Centro Universitário de Brasília

Instituto CEUB de Pesquisa e Desenvolvimento - ICPD

José Ozete Cunha Filho

SIMULAÇÃO E ANÁLISE DE *BOTNET* COM CONTROLADORES EM SERVIDORES IRC: UMA IMPLEMENTAÇÃO COM MÁQUINAS VIRTUAIS

BRASÍLIA

2012

José Ozete Cunha Filho

SIMULAÇÃO E ANÁLISE DE *BOTNET* COM CONTROLADORES EM SERVIDORES IRC: UMA IMPLEMENTAÇÃO COM MÁQUINAS VIRTUAIS

Monografia apresentada ao Centro Universitário de Brasília (UniCEUB/ICPD) para obtenção do título de especialista em Redes de Computadores com Ênfase em Segurança.

Orientador do Projeto: Marco Antônio de O. Araújo, MSc.

BRASÍLIA

2012

Dedicatória

À minha esposa, Roberta, pela compreensão, incentivo e amor incondicional.

À minha família, sem a qual não estaria aqui, pelo incentivo e investimento na educação e formação do caráter.

A Deus, por todas as oportunidades da vida.

Agradecimentos

Agradeço primeiramente ao meu orientador, Prof. MSc. Marco Antônio, que de imediato se prontificou a acompanhar meu trabalho de pesquisa.

Aos demais professores do curso, que transmitiram o conhecimento necessário para o aperfeiçoamento na vida profissional.

Ao pessoal da Secretária da Pós-Uniceub.

Aos colegas de turma, em especial ao Bruno, Leandro e Renan, pelo companheirismo e dedicação aos trabalhos e provas durante o curso.

Ao colega de trabalho, Fava, pelo apoio no farto material cedido e pela grande experiência e conhecimento.

A dificuldade induz ao desafio e a dedicação define o caminho da vitória. (Autor desconhecido)

RESUMO

Nos últimos anos temos observado um aumento constante e significativo no número de computadores de uso doméstico em todo mundo. O avanço da tecnologia da informação, a internet e a inclusão digital propiciaram o acesso amplo à informação e à democratização do conhecimento. Porém essa ampla utilização dos serviços prestados pelo uso dos computadores também ocasionou diversos problemas, como a fragilidade da segurança e privacidade das informações. Inerente a isto, o anonimato permitido pela internet incentiva que pessoas mal intencionadas desenvolvam diversas técnicas de invasão a sistemas computacionais, podendo gerar inúmeros problemas. Uma dessas técnicas é a *botnet* ou rede zumbi. *Botnet* é uma rede de computadores em que cada máquina é contaminada por código malicioso, sendo controlada por um agente externo, o *botherder*. As intenções, de um modo geral, de uma rede zumbi são o ganho financeiro, envio de *spam*, ataques *ddos*, *phishing*, *hacktivismo* e até terrorismo. Para conseguir tais objetivos, os *botherder* vêm desenvolvendo vários modelos de realizar ataques através das *botnets*. No presente estudo será apresentada a RxBot, botnet controlada por meio de servidor Internet Relay Chat, sendo mostrada sua dinâmica de criação, contaminação e propagação. Nesse contexto, foi criada uma plataforma de simulação através de máquinas virtuais, provendo um serviço de IRC em uma estação Linux, e demais estações com Windows XP, simulando os nós vítimas.

ABSTRACT

In the recent years, a constant and significant increase in the number of household computers worldwide has occurred. The advance of information technology, Internet and digital inclusion led to a widespread access to information and democratization of knowledge. But the wide use of services provided by the use of computers has also caused problems such as the fragility of security and privacy of information. Inherent in this, the anonymity the "internet space" encourages people to develop various malicious hacking techniques to computer systems and can generate numerous problems. One of these techniques is the botnet or zombie network. Botnet is a network of computers where each machine is infected by a malicious code controlled by an external agent, the botherder. The intent, in general, of a botnet is financial gain, sending spam, ddos attacks, phishing, hacktivism and even terrorism. To achieve these objectives, the botherder developed several models to carry out attacks via botnets. The present study will present the RxBot, botnet controlled through Internet Relay Chat server, and displayed it´s dynamic creation, and spread contamination. In this context, we built a simulation platform using virtual machines, providing a IRC service in a Linux station, and other stations with Windows XP, simulating the victims.

Lista de Figuras

Lista de Tabela

Lista de Abreviaturas

A/V	Antivírus
C&C	Comando e Controle
DCOM	Distributed Component Object Model
DDOS	Distributed denial-of-service
DNS	Domain Name System
DOS	Denial-of-service
FOPE	Microsoft Forefront Online Protection for Exchange
FTP	File Transfer Protocol
HTML	Hypertext Markup Language
HTTP	Hypertext Tranfer Protocol
ICMP	Internet Control Message protocol
IP	Internet Protocol
IRC	Internet relay Chat
ISP	Internet Service Provider
MAC	Media Access Control
MTU	Maximum Transmission Unit
NAT	Network Adress Translation
NVD	National Vunerability Database
P2P	Peer-to-Peer
RFC	Request for Comments
SMTP	Simple Mail Transfer Protocol

SSL	Secure Socket Layer
TCP	Transmission Control Protocol
UCE	Unsolicited Commercial E-mail
UDP	User Datagrama Protocol
URL	Universal Resource Locator
VM	Virtual Machine
VMM	Virtual Machine Monitor

SUMÁRIO

1. INTRODUÇÃO

Botnet é uma rede de computadores, onde cada estação é infectada por um código malicioso, tornando-a controlada por um agente externo, também conhecido por *botmaster* ou *botherder*. Já as estações contaminadas são conhecidas por "nó" ou "zumbi", e nada mais são que computadores comuns de residências ou escritórios ao redor de todo o mundo (MICROSOFT, 2011a).

Segundo o último relatório de ameaças da Symantec, o número de ataques de *malwares* pela web alcançou 3,1 bilhões, sendo detectados cerca de 286 milhões de variantes destes softwares maliciosos (SYMANTEC, 2011a).

No mesmo estudo, o Brasil se mantém em primeiro lugar entre os países latino-americanos, e em quinto lugar mundial, com maior número de computadores infectados por *botnet* (SYMANTEC, 2011b).

Já no Relatório de inteligência de segurança da Microsoft, o Brasil é um dos locais com as maiores taxas de infecção, chegando ao segundo lugar no ranking, perdendo apenas para os Estados Unidos, e alcançando marcas de dois milhões a três milhões de relatos de detecção de *malwares*, por trimestre, nos ambiente Microsoft Windows (MICROSOFT, 2011b).

Para Strayer et al. (2010), botnet é a forma mais perigosa de ataques em uma rede de computadores porque o número de hosts envolvidos – as vítimas – é muito grande e, também, pelo fato de serem controlados à distância.

Ainda, para os autores, a utilização de servidores IRC –*Internet Relay Chat* –é a forma mais comum de se obter um mecanismo de controle de *botnet*, isso devido a sua

escalabilidade e manutenção do anonimato. Nessa forma de cadeia de controle, os atacantes comandam os zumbis através dos serviços de chat que o servidor IRC disponibiliza, que por sua vez, são servidores, quase sempre, comprometidos.

Stinson e Mitchell (2010) explicam que cada participante da rede zumbi executa independentemente um comando com alguns parâmetros recebido do C&C (Comando e Controle) da *botnet*, tudo de forma imperceptível.

No estudo de Bambenek e Klus (2010) é apontado seis características desejáveis de toda *botnet* para obter o máximo de ganho financeiro ou o máximo de impacto no alvo, são elas: alta capacidade, baixo *overhead*, respostas rápidas, flexibilidade, anonimato e algo que eles chamam de “manter-se quieto”.

A alta capacidade consegue manter diversos hosts sobre seu domínio e, nesse aspecto, o protocolo IRC se sobressai por conseguir que milhares de máquinas se concentrem em um mesmo canal e comandos simples sejam enviados para todas as estações simultaneamente.

Há que se destacar, segundo os autores, como características principais o anonimato e a indetectabilidade de uma *botnet*, à medida que o sucesso de propagação depende de quão eficiente seja sua não rastreabilidade.

Diante do que já foi exposto, percebe-se que o impacto na utilização de *botnet* é gigantesco. Fato este corroborado pela Microsoft (MICROSOFT, 2011c), que considera a proliferação das *botnets* algo muito preocupante e tem levado, incrivelmente, algumas pessoas a acreditar que é um problema sem solução. De acordo com seus dados, estima-se que os controladores de *botnet* produzam um rombo de aproximadamente 780 milhões de dólares somente com o envio de *spam* em todo mundo.

No Brasil, Howard (2009) discorre um capítulo inteiro sobre o cenário das ameaças cibernéticas em nosso País, o qual é qualificado como "*altamente especializado*" em relação às atividades fraudulentas, sobretudo de *trojans* bancários disseminados através de "*sofisticados ataques*" de *phishing*.

No que tange o modelo escalar pretendido para realizar a simulação de um ambiente virtual semelhante à internet, Muzzi (2010) propôs, em sua tese de doutorado, uma plataforma escalar para analisar a propagação e comunicação de *botnets*. Para tanto, foi utilizado a técnica de virtualização, onde foi possível construir uma plataforma batizada de XnetSim, a qual o autor realizou um série de estudos quanto à propagação e contenção das *botnets*.

1.1 Motivação e Justificativa

Em todos os casos, o objetivo perquirido pelos controladores de *botnet* é para fins ilícitos, tais como envio de *spam*, ataques de negação de serviço, coletas de contas de e-mails de terceiros, roubo de identidade e, de uma forma mais avançada, realizar computação distribuída (MUZZI, 2010).

Deste modo, considerando os mais recentes Relatórios de Ameaças à Internet de grandes empresas, como Microsoft, Symantec, Trend, Macfee, fica evidente a crescente preocupação com o número de computadores infectados por códigos maliciosos, os quais fazem parte dessa grande teia de computadores.

Em outro contexto, podem ser verificadas também as conseqüências devastadoras que uma *botnet* pode causar em casos de ciberataques ou ciberguerras. Estas extrapolam o campo técnico-científico, pois são, ordinariamente, baseadas em questões políticas,

econômicas e no fundamentalismo religioso, colocando em risco a integridade e a soberania de um Estado.

Para Andress e Winterfeld (2011), a definição de ciberguerra não é nada fácil. Para tanto, eles discorrem apartadamente sobre os conceitos de ciberespaço e de guerra, e, por fim, faz um questionamento de tais enfoques com o que acontece no mundo virtual atual. Então, conclui-se, sucintamente, que ciberguerra é uma modalidade de guerra realizada no campo virtual.

Já no estudo de Sampaio (2011), a definição de ciberguerra é clarificada e estendida além dos termos de guerra eletrônica, abrangendo também as operações de guerra psicológica, a teoria da mentira, o terrorismo seletivo e até a manipulação do sistema nervoso humano, visando o domínio, controle e a conseqüente paralisação de um adversário, no caso, um país, um bloco econômico ou militar.

Uma modalidade de ciberguerra, o ciberterrorismo, é a convergência do terrorismo para o ciberespaço. São ataques ilegais contra computadores, redes ou informações neles armazenadas, quando é feito com pretensão de intimidação ou coação de um governo ou povo, com objetivos políticos, religiosos ou sociais, com violência contra pessoas ou bens ou com danos que, de alguma forma, gerem medo (DENNING,2011).

Nesta proposição de ciberguerra, os militares tornam-se civis, as armas são computadores e o campo de guerra é a própria internet, porém, valendo-se de um grande aliado, o anonimato que o ambiente virtual proporciona.

1.2 Objetivos

1.2.1.Objetivos Gerais

Análise do comportamento de uma *botnet* operada por um servidor IRC em uma plataforma de simulação, mostrando seu funcionamento e provendo dados que visem seu estudo.

1.2.2 Objetivos Específicos

- Fornecer um ambiente que simule uma rede de computadores e, através deste, analisar o comportamento de uma *Botnet*.
- Analisar as formas de contaminação de um zumbi e as vulnerabilidades presentes.
- Deixar criado um ambiente de treinamento onde possa ser mostrado o ciclo de vida de uma *botnet*.

Então, baseado na estratégia de conhecer para atacar, e para intensificar os estudos sobre a prevenção e defesa de infecções de computadores em redes zumbis, deve-se, antes de tudo, conhecer o funcionamento de uma *botnet*, analisando-se, primeiramente, as formas que um alvo é infectado por um código malicioso, identificando as vulnerabilidades dos sistemas operacionais e programas antivírus, bem como as fraquezas do próprio usuário e, posteriormente, verificar o comportamento da *botnet*, desde a propagação e respectiva contenção.

1.3 ORGANIZAÇÃO DO TRABALHO

O presente estudo pretende mostrar, inclusive de forma prática e através da utilização de máquinas virtuais, o funcionamento de uma *botnet*. Para conseguir tais objetivos, pretende-se percorrer as seguintes etapas sequencialmente:

A partir do capítulo 2, tem-se o estudo e compreensão dos principais tópicos relacionados ao tema de *botnet.* A primeira contextualização sobre o tema foi abordada em conceitos e histórico. Em seguida, é apresentada a estrutura geral de uma *bot*, como suas entidades, ciclo de vida, as finalidades e as principais estratégias para controle de uma rede zumbi.

Ainda no capítulo 2, é detalhada a estratégia de controle a partir do protocolo IRC, sendo apresentada a tecnologia desde os primórdios básicos do IRC até sua descaracterização para o cometimento de fraudes com IRC *Botnets*.

O capítulo 3 trata da *botnet* que foi utilizada no presente trabalho, a RxBot, sendo apresentado suas características principais, seu início de vida, os sinais de que a máquina está comprometida, bem como a análise do seu código fonte.

Por fim, no capítulo 4 é criada a plataforma de demonstração que permite a visualização todos os passos, de forma educativa, desde a contaminação até sua propagação. Também são apresentados os conceitos relacionados às máquinas virtuais, suas vantagens, propriedades, utilização e os principais aplicativos utilizados atualmente.

2. AS BOTNETS

2.1 Conceitos

De início, com o advento da internet, a palavra *bot* fazia alusão a robô, referindo-se a programas de softwares automatizados que executam tarefas em uma rede com certo grau de autonomia. Porém, para a Microsoft (2011a), tais *bots* passaram a ser desenvolvidos para fins maliciosos, como montagem de uma rede de computadores comprometidos, as *botnets*, que são controladas remotamente por um *botherder*.

Botnet é um verdadeiro exército de computadores comprometidos a serviço de um controlador, o *botherder*, com o intuito de ganhos financeiros, cometimento de crimes ou alguma forma de atividade maliciosa (SCHILLER,2007).

A CISCO também compara uma *botnet* com um exército de estações contaminadas, denominadas de *zombies*, sob o comando e controle de um *bootmaster*, e que o poder de uma *botnet* foi favorecido pelo aumento da banda larga, bem como o próprio desenvolvimento tecnológico, ocasionando um crescimento de ataques de negação de serviço, infecções de milhões de computadores com *spywares* ou quaisquer outros códigos maliciosos, tudo isso visando, via de regra, o cometimento de atividades fraudulentas (CISCO,2011).

Geralmente, os computadores tornam-se nós de uma *botnet* quando os atacantes, de forma ilícita, instalam um *malware* secretamente que os fazem conectar à *botnet*. Essa forma de ataque explora vulnerabilidades de softwares, do sistema operacional ou técnicas de engenharia social para induzir que os usuários instalem o *malware*.

Muzzi (2011) explica que *botnet* é uma forma moderna de ameaça à segurança dos sistemas computacionais, sendo uma rede formada por bots ou robôs que tornam o computador da vítima infectado e monitorado por um agente externo. Esse computador vitimado responderá aos comandos enviados pelo atacante, e, assim, será controlado para fins ilícitos como ataques distribuídos de negação de serviço, roubo de senha, envio de *spam*, roubo de identidades, *phishing* e outros.

Na Figura 2.1 podemos ver um grupo de computadores *zombies* são controlados remotamente pelo *botmaster*. Aqueles foram, de alguma maneira, contaminados por vírus e estão em estado de hibernação esperando que quem os controla forneça algum comando. As ações típicas de uma rede zumbi são de "*spammer*", "*phisher*" ou "*attacker*".

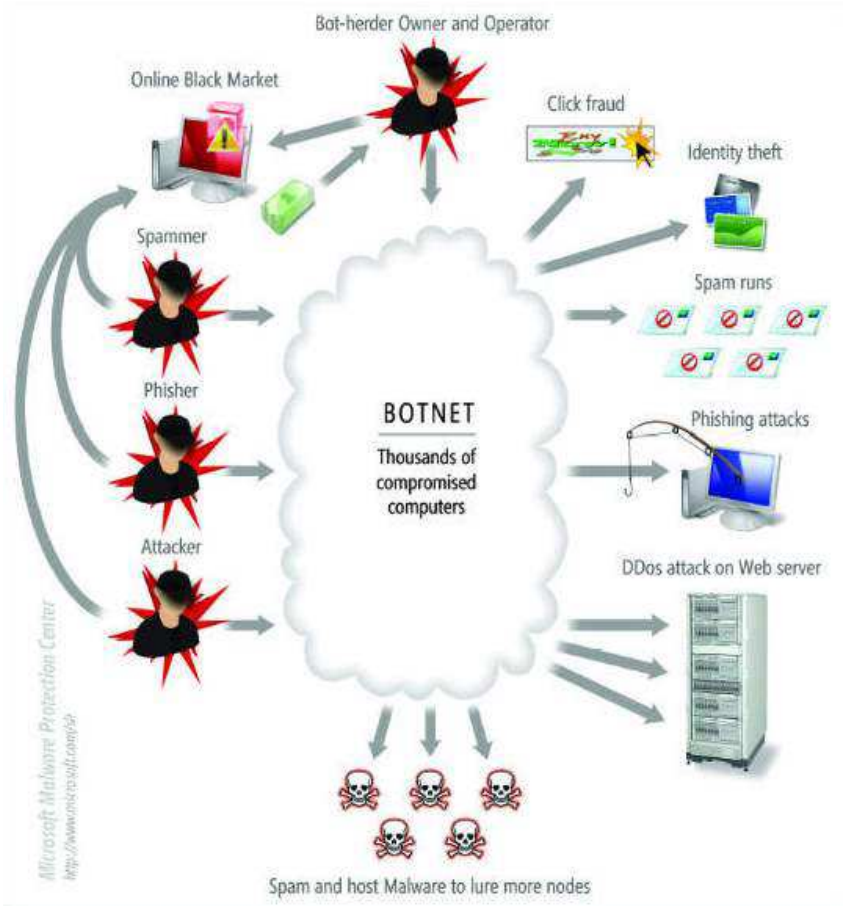

Figura 2.1 – Exemplo típico de uma botnet em ação. Fonte: MICROSOFT, 2011a

2.2 Histórico

Para a Microsoft (2011a) a origem das *botnets* está relacionada com utilitários que foram desenvolvidos para gerenciamento em rede de *Internet Relay Chat*. Essa rede é composta por um certo número de servidores, e estes, por sua vez, possuem canais, nos quais usuários podem conversar. Já os canais são administrados por operadores de canal.

Visando estender funcionalidades do IRC, operadores de canais desenvolveram scripts automatizados – o *IRC Bot* – que passou a executar algumas tarefas pré-determinadas como registros de estatísticas, execução de jogos, transferência e coordenação de arquivos.

Com a popularidade e desenvolvimento dos chats, acirrou o conflito entre os operadores pelo controle dos canais mais populares. Esse esforço para ganhar o comando de canais fez com que operadores mal intencionados ampliassem os *IRC bots* para realizar ataques e obter o controle do canal. Era, então, o primeiro passo dado para a criação das *botnets* conhecidas atualmente.

Seguindo a mesma linha, Dunham e Melnick (2009), após distinguir o que eles chamam de *bots* “ruins” e *bots* “más”, expõem e fazem um comparativo da origem das botnets com o avanço dos *trojans* ou “cavalo de tróia”. Assim, as antigas e simples *bots* “más” juntamente com os *trojans* foram aperfeiçoados até chegar à definição atual das redes zumbis.

2.3 Entidades de uma Botnet

Para continuar na compreensão do funcionamento de uma *botnet*, Amoroso (2011) discorreu sobre os participantes que compõem um ataque de *botnet*:

- **Operador**: pode ser representado por um único indivíduo, um grupo ou até um país, de acordo com as finalidades que se propõe a *botnet*. É responsável pela criação, inclusive sua instalação e operacionalização. No caso de *bots* que tem o desígnio deliberado de auferir lucros financeiros, o operador será o principal beneficente.

- **Controlador**: é um servidor ou conjunto de servidores que comandam e controlam as operações de *botnets*. Geralmente, são servidores que foram comprometidos unicamente com essa finalidade, fazendo com que, muitas vezes, o proprietário do servidor não perceba o que tenha acontecido. O comando e controle têm como funções o recrutamento de nós, as configurações dos servidores, a forma de comunicação e as atividades de ataques. As botnets mais avançadas contam com vários controladores espalhados ao redor do mundo.

- **Coleção de bot**: são os nós, os zumbis, ou seja, os usuários finais, que por sua vez, quase sempre, estão conectados em banda larga e foram infectados por um malware. Como os administradores de servidores comprometidos, esses usuários, geralmente, também desconhecem o fato de estarem contaminados e que estão participando de ataques.

- **Software *Drop***: é o arsenal que uma botnet utilizará durante seu ciclo de vida. Para manutenção do anonimato, tais softwares também são mantidos em servidores comprometidos.

- **Alvo**: como o próprio nome indica, é o local de ataque. Pode ser pessoas, sites, sistemas, redes, religiões, países ou até continentes. Como podemos observar, ninguém está imune, pois o propósito do ataque é que definirá o tipo de alvo.

2.4 Ciclo de vida de uma Botnet

Durante toda a sua existência, as *botnets* sempre seguem um conjunto de passos semelhantes, comumente chamados de ciclo de vida. Para entendimento do funcionamento de uma botnet também se faz necessário compreender tal ciclo.

Segundo Zhu, citado por Muzzi (2010), o invasor inicia suas atividades explorando as vulnerabilidades do sistema alvo, executando o software malicioso do bot.

Schiller (2007) compartilha do mesmo ensinamento informando que o ciclo de vida de um zumbi – ou *botclient* – inicia quando ele sofre algum tipo de exploração. Essa exploração consiste basicamente em umas das formas: um usuário, inocentemente, executa um código malicioso; ataques contra vulnerabilidades não corrigidas; *backdoors* [1]; adivinhação de senhas e tentativas de acesso com força bruta.

[1] Backdoor é uma falha de segurança presente em sistemas operacionais ou aplicativos que permite a invasão do sistema, deixando a estação comprometida e vulneráveis à ataques de hackers.

2.4.1 Exploração e contaminação

Schiller (2007) continua explicitando cada caso relacionado às principais formas de exploração e contaminação:

2.4.1.1 Contaminação por um código malicioso

Existem diversas formas de uma máquina vítima ser contaminada por um código malicioso, e, a partir daí, torna-se mais um nó em uma rede zumbi. Tais formas evoluem e se adaptam, de acordo com a criatividade dos fraudadores, visando sempre iludir os usuários.

Para conseguir seus objetivos, os criminosos virtuais tentam ludibriar de qualquer forma suas vítimas, apelando para todo tipo de abordagem, desde simples mensagens até pressões psicológicas ou ameaças mais contundentes.

Assim, eles aproveitam da inocência das pessoas, exibem fatos incríveis ou estranhos que excitem a curiosidade, recorrem ao lado social e religioso, exibem personalidades conhecidas da mídia e tentam passar credibilidade para a vítima, fornecendo até dados de contato verdadeiros, tudo para conquistar o “clique” da vítima.

- Sites aliciadores com *trojan code*:

Figura 2.2 – Site com código malicioso

- E-mails com anexos que quando abertos executam códigos maliciosos;

Figura 2.3 – E-mail com código malicioso

Figura 2.4 – E-mail com código malicioso

- Spams enviados por pessoas conhecidas que fazem o usuário erroneamente acreditar se tratar de algo sério.

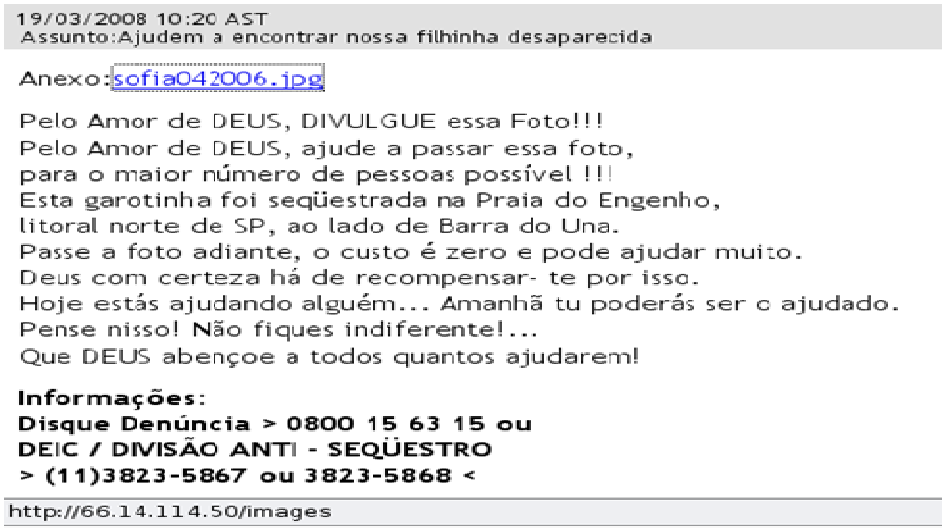

Figura 2.5 – E-mail enviado em spam com código malicioso

- E-mails de *phishing* que atraem ou incitam o usuário a acessar um site falso para que seus dados sejam capturados, como exemplo uma cópia de site de internet banking, onde o usuário fornecerá seus dados bancários, inclusive sua senha.

Figura 2.6 – Modelo de phishing scam

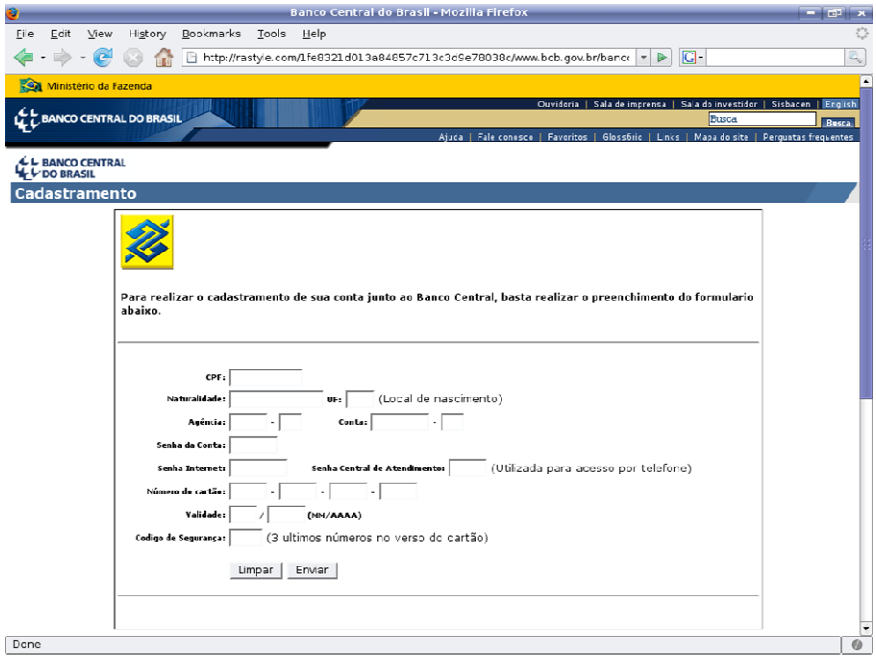

Figura 2.7 – Modelo de phishing scam

2.4.1.2 Ataques contra vulnerabilidades não corrigidas

Como forma de difusão de ataques contra sistemas operacionais ou aplicativos que apresentam vulnerabilidades sem correção, cada nó ou zumbi pode ter a capacidade de realizar um escaneamento, fazendo com que a botnet se expanda cada vez mais.

Essa capacidade consiste em fazer uma varredura no sistema para identificar portas abertas. Em seguida, com a lista de sistemas com portas abertas e usando ferramentas de varredura de vulnerabilidades especificas, tenta-se identificar a atualização ou não do sistema.

Deste modo, a botnet varrerá diretamente os *hosts* que apresentam uma ou mais dessas vulnerabilidades, e, uma vez comprometido, permitirá o controle remoto das estações vulneráveis.

A lista das principais vulnerabilidades presentes atualmente é algo muito dinâmico, ampliada e atualizada a cada dia, não cabendo ser destacada no presente trabalho. Para tanto, um rol exaustivo pode ser encontrado no site da NVD – National Vulnerability Database (2011).

2.4.1.3 Backdoor

Algumas *botnets* têm a capacidade de buscar hosts que já tenham sido contaminados por algum *backdoor*, como os da família *remote access trojan*, permitindo o

controle remoto por outro computador sem a anuência do responsável. Nesse caso, o trabalho da *botnet* é otimizado, já que esta recruta estações potencialmente fragilizadas.

2.4.1.4 Adivinhação de senhas e ataque de força bruta

Existem famílias de *botnet* que possuem a habilidade de buscar e adivinhar usuários e senhas de um sistema operacional vítima. Primeiramente, ela tenta obter acesso com a conta de usuário que está sendo atacado, se não conseguir enumera uma lista de todos os usuários do computador. Uma vez, ainda sem sucesso, ela então utilizará uma lista com diversos usuários default para conseguir obter acesso ao sistema.

2.4.2 Reunião e manutenção

Em seguida, após ter conseguido seu primeiro objetivo - a contaminação do alvo - Schiller (2007) explicita que os novos clientes zumbis começam a se reunir e formar a grande teia em torno do servidor de Comando e Controle, que é o responsável pelo gerenciamento e atribuição de tarefas.

Conectado com o C&C, os zumbis podem requerer atualizações, como *update* de *exploits*, listas de IPs, listas de canais ou até mesmo de novos servidores de comando e controle. Essa atualização de servidores de C&C é de extrema importância, pois, no caso de falhas, indisponibilidades ou qualquer outra interrupção no comando, o nó não será perdido e automaticamente ficará sob domínio de outro servidor.

Logo em seguida, o arquivo infectado do cliente deve-se manter invisível e, para tanto, a *bot* solicita localização da ferramenta anti-antivírus mais atualizada para o controlador. Esse anti-A/V será capaz de remover o antivírus da estação ou, ainda melhor, torná-lo ineficaz a ponto de não identificar as atividades maliciosas nem conseguir obter suas atualizações.

Com as atividades ocultadas, o *botherder* tenta vasculhar na máquina vítima o que lhe pode ser útil, e, usando arquivos em lote, lança uma série de utilitários capazes de varrer e gravar informações dos usuários, bem como categorizar e agrupá-lo de acordo com a velocidade da rede, espaço no disco rígido ou velocidade do processador.

Uma vez garantido o cliente bot, cabe agora à estação vítima ficar em modo *listen*, ou seja, aguardando e ouvindo os comandos do controlador central.

2.5 Finalidades de uma *Botnet*

Configurar, deixar os serviços rodando e contaminando outras estações é somente um primeiro passo, pois as *botnets* podem ser usadas como plataformas para uma grande variedade de atividades criminosas, dependendo de como o *botherder* escolheu a configuração para cada nó. Assim temos como finalidades de uma *botnet*:

2.5.1 *Spamming*

De um modo geral, *spam* é qualquer mensagem eletrônica que foi recebida sem o consentimento do destinatário.

Desde os primórdios da mídia, a publicidade é ligada diretamente aos fatores econômicos, às vendas, à conscientização das pessoas ou quaisquer outras influências sobre produtos ou serviços (POSLUNS; SJOWERMAN, 2004). A internet evolui para revolucionar a maneira de fazer negócios, tornando-se um veículo de comunicação muito importante que viabilizou a globalização (ANTISPAM.BR, 2011). Houve então uma migração rápida e progressiva do marketing feito em rádio, jornal, televisão ou qualquer outro meio de comunicação para a internet. Foi nesse ponto que atitudes mal intencionadas começaram a nascer com a utilização de *spams*. Sendo a maneira perfeita de atingir milhares de clientes em potencial, de forma instantânea, sem limites geográficos ou de tempo, sem concorrentes e com um custo infinitamente menor.

Spam é um problema crescente enfrentado por todos os usuários de correio eletrônico em todo mundo e, atualmente, a maioria é alimentado através do uso das *botnets*, que permitem que os atacantes utilizem cada zumbi como motor de *spam* em massa (DUNHAM; MELNICK, 2009).

Assim, conforme preceituado (SCHILLER, 2007), tão logo seja identificado um spam de nada adiantará colocá-lo na *blacklist* ou entrar em contato com o ISP (*Internet Service Provider*), justamente pelo fato de que os *spammers* alugam ou compram botnet com o intuito de enviar as mensagens para múltiplos zumbis. Logo, perder um único nó não ocasionará prejuízo no fluxo de mensagens enviadas.

No primeiro relatório de ameaças da McAfee (2011a) foi reportado que o número de *spams* atingiu picos de cinco trilhões de mensagens diárias, porém esse número foi diminuindo aos poucos, até alcançar a média de um trilhão e meio. Essa redução significativa deveu-se principalmente à decapitação da botnet Rustock[2].

De fato, já no relatório seguinte (MCAFEE,2011b), os dados estatísticos continuaram em discreto declínio. Para a empresa, apesar da queda não ser significativa, tal redução foi favorecida pelo esforço conjunto das empresas de segurança, dos provedores e dos centros de estudos, tratamento e respostas de incidentes, bem como a aplicação mais vigorosa das leis.

Apesar dos números decrescentes, a McAfee relata que o desenvolvimento de *spams* vem evoluindo, tornando-os mais sofisticados, aumentando sua capacidade de ludibriar e invadir caixas de mensagens, como é o caso dos *spams* direcionados ou *spear phishing*[3].

O quantitativo enorme de *spams* que circulam na grande rede também é demonstrado pela Microsoft que, como proprietária dos sistemas operacionais da família Windows, conta com uma ferramenta de detecção e bloqueio de *spams* em seus sistemas, o FOPE - Microsoft Forefront Online Protection for Exchange – o qual foi possível observar uma diminuição de mensagens indesejáveis do ano de 2010 para 2011, tendo a média de 90 milhões de *spams* decrescida para 25 milhões por mês (MCAFEE, 2011a).

Assim como toda atividade criminosa, Posluns e Sjouwerman (2004) fazem um comparativo do mundo virtual com o real, mostrando que as tarefas envolvidas com os

[2] Rustock foi considerado a maior botnet de spam de todos os tempos, começou a operar em 2006 e teve seu declínio em 2011 a partir da ofensiva da *Microsoft Digital Crimes Unit* com a cooperação de especialistas das áreas industriais e acadêmicas.

[3] No *phishing* comum, e-mails são enviados para tentar "pescar" senhas e outros dados bancários de vítimas, já no *spear phishing* ("pescaria com arpão"), são enviados e-mails direcionados a alvos específicos, a mensagem é cuidadosamente montada para que se passe por verdadeira dentro de uma empresa ou órgão.

spammers são bem compartimentadas, onde cada integrante tem uma função ou responsabilidade específica.

Além do papel principal de *spamar*, existem os hackers que possuem a tarefa exclusiva de invadir sites para adquirir novas listas de contatos. Tais listas serão a moeda de troca entre eles, que poderá ser trocada ou vendida.

Ainda no comparativo, existe a figura do responsável pela propaganda, que é a pessoa que deseja que seu produto ou site seja promovido e para isso apela para tais atividades espúrias.

Esses spams cujo conteúdo é exclusivamente comercial são conhecidos como UCE – *Unsolicited Commercial E-mail* – porém, com o passar do tempo, eles deixaram de ter função de apenas propaganda e marketing e adquiriram novas funcionalidades como (ANTISPAM.BR,2011):

- **Spam Zombies**: são computadores de usuários finais que foram comprometidos por códigos maliciosos, tornando-os servidores de e-mails para o envio de spam;
- ***Chain letters* ou correntes**: é quando a mesma mensagem é enviada para várias pessoas e há um apelo para que cada usuário repasse para o maior número possível de outros usuários. A intenção, nesses casos, é obter o maior número de endereços eletrônicos válidos, favorecendo as atividades delitivas.
- Brincadeiras, ameaças e difamação;
- Pornografia, inclusive infantil;
- Boatos ou hoax, difamatórios ou de filantropia;

- *Spams* em redes de relacionamento.

2.5.2 *Phishing*

Enganar outras pessoas a fornecer suas senhas ou quaisquer outras informações confidenciais tem uma longa tradição nas comunidades hackers. De início, a atividade era realizada através de engenharia social, mas com a popularidade da internet, os atacantes automatizaram o processo e passaram a realizar ataques às grandes massas consumidoras (THE HONEYNET PROJECT, 2008a).

Phishing é um método no qual o atacante engana e rouba credenciais de usuários de internet, obtendo informações pessoais e financeiras. O atacante envia mensagens para os usuários, passando-se por uma instituição confiável e legítima, como um banco, empresa ou um site popular. As mensagens de *phishing* são, via de regra, enviadas em forma de *spam* através de *bots,* e direciona os usuários a sites falsos, porém muito idêntico ao verdadeiro, e induz o usuário a fornecer e enviar seus dados pessoais, inclusive senhas (MICROSOFT, 2011a).

O termo *pishing*, do inglês "*fishing*", é uma analogia feita pelos fraudadores em que os emails das vítimas representam iscas, e são utilizados para "pescar" os dados dos usuários, sobretudo financeiro.

Bambenek e Klus (2010) consideram que *phishing* é a consequência lógica do *spamming*, ou ainda, um spam avançado que tem a capacidade de induzir ao erro os incautos usuários, capturando os dados fornecidos por estes.

Em (FSTC,2012) foi realizado um estudo sobre as fases de um ataque de *phishing*, que consiste basicamente em planejamento, configuração, ataque, coleta, fraude e o pós-fraude.

O planejamento é a fase prévia em que são decididos os objetivos e selecionados os alvos, que podem ser pessoas, pequenas empresas até grandes instituições bancárias. Também serão definidas as credenciais (nome, nome dos pais, endereço, CPF, conta bancária, senhas e outros) que se pretendem capturar e o método que será usado na fase do ataque.

Na fase de configuração, o responsável pelo *phishing*, o *pisher*, cria matérias que farão partes dos e-mails ou sites maliciosos. Como já visto, estas matérias são sites falsos ou algum tipo de propaganda que tenha o objetivo de lesar o usuário de internet, fazendo, erroneamente, a digitar seus dados confidenciais. Nessa fase, também será obtida as listas de endereços eletrônicos das potenciais vítimas, bem como estabelecido as configurações da *botnet*, das páginas web, dos servidores, inclusive servidores proxy e demais serviços necessários para cometer a atividade ilícita.

O ataque é então realizado conforme o planejamento e as credenciais coletadas através de página web, respostas de e-mails, capturados por *malwares* baixados no computador da vítima ou até por telefone.

A próxima fase, da fraude, é a mais impactante, pois é nesta que as informações obtidas são transformada em dinheiro, bens ou serviços, ou seja, é onde as vítimas tem efetivamente o prejuízo financeiro.

Por fim, os *phishers* precisam encerrar as atividades, desligando o mecanismo de ataque, apagando evidências e evitando deixar pistas que os comprometam.

Os ataques de *phishing* são realizados utilizando-se de ferramentas e técnicas capazes de enganar usuários desavisados, e a estrutura que permite o golpe pode ser simples como uma página HTML inserida em um servidor web, geralmente comprometido, e um script que captura todos os dados inseridos pelos usuários, ou mais complexa, empregando técnicas de *pharming* ou envenenamento de DNS[4](HONEYNET PROJECT, 2008a).

O Projeto Honeynet (2008) também mostra a estreita relação entre *phishing* e *spamming*, uma vez que os *phishers* requerem um método que possa atingir um número máximo de vítimas, e, assim, eles têm encontrado a parceria ideal na forma de *spam* em suas atividades criminosas.

Abaixo, a figura representa uma captura de ataque de *phishing* realizado através de spam:

[4] *Pharming* é a técnica de ataque baseada no *DNS cache poisoning* ou envenenamento de cache DNS, pelo qual o criminoso, aproveitando brechas no domínio, injeta um endereço falso dentro do servidor DNS. Assim, se o usuário digitar o endereço de um site e o servidor DNS estiver envenenado, o endereço correspondente será apontado para uma página falsa controlada pelo fraudador.

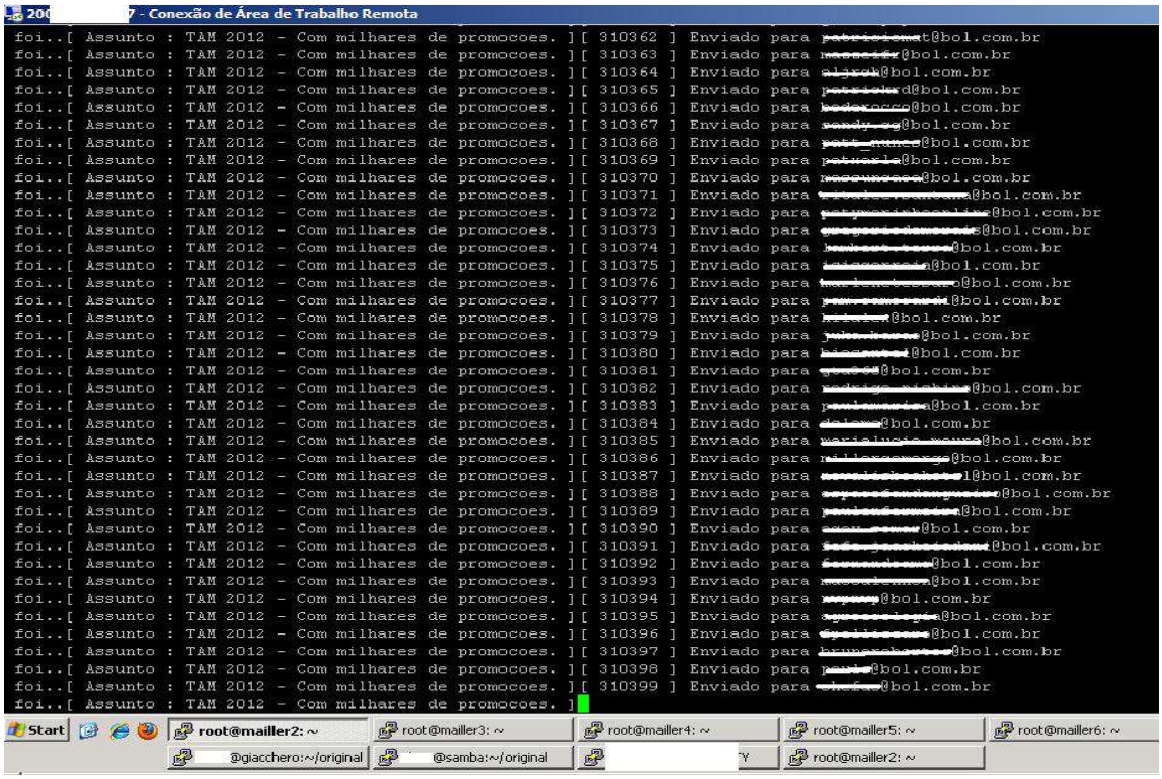

Figura 2.8 – Ataque de phishing

2.5.3 DDoS

Para a Microsoft (2011a) um dos mais antigos ataques realizados através de *botnet* foi justamente para realizar um ataque distribuído de negação de serviço, ou DDOS, do inglês *Distributed Denial of Service*.

Primeiramente cabe definir um ataque DOS, e para Nakamura e Geus (2007) um ataque de negação de serviços é quando recursos de computadores ou sistemas são agressivamente explorados, fazendo com que os usuários legítimos fiquem impossibilitados de utilizá-lo.

A Microsoft (2011a) continua e informa que, normalmente, tais ataques inundam os recursos disponibilizados aos usuários através de grande tráfego na rede, saturando sua largura de banda e, consequentemente, deixando indisponíveis os serviços.

Assim, a evolução natural de ataques do tipo DOS foram os ataques coordenados, ou também conhecidos por DDOS. E essa modalidade, explicada por Nakamura e Geus (2007), é quando diversos host distribuídos são atacados e coordenados por um hacker, realizando ataques simultâneos aos alvos. E isso resulta um ataque extremamente eficiente, dificultando a rastreabilidade da origem do ataque, visto que procedem de vários hosts que foram vitimados.

Howard (2009) e outros fazem uma importante observação ao informar que os ataques DOS, apesar dos bem sucedidos ataques no passado, tem pouquíssima eficiência atualmente, isto porque os servidores WEB atuais geralmente contam com grandes quantidades de armazenamento em disco e alto poder de processamento, além de dispor de largura de banda bem maior que antigamente. Assim, tem se tornado cada vez mais difícil um único computador obter êxito em um ataque de negação de serviço.

É nessa linha de raciocínio que entram as *botnets*, pois, mais uma vez, elas têm se tornado a aliada perfeita para o lançamento dessa modalidade de ataque. Howard (2009) considera que essa aliança provavelmente é a mais devastadora possível em um curto período de tempo, bem como a proporção do de danos causados com esse tipo de atividade.

Inicialmente, e como já dito, os ataques de negação de serviço visam interromper e privar os usuários legítimos de serviços ou recursos prestados por um sistema computacional, mas, além dessa intencionalidade imediata existem, conforme Howard (2009) e Dunham e Melnick (2009), algumas motivações mediatas:

- **Ganho financeiro**: como toda atividade criminosa, o principal ato motivador é o ganho financeiro. Assim, os ataques DDOS têm sido utilizados para extorquir ou chantagear, na medida em que ameaçam serviços online a menos que as empresas o paguem; terceirizar o ataque para outras pessoas que não detém o conhecimento; ou alugar o serviço e infraestrutura do ataque;

- **Travessura**: *script kiddies*[5] treinando ou se divertindo com as técnicas de ataques;

- **Vingança**: diversas empresas, sobretudo de anti-spam e antivírus foram alvos de hackers como forma de vingança e protesto;

- **Hacktivismo**[6]: promover um tumulto com intenção de expor suas ideologias políticas, sociais ou humanitárias.

- **Terrorismo**: o mundo atual não conheceu, de fato, uma cyber guerra. Os autores consideram que as atividades vivenciadas até hoje envolvendo afrontas do poderio de guerra cibernética foram fatos isolados. Assim, para eles, tal modalidade de ataques DDOS visando atos para fins terroristas foram, na verdade, uma espécie de hacktivismo político.

- **Competição**: empresas ou especialistas em segurança atacam seus rivais, visando demonstrar seu conhecimento e poder.

[5] Indivíduos inexperientes que, por não ter conhecimento avançado, utilizam-se de programas prontos para invadir sistemas, realizar ataques ou quaisquer outras formas de violar a segurança da informação.
[6] Designação das palavras hack mais activismo, a qual pessoas utilizam a informática para defender seus ideais políticos, filosóficos ou humanitários.

Nakamura e Geus (2007) consideram que os maiores responsáveis pelos ataques de negação de serviço são os próprios desenvolvedores de softwares, devidos aos *bugs* em serviços, aplicativos e sistemas operacionais. Deste modo, diversas falhas na concepção e implementação destes serviços, programas ou protocolos abrem brechas que são exploradas e aproveitadas para ataque em massa.

Howard (2009) classifica os ataques de negação de serviço coordenados em dois grandes grupos: o de esgotamento de largura de banda e o de esgotamento de recursos. O primeiro tenta subjugar o alvo com uma quantidade enorme de tráfego indesejado, impedindo, então, que as solicitações legítimas alcancem o host afetado. Já o segundo, como o nome sugere, tenta esgotar os recursos do sistema alvo, e estão intimamente ligadas às vulnerabilidades internas ou às configurações mal feitas ou simplistas de servidores.

Nesse contexto, o autor continua mostrando as principais modalidades utilizadas para realização de ataques do tipo DDOS:

- **User Datagrama Protocol (UDP) Flood Attack**: é um ataque de inundação em que os agentes enviam uma grande quantidade de pacotes UDP diretamente para a vítima. Como essa ação é executada por diversos agentes, a largura de banda do alvo não é capaz de prover todas as requisições.

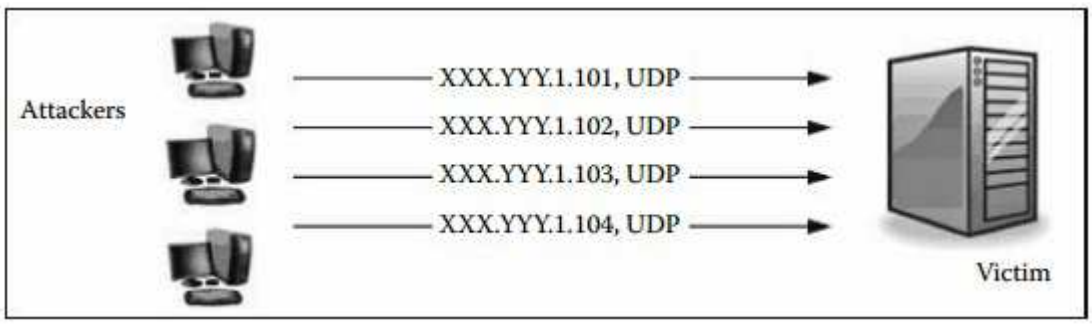

Figura 2.9 – UDP Flood Attack. Fonte: HOWARD, 2009

- **Ping Flood Attack**: devido à fragmentação de pacotes de IP, o qual permite especificar a quantidade máxima de dados que podem passar em um pacote através da rede física, um ataque de inundação de *ping* aproveita da forma de como a fragmentação e o reagrupamento desses pacotes foram implementadas. O método consiste em enviar pacotes ICMP com o MTU maior que o normal, fazendo com que os sistemas travem devido a grande sobrecarga do *buffer* da pilha TCP/IP. Para Dunham e Melnick (2009), atualmente essa forma de ataque caiu em desuso por causa das corretas implementações e atualizações nos sistemas, utilização de firewall, bem como modernas e mais bem sucedidas formas de ataque.

- ***Smurf e fraggle***: estes ataques são reflexivos, pois entre a vítima e o atacante existe a presença de um intermediário, que torna a identificação ainda mais difícil, tendo em vista que o tráfego que chega ao alvo parte do intermediário e não do atacante. O *smurf* e o *fraggle* consistem em enviar uma grande quantidade de pacotes ping, ICMP *echo* e UDP *echo* respectivamente, para o *broadcast* da rede, como se o endereço de origem fosse o IP da vítima através de outra técnica, o IP *Spoofing*[7]. Então, cada host receberá a requisição e responderá para o endereço da vítima.

[7] IP Spoonfig é a técnica que mascara o endereço de um atacante. Nos ataque DDOS, os pacotes com as respostas são direcionadas para o endereço da vítima, que foi forjado pelo atacante.

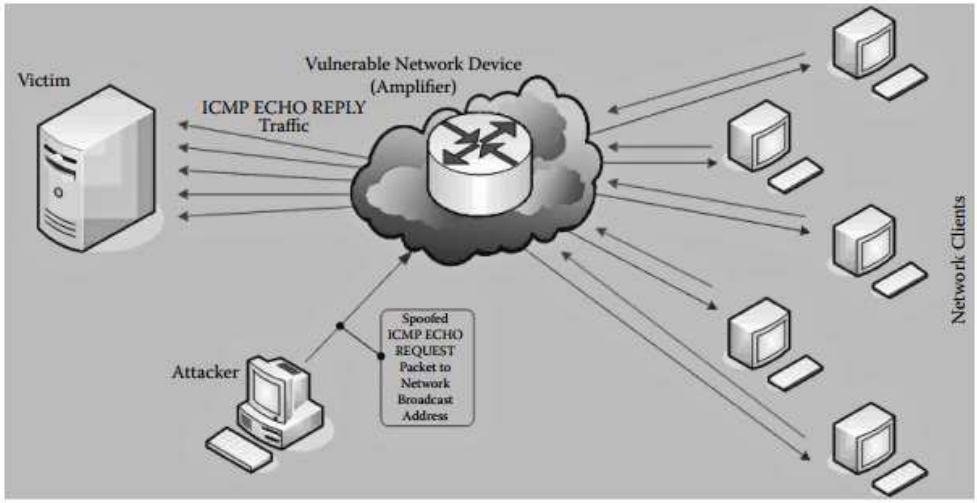

Figura 2.10 – Smurf e Fraggle. Fonte: HOWARD, 2009

- **Syn Flood Attack**: uma das formas de estabelecimento de conexão, realizada na camada de transporte, é o *handshake* de três vias, e, conforme explica Tanenbaum (2003), duas estações confirmam uma a outra que está pronta para iniciar a transmissão, sincronizando o número de sequencia entre elas através dos pacotes Syn e Ack. Então, conforme é esclarecido por Nakamura e Geus (2007), esse ataque gera um grande número de pedido de conexão (pacotes syn), não conseguindo o servidor atender a todas as demandas. Daí, um *overflow* ocorre na pilha de memória, fazendo com que as requisições legítimas não sejam atendidas.

- **Push and Ack Atacks**: semelhantes à inundação de pacotes *syn*, os atacantes enviam pacotes TCP com os bits do *push* e *ack* setados com valor um. Os pacotes setados desta forma instrui que o sistema da vítima descarregue os dados do *buffer* TCP, mesmo que este não esteja cheio. Com a repetição deste processo, o sistema não consegue processar o grande volume de entrada, o que ocorrerá a negação de serviço.

- **Land Attacks**: atualmente este ataque é ineficaz em sistemas atualizados, porém, devido à simplicidade, foi muito explorado. Ele consistia em criar um pacote IP especial em que o endereço de origem era o mesmo de destino, fazendo com que o computador ficasse, continuamente, respondendo a si mesmo.

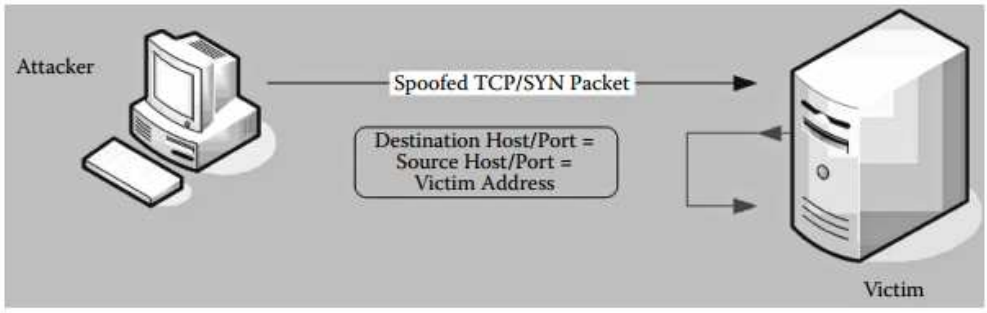

Figura 2.11 – Land Attack. Fonte: HOWARD, 2009

2.5.4 Instalação de *Malware*

O Relatório de Segurança da Microsoft (2011a) explica que é comum que as *botnets* também sejam utilizadas para realizar automaticamente downloads de *malwares* para as estações das vítimas. Deste modo, o intuito dos malfeitores é forçar a instalação de *adwares*, *spywares* ou quaisquer outros softwares potencialmente indesejáveis, visando rapidamente obter lucros.

Umas das funcionalidades da instalação de tais códigos servem para campanhas publicitárias, como o caso do *pay-per-click* ou link patrocinado, o qual a matéria veiculada na internet é paga quando tem seu anúncio clicado. A *botnet*, então, faz com que as unidades

comprometidas aumentem artificialmente o contador de cliques, gerando lucros para seus criadores.

De um modo geral, os *malwares* instalados pelas *botnets* trabalham silenciosamente, evitando alertar ao usuário que o computador está infectado. Porém, em outros casos, algumas botnets forçam o usuário a baixar e executar falsos programas disfarçados de *antimalwares* legítimos, e, consequentemente, exibe alertas de infecções inexistentes e a promessa de removê-los caso o usuário compre a versão completa.

Ainda, para a Microsoft, a instalação de códigos maliciosos é tão drástica, que ela pode, inclusive, instalar outras famílias de *bot*, tornando-a mais devastadora.

2.5.5 Roubo de Dados Confidenciais

Dunham e Melnick (2009) ponderam que informação e o conhecimento é poder, e, para os criminosos, vale dinheiro. Então, muito *bots* têm sido instruídos para vasculhar nos computadores pessoais quaisquer informações que sejam relevantes e que possam ser usadas de alguma forma para obter lucro.

Tais informações podem ser nomes de pessoas, usuários, lista de e-mails, credenciais, números de contas bancárias e senhas, trilhas de cartões de crédito, chaves de produtos ou qualquer outro dado sensível que deveria ser guardado em sigilo. Tal contexto pode também ser aplicado em grandes empresas, para fins de espionagem e segredos comerciais, tornando problema ainda maior.

Os autores ainda alertam que as *botnets* mais modernas têm a capacidade de igualmente vasculhar o tráfego da rede, permitindo que um *bot* consiga realizar um *eavesdrop*[8], podendo mapeá-la, conhecer serviços, programas e versões.

2.6 Estratégias de C&C

Como bem pronunciado por Schiller (2007), "Controlar um computador é relativamente fácil. Controlar mil computadores torna-se um pesadelo logístico". Daí a extrema necessidade de existir alguma forma automatizada de controlar os computadores vítimas.

2.6.1 IRC C&C

Há tempos as botnets utilizam a tecnologia de IRC para exercer as atividades de comando e controle, pois, além de ser robusta, possui certas características desejáveis, tais como:

- Bastante interativa, e a simplicidade de seu protocolo, com comunicação full-duplex, permite uma interação dinâmica entre ambas as partes, estilo cliente-servidor;
- Facilidade de criação e uma diversidade de servidores que suportam o protocolo;

[8] *Eavesdrop é uma técnica de hacking, baseada na espionagem, que consiste em violar a confidencialidade, capturando pacotes nos segmentos de rede. Muito semelhante ao grampo telefônico.*

- Facilidade de controlar as *botnets*, inclusive quando mais de uma botnet é gerenciada em um mesmo servidor;
- A redundância, garantindo a continuidade da operação, pode ser conseguida com um simples link entre servidores.

Quando uma infecção ocorre, o novo *bot* informa para o controlador, confirmado o estado de contaminação. Esse aviso pode ser através de uma mensagem privada ou quando automaticamente o zumbi adere a um canal de chat. Schiller (2007) exemplifica que essa mensagem consiste em algo do tipo: “Oi. Estou aqui, senhor. Meu IP é 127.0.0.1 e estou escutando na porta 666!”.

Como já dito anteriormente no decorrer do trabalho, *botnet* é uma analogia de um exército de computadores infectados que recebem ordens de um comando central. Então, apesar da facilidade de controle de *botnets* com servidores IRC, um ponto fraco de seu uso seria justamente o fato dele ser centralizado, conforme modelo da figura abaixo. E, como num exército real, esse nó central pode ser passível de um ataque ou interrupção, ficando bloqueado para exercer suas atividades.

De início, quando um C&C sofria uma paralisação, toda a botnet praticamente parava de atuar. Na prática, o controlador não mais conseguia emitir instruções nem saber quais *bots* estavam sob seu domínio.

Depois, *botnets* mais especializadas inovaram com controles alternativos. Um tipo de controle podia vir na própria contaminação inicial, onde um *backdoor* já liberava uma porta TCP, permitindo uma ligação remota e a recuperação do controle.

Em outros casos, a unidade de Comando e Controle possui seu próprio backup, e, caso a comunicação seja perdida, os *bots* restabelecem a comunicação automaticamente com o *botherder*.

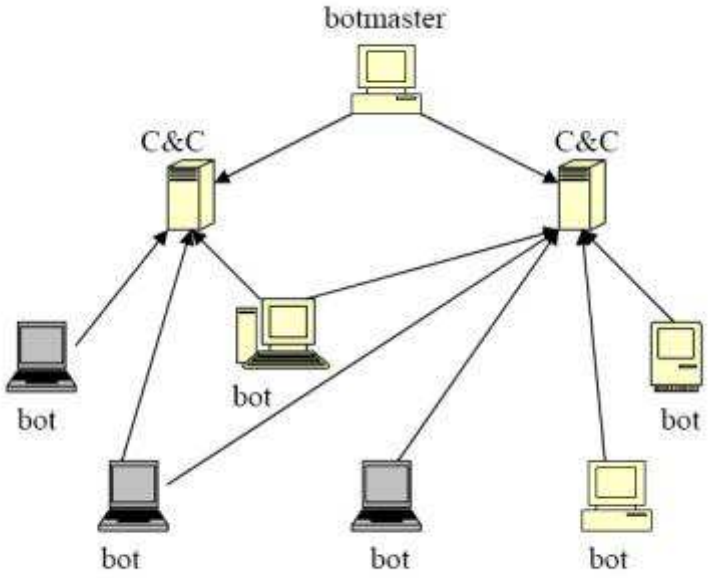

Figura 2.12 – Modelo centralizado de controladores de *botnet*. Fonte: WANG, 2012

Ocorre que a extrema simplicidade do protocolo permitiu, com o tempo, que *botnets* concorrentes tomassem para si o controle dos *bots*, mudando suas características ou mesmo a desmontassem. Por conseguinte, o próprio meio fez nascer novas tecnologias de controle de *botnets*, substituindo os servidores de IRC ou completando-os.

2.6.2 Web C&C

Depois do uso do IRC como controlador, o mais comumente tipo de C&C usado são os servidores web. A diferença básica é o canal de controle que é um protocolo muito diferente que o utilizado em servidores IRC.

O relatório de inteligência da Microsoft (2011a) pondera que a utilização de HTTP como um controlador aumentou muito nos últimos anos, embora eles ainda sejam responsáveis por menos infecções do que os *bots* de IRC, como pode ser visto no gráfico da figura 2.13.

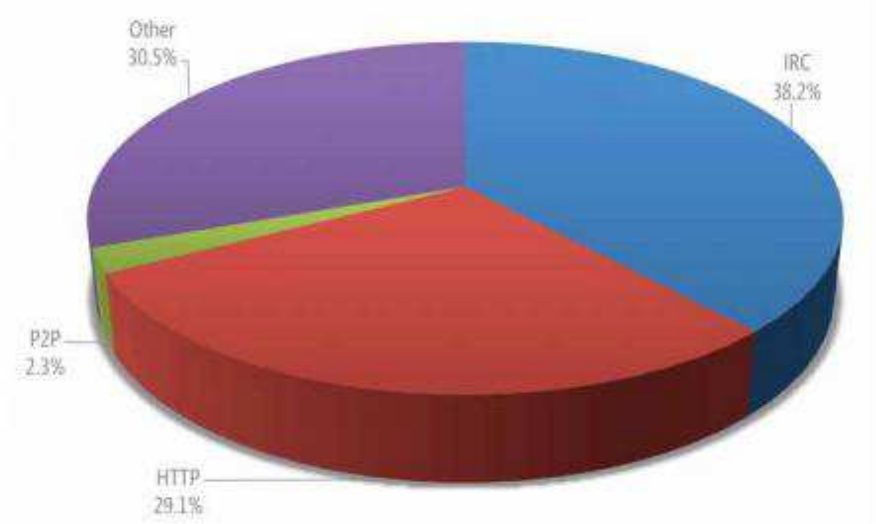

Figura 2.13 – Quantitativo médio dos mecanismos de C&C. Fonte: MICROSOFT, 2011a

Schiller (2007) dividiu em dois tipos os controladores baseados em web: o *echo* e o comando.

No echo, simplesmente é anunciada a presença do novo *bot* para o comando central. Esse aviso se realiza de algumas maneiras:

- **Conectar e esquecer**: o *bot* somente se conecta ao sistema. Já ao controlador cabe registrar essas conexões e seus endereços IP que, geralmente, são feitas em arquivos de logs. Em alguns casos, os conhecidos contadores de visita em sites são utilizados para este fim.

- **Arquivo de dados**: semelhante ao anterior, porém os servidores possuem arquivos com uma lista de instruções que serão baixadas pelo cliente quando de sua conexão. Ao invés de uma lista de instruções pode existir também um arquivo executável que, da mesma forma, será baixado e instalado.

- **Dados de URL**: nesse caso, o servidor recebe do cliente um endereço que conterá informações imprescindíveis como porta que será usada pelo backdoor ou senha para acessar o bot.

Já a baseada em comandos consiste em um complemento a qualquer outro tipo de *botnet*, ajudando o comando central a gerenciar seu exército. De um modo geral, são interfaces gráficas em que o C&C emitirá instruções aos zumbis. Nesse caso, os controladores que se conectam ao *bots*, e não o contrário, como na situação anterior em que os clientes se conectavam ao seu comandante. Esses comandos permitem, dentre outros, as seguintes tarefas:

- Realizar downloads e os execute nas estações alvos;

- Transferência de arquivos;

- Executar comandos Shell;

- Realizar screenshots;

- Bloquear urls;

- Alterar o arquivo host, fazendo com que usuários sejam redirecionados para sites maliciosos, em vez do que ele pretende de fato navegar;

- Algumas interfaces gráficas mais modernas permitem escolher *bots* por país, por ISP, por largura de banda, bem como coletar dados estatísticos.

Dunham e Melnick (2009) asseveram a facilidade e importância do surgimento das *botnets* controladas por servidores web, onde existe a possibilidade de comprar, inclusive, toolkits por poucos dólares preparadas para realizar um ataque. E tais ferramentas prontas podem incluir *exploits*, códigos maliciosos não detectáveis por antivírus e mecanismos para vasculhar, copiar e classificar dados roubados, sendo também intuitivo e muito fácil de usar.

Continuando, os autores fazem um contraponto, mostrando as principais vantagens da utilização de controladores em servidores web sobre os servidores de IRC:

- Como a porta TCP 80 é quase sempre liberada para o tráfego normal na rede, o tráfego de botnet passa geralmente despercebido;
- Os zumbis de servidores web não requerem uma conexão constante, pois estes informam quando estão online. Consequentemente, o tráfego de rede entre o zumbi e o nó central será bem menor, dificultando ainda mais a detecção de atividades irregulares;
- A escalabilidade de um serviço web é maior que a capacidade de computadores que as salas de bate-papo conseguem gerenciar;
- Quanto à usabilidade, as redes zumbis baseadas em servidores web são mais fáceis de serem operadas por um *botherder* médio, pois não exigem uma programação mais apurada ou personalizada.

A Microsoft (2011a) também compartilha da facilidade que o HTTP proporcionou, pois é o protocolo primário para navegação em rede, dificultando a detecção do tráfego ilegal de uma *botnet*. Apresenta ainda que o HTTP é muito usado pelas redes zumbis para baixar atualizações e outros *malwares*, independentemente do mecanismo de C&C, pois

aquelas já contam com seus próprios servidores web para hospedar sites de *phishing* ou qualquer outro conteúdo ilegal.

Em (DUNHAM; MELNICK, 2009) é citado ainda as precauções que os *botherder* devem ter ao configurar o *bot*, pois, a comunicação com um servidor web baseado em um único endereço IP ou a um domínio específico é facilmente mitigada, tendo em vista que o endereço pode ser facilmente bloqueado e o ISP pode ser notificado para encerrar o comportamento malicioso daquele IP identificado.

Para tanto, concluem que existem alguns procedimentos ou técnicas para dificultar a investigação da atividade maliciosa e postergar um possível *shutdown* no sistema de bot. A idéia básica, seria primeiramente configurar um domínio para ter vários IPs, técnica conhecida como *multihomig*, e caso algum endereço seja bloqueado, outro automaticamente assume o lugar. Outra situação, que por sinal já tornou prática atualmente, consiste em controlar uma *botnet* com outra *botnet*. Então, botherder pequenas, de no máximo 20 *bots*, emitem comandos para *botherders* maiores, dividindo o controle e consequentemente, dificultando a identificação do *botherder* master.

2.6.3 Peer-to-Peer C&C

Primeiramente, conforme define Tanenbaum (2003), redes *peer-to-peer* ou P2P é um sistema não hierárquico, totalmente distribuído, onde cada nó é simétrico e não existindo nenhum tipo de controle ou hierarquia entre eles. A vantagem principal desse modelo é uma variedade de recursos, como armazenamento, processamento, largura de banda, resistência às falhas e uma ótima escalabilidade.

Em (MICROSOFT,2011a) é exposto que, historicamente, a quase totalidade das *botnets* possuía mecanismos de controle centralizado, porém algumas famílias de botnets, atualmente, têm adotado a tecnologia de *peer-to-peer*. Deste modo, os criminosos reforçam o anonimato, fugindo do controle e resistindo a possíveis desligamentos do sistema.

Como visto nos mecanismos de comando e controle anteriores - sobretudo nos mais antigos que não contavam com recursos de redundância, backup, DNS dinâmico - os servidores de C&C são basicamente o único ponto de falha. Ao passar o controle central para arquitetura P2P, a rastreabilidade dos cabeças da organização e a averiguação do tamanho real da rede maliciosa torna-se tarefa mais complexa.

Dunham e Melnick (2009) informam que as redes P2P chamaram atenção dos *botherders* logo após a grande popularidade alcançada com os serviços de músicas e compartilhamento de arquivos pelo Napster[9], permitindo a criação de redes privadas P2P com o fim específico de controlar *bots*, gerando um modelo eficiente de gerenciamento sem um único ponto de falha (figura 2.14).

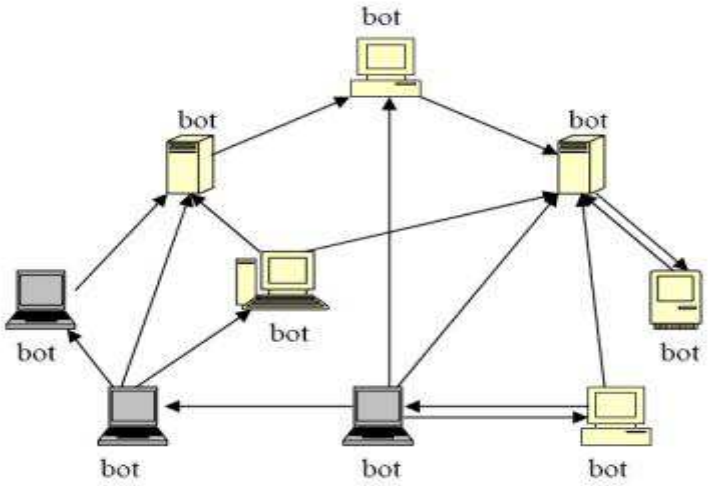

Figura 2.14 – Modelