobre a flutuação dentro da *botnet*. Entretanto, implementações modificadas tem sido observadas. Por exemplo, comandos já incluídos no protocolo de bate-papo IRC podem ser removidos e a comunicação pode passar a ser realizada através de mensagens privadas entre o *botmaster* e os *bots* com o intuito de se esconder a presença dos *bots* uns dos outros. O protocolo também pode usar criptografia, limitando a quantidade de informação que pode ser obtida por residentes do canal de comando e controle.

Exemplo de aplicação da técnica de medição IRC:

- Rajab (2006) implantou uma infra-estrutura de detecção para *botnets* baseadas no protocolo IRC. Ao longo de um período de mais de três meses,

esta infra-estrutura foi utilizada para rastrear 192 *botnets* IRC de tamanho variando entre algumas centenas e vários milhares de *bots*. Além disso, foram descobertas evidências de infecção *botnet* em 11% dos 800 mil domínios DNS examinados, indicando uma alta diversidade entre as vítimas rastreadas.

Com relação à avaliação da técnica de medição IRC tem-se:

1) Complexidade: Esta técnica possui um nível de complexidade muito alto uma vez que é necessário a utilização de outra técnica de detecção, *honeypot* por exemplo, para a obtenção das informações iniciais sobre o servidor de comando e controle IRC.
2) Evasão do *botmaster*: A evasão do *botmaster* desta técnica é difícil pois a máquina de monitoramento se comporta como um *bot* conectado a um canal de comando e controle do servidor IRC. Assim sendo, torna-se complexa a tarefa de identificar esta máquina monitora e se evadir desta. Os *botmasters* podem criar contramedidas como:
    - Autenticação: A necessidade de autenticação para o estabelecimento da conexão com o canal de comando e controle do servidor IRC pode dificultar atividades de detecção de *botnets*.
    - Criptografia: Dificulta a obtenção de informações a partir de um canal de comando e controle do servidor IRC.
    - Canal em modo privado: O canal em modo privado impossibilita que uma máquina de monitoramento conectada a um canal de comando e controle de um servidor IRC possa enxergar as outras máquinas (*bots)* conectadas a este canal.
3) Falso-positivo: Esta técnica tende a possuir uma taxa de falso-positivo muito baixa uma vez que ela monitora canais de comando e controle de servidores IRC de *botnets*, meios maliciosos por definição.
4) Falso-negativo: Esta técnica tende a possuir uma taxa de falso-negativo muito baixa uma vez que ela monitora canais de comando e controle de servidores IRC de *botnets*, meios maliciosos por definição.

4.1.2.5. Enumeração P2P

Apesar do protocolo IRC ser o mais utilizado no controle de *botnets*, outra abordagem comum é o emprego da infra-estrutura *peer-to-peer* baseada no protocolo P2P. Segundo ENISA (2011), a idéia básica é criar uma rede com seu próprio esquema de endereçamento e protocolo de encaminhamento de mensagens, apenas entre os participantes. *Botnets* famosas, como por exemplo *egStorm*, *Waledac* e *Conficker*, utilizam mecanismos *peer-to-peer*.

Conforme verificado no capítulo 2 deste trabalho, em uma *botnet peer-to-peer*, as *bots* formam uma rede de baixo acoplamento em que cada par conhece apenas um grupo limitado de outros pares. Estas relações fornecem uma arquitetura robusta, sem a necessidade de servidores centralizados para administração da *botnet*. Neste contexto, a robustez desta arquitetura está justamente no fato de não haver um ponto único de falha, o que a torna mais resistente contra medidas de desativação tais como o bloqueio do servidor de comando e controle.

Como se verifica na Figura 20, cada nó neste tipo de rede é identificado por uma chave única, combinada com informações adicionais como o endereço IP e o número da porta. As chaves são utilizadas para indexação e busca de nós e para encaminhamento de mensagens. Cada *peer* mantém uma lista com as chaves dos *peers* vizinhos para manter a conectividade com a rede. (BALAKRISHNAN, 2003)

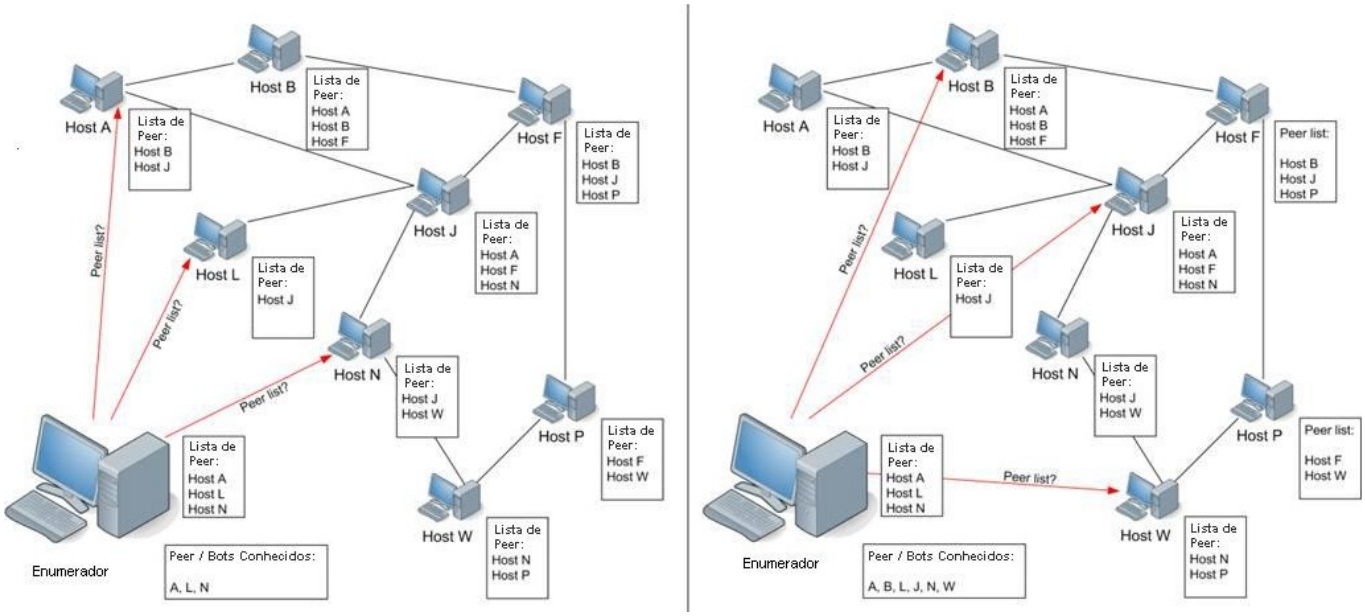

Figura 9: Enumeração *peer-to-peer*
Fonte: ENISA (2011)

Apesar das informações estruturais sobre a *botnet* não estarem disponíveis em um ponto central, a estrutura da rede pode ser explorada para fins de medição, através de uma enumeração recursiva. A realização desta técnica ocorre em dois passos principais analogamente à técnica de medição IRC:

- 1º passo - Participar de uma *botnet* P2P: Esta participação pode ser obtida através de um *honeypot* ou de engenharia reversa do protocolo de comunicação.
- 2º passo - Realizar a enumeração P2P: Implementação de um *bot* para realizar a tarefa de enumeração, onde através de consultas às listas de *bots* vizinhos de cada nó, é possível a enumeração da *botnet* recursivamente.

Exemplos de aplicação da técnica de enumeração P2P:

- Holz (2008) Stock (2009) demonstraram abordagens para a sistemática de medição de uma botnet *peer-to-peer* com base na enumeração recursiva dos *peers*. Suas abordagens dependiam de uma reconstrução do protocolo de comunicação, que envolvia a exploração de falhas da criptografia utilizada, para a obtenção da capacidade de comunicação com os *bots* da *botnet* através deste protocolo.
- Enright (2008) criou um rastreador para a *botnet* P2P *Storm*, chamado *Stormdrain*. Ele rastreou a inserção de cerca de 5000 novos nós na *botnet* em um período de duas horas.

Com relação à avaliação da técnica de enumeração P2P tem-se:

1) Complexidade: Esta técnica possui um nível de complexidade muito alto uma vez que é necessário a utilização de outra técnica, engenharia reversa do protocolo de comunicação por exemplo, para a obtenção da capacidade de comunicação com os *peers* da rede P2P.
2) Evasão do *botmaster*: A evasão do *botmaster* desta técnica é difícil pois a máquina de monitoramento se comporta como um *bot* conectado a rede P2P. Assim sendo, torna-se complexa a tarefa de identificar esta máquina monitora e se evadir desta. Entretanto, a estrutura de rede P2P oferece flexibilidade e

robustez ao *botmaster*, uma vez que a *botnet* não fica centralizada em um servidor de comando e controle como acontece nas outras estruturas de rede.

3) Falso-positivo: Esta técnica tende a possuir uma taxa de falso-positivo muito baixa uma vez que ela monitora meios maliciosos por definição.
4) Falso-negativo: Esta técnica tende a possuir uma taxa de falso-negativo muito baixa uma vez que ela monitora meios maliciosos por definição.

4.1.2.6. Infiltração

Segundo ENISA (2011), a infiltração estende a idéia das técnicas Medição IRC e Enumeração P2P. Ao invés de emular ou modificar o *bot* em uma máquina de monitoramento com a intenção de participar da *botnet* e realizar medições internamente, a infiltração vai um passo além e pretende assumir o controle da *botnet*. Isso geralmente requer como ponto de partida a engenharia reversa dos mecanismos de comunicação utilizados pela botnet, onde uma análise profunda busca identificar fraquezas potenciais. O conhecimento obtido neste processo de análise pode ser explorado em medidas para alcançar uma posição de comando e controle dentro da *botnet*.

Com relação à avaliação da técnica de infiltração tem-se:

1) Complexidade: Esta técnica possui um nível de complexidade muito alto uma vez que é necessário a utilização da técnica de engenharia reversa do protocolo de comunicação para a obtenção da capacidade de infiltração na *botnet*.
2) Evasão do *botmaster*: A evasão do *botmaster* desta técnica é muito difícil pois a máquina de monitoramento se comporta como um *bot* conectado a rede e atua diretamente sobre as deficiências do protocolo de comunicação que mantém a rede maliciosa.
3) Falso-positivo: Esta técnica tende a possuir uma taxa de falso-positivo muito baixa uma vez que ela monitora meios maliciosos por definição.
4) Falso-negativo: Esta técnica tende a possuir uma taxa de falso-negativo muito baixa uma vez que ela monitora meios maliciosos por definição.

4.2. Quadro comparativo

Sendo assim, o terceiro objetivo específico da pesquisa - Consolidar um estudo comparativo entre as técnicas de detecção das *botnets,* tendo por base os protocolos HTTP, DNS, IRC e P2P – foi cumprido neste capítulo, onde foram pesquisadas 10 técnicas de detecção de *botnets*, classificadas como passivas e ativas, exemplificadas, analisadas segundo os parâmetros: complexidade, evasão do *botmaster,* taxa de falso positivo e taxa de falso negativo; e sintetizadas neste quadro comparativo entre as técnicas de detecção.

TECNICA	METRICA	PROTOCOLOS				PARAMETROS DE AVALIAÇAO			
		DNS	HTTP	IRC	P2P	Complexidade	Evasão do *botmaster*	Taxa de falso positivo	Taxa de falso negativo
Inspeção de pacotes	- Assinaturas de detecção		X	X	X				
Análise de registros de fluxo	- Registros de fluxo: • IPs origem e destino • Portas origem e destino • Protocolo • Duração da sessão • Tamanho cumulativo • Número de pacotes		X	X	X				
Análise de arquivis de log	- Registros de arquivos de log		X	X	X				
Honeypots	- Captura e análise de *malware*		X	X	X				
Sinkholing	- Acessos a nomes de domínio maliciosos	X							
Snooping cache DNS	- Acertos em *cache* de nomes de domínio maliciosos	X							
Monitoramento de redes *fast-flux*	- Nomes de domínio com TTL baixo e que devolvem um conjunto de endereços IP com pouca ou nenhuma semelhança topológica	X							
Medição IRC	- Engenharia reversa do protocolo de comunicação do *malware* - Análise de canais de servidores de comando e controle IRC			X					
Enumeração P2P	- Engenharia reversa do protocolo de comunicação do *malware* - Enumeração recursiva de *bots*				X				
Infiltração	- Engenharia reversa do *malware* - Domínio da *botnet*			X	X				

LEGENDA DAS TÉCNICAS
Técnicas passivas
Técnicas ativas

LEGENDA DOS PARÂMETROS	
Muito positivo	
Positivo	
Neutro	
Negativo	
Muito negativo	

5 CONCLUSÃO

O presente trabalho pautou-se no estudo sobre os mecanismos de detecção das *botnets*, tendo por base os protocolos HTTP, DNS, IRC e P2P.

O primeiro objetivo específico da pesquisa - Revisar os protocolos HTTP, DNS, IRC e P2P - foi cumprido no segundo capítulo, onde foi realizada uma revisão sobre estes protocolos, o que proporcionou o entendimento pleno do funcionamento das técnicas de detecção de *botnets* fundamentadas nestes protocolos.

O segundo objetivo específico da pesquisa - Estudar as *botnets*: definição, componentes, tipos, objetivos e ciclo de vida – foi cumprido no terceiro capítulo, onde foi desenvolvido um estudo sobre estes aspectos conceituais relacionados às *botnets.* Este estudo viabilizou a compreensão da relação entre as técnicas de detecção e os componentes, tipos, objetivos e ciclo de vida das *botnets*.

O terceiro objetivo específico da pesquisa - Consolidar um estudo comparativo entre as técnicas de detecção das *botnets,* tendo por base os protocolos HTTP, DNS, IRC e P2P – foi cumprido no quarto capítulo, onde foram pesquisadas 10 técnicas de detecção de *botnets*, classificadas como passivas e ativas, exemplificadas e analisadas segundo os parâmetros: complexidade, evasão do *botmaster,* taxa de falso positivo e taxa de falso negativo. Esta pesquisa possibilitou a análise das técnicas de detecção tanto isolada quanto relativamente às outras técnicas.

Nessa perspectiva, foram reunidas informações que possibilitaram a constituição de um embasamento teórico consistente para subsidiar o cumprimento do objetivo geral da pesquisa - Realizar um estudo sobre os mecanismos de detecção das *botnets*, tendo por base os protocolos HTTP, DNS, IRC e P2P. Sendo assim, consolidou-se o quadro comparativo (conforme verificado no capítulo 4) entre as técnicas de detecção das *botnets,* tendo por base os protocolos HTTP, DNS, IRC e P2P. Por fim, este estudo configura-se como subsídio para outras pesquisas sobre o assunto, como por exemplo, os trabalhos em andamento e trabalhos futuros no Instituto Militar de Engenharia: PIBITI (Programa Institucional de Bolsas de Iniciação em Desenvolvimento Tecnológico e Inovação), TD (Trabalho Dirigido), IP (Iniciação à Pesquisa), PFC (Projeto de Fim de Curso) e teses de Mestrado e Doutorado.

REFERÊNCIAS

ALBITZ, Paul; LIU, Cricket. DNS e BIND, 4ª ed., Rio de Janeiro: Campus, 2001.

APPLIED META COMPUTING. (2000). Legion - An Integrated Architecture for Secure Resource Sharing. Applied Meta Computing Peer-to-Peer Architectural Proposal.

BALAKRISHNAN, h., KAASHOEK, m., KARGER, d., MORRIS, r., STOICA, I. (2003). Looking up data in p2p systems. 42-43. ACM.

BINKLEY, J. R., SINGH, S. An algorithm for anomaly-based botnet deteccion. Proceedings of the second conference on steps to reducing unwanted traffic on the Internet (SRUTI 06), 2006.

CHOI, H., LEE H., KIM H. Botnet Detection by Monitoring Group Activities in DNS Traffic. Proceedings of the 7th IEEE International Conference on Computer and Information Technology (CIT '07), 2007.

COMER, Douglas E., Interligação em Rede com TCP/IP, Volume I, Princípios, Protocolos e Arquitetura. Tradução da 5ª edição, Editora Campus, 2006.

CUI, W., PAXSON, V., Weaver, N. C. GQ: Realizing a System to Catch Worms in a Quarter Million Places. ICSI technical report TR-06-004, 2006.

DE CAMPOS, David Robert Camargo; JUSTO, Rafael Dantas. Tutorial DNSSEC. Versão 1.4.4. Disponível em <ftp://ftp.registro.br/pub/doc/tutorial-dnssec.pdf>. Acesso em 29 Ago. 2011.

ENRIGHT, B., VOELKER, G., SAVAGE, S., KANICH, C., LEVCHENKO, K. Storm: When researchers collide. USENIX; login, vol. 33(4), 2008.

FONSECA, Fernando Sérgio Santos. Sistema de tratamento de arquivos de logs. Universidade Federal de Lavras, 2005.

GOEBEL, J., HOLZ, T., WILLEMS, C. Measurement and Analysis of Autonomous Spreading Malware in a University Environment. Proceedings of the 4th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA '07), 2007.

HAMMAN, Robin. Computer networks linking network communities: effects of AOL use upon pre-existing communities. 1999. Online em <http://www.socio.demon.co.uk/cybersociety/> Acesso em 01 Set. 2011.

HIGGINS, Kelly Jackson. The world´s biggest botnets. Disponível em <http://www.darkreading.com/security/security-management/208808174/index.html>. Acesso em: 25 Abr. 2012.

HOLZ, T., NAZARIO, J. As the Net Churns: Fast-Flux Botnet Observations. Proceedings of the 3rd International Conference on Malicious and Unwanted Software (MALWARE'08), 2008.

HOLZ, T., STEINER, M., DAHL, F., BIERSACK, E., FREILING, F. Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET'08), 2008.

INTERNET CORPORATION FOR ASSIGNED NAMES AND NUMBERS – ICANN. ICANN destaca vulnerabilidade do Sistema de Nomes de Domínio e fornece ferramentas: ICANN chama a atenção para importante problema de segurança na Internet. Disponível em <http://www.icann.org.br/announcements/announcement-06aug08.htm>. Acesso em: 13 Set. 2011.

KUROSE, James F. e ROSS, Keith W., Redes de computadores e a Internet - Uma Abordagem top-down,3ª edição, Pearson Addison Wesley, São Paulo, 2006.

LEDER, F., WERNER, T. Containing Conficker - To Tame A Malware. Honey Project, Know Your Enemy series, 2009.

LI, Zhichun. GOYAL, Anup. CHEN, Yan. Honeynet-based Botnet Scan Traffic Analysis. Invited book chapter for "Botnet Detection: Countering the Largest Security Threat", Springer, 2007.

LINARI, A., BUCKLEY, O., DUCE, D., MITCHELL, F., MORRIS, S. A methodology for anomaly and botnet detection and characterisation from application logs, 2010.

MICHEL, Jean-Christophe. Voyage en DNS au centre de l’espace de nommage d’Internet. Disponível em: <http://www.gecif.net/articles/linux/dns.html>. Acesso em: 25 Ago. 2011.

ORAM, A. (2001). Peer-to-peer: Harnessing the power of disruptive technologies. O’Relly.

RAJAB, M. A., MONROSE, F., TERZIS, A., PROVOS, N. Peeking through the cloud: DNS-based estimation and its applications. Proceedings of the 6th international conference on Applied cryptography and network security (ACNS'08), 2008.

RAJAB, M. A. ZARFOSS, J. MONROSE, F. TERZIS, A. A Multifaceted Approach to Understanding the Botnet Phenomenon. Department of Computer Science, Johns Hopkins University, 2006.

SACCHETIN, M. C. et al. Botnet detection and analysis using honeynet. International Journal of Forensic Computer Science, 2008.

MALL, S., MASON, J., MONROSE, F., PROVOS, N., STUBBLEFIELD, A.To catch a predator: a natural language approach for eliciting malicious payloads. Proceedings of the 17th conference on Security symposium (SS'08). 2008.

SPITZER, L. Honeypots: Tracking Hackers. Addison-Wesley Professional, 2002.

STOCK, B., GOEBEL, J., ENGELBERTH, M., FREILING, F., HOLZ, T. Walowdac - Analysis of a Peer-to-Peer Botnet. European Conference on Computer Network Defense (EC2ND), 2009.

SYMANTEC. State of Spam & Fishing - A Monthly Report. Issue December 2010.

TANENBAUM, Andrew. Redes de Computadores. 4. ed. Rio de Janeiro: Campus Elsevier, 2003.

_____. Distributed Systems. 2. ed. Amsterdam: Pearson Prentice Hall, 2006.

VAZ, Ricardo. Tudo sobre registro de domínio: DNS e DNSSEC. Disponível em <http://ricardovazmonteiro.blogspot.com/2007/10/dns-e-dnssec.html>. Acesso em: 09 Set. 2011.

VILLAMARIN-SALOMON, R., BRUSTOLONI, J.C. Identifying Botnets Using Anomaly Detection Techniques Applied to DNS Traffic. Proceedings of the 5th IEEE Consumer Communications and Networking Conference (CCNC'08), 2008.

VOSS JR, J. Protótipo de software para compartilhar informações entre computadores através da tecnologia peer-to-peer (p2p), usando a plataforma jxta. 2004. 71f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) Universidade Regional de Blumenau, Blumenau – 2004.

YANG, B., & GARCIA-MOLINA, H. (2001). Comparing Hybrid Peer-to-Peer Systems. pp. 561-570.

YEN, T. F., Reiter, M. K. Traffic aggregation for malware detection. Proceedings of the 5th international conference on detection of intrusions and malware, and vulnerability assessment (DIMVA 08), 2008.

ZHU, Z. et al. Botnet research survey. In: Computer Software and Applications, 2008, Turku, Finland. Proceedings… Turku, Finland: IEEE Intenational, 2008.

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 – Praia Vermelha
Rio de Janeiro – RJ CEP 22290-270

621.39 Moreira, Rômullo Girardi. Gonçalves, Diego Furtado.

Estudo sobre detecção de *botnets* / Rômullo Girardi Moreira, Diego Furtado Gonçalves; orientado por Sérgio dos Santos Cardoso Silva. Rio de Janeiro: Instituto Militar de Engenharia, 2012.

63 f : il.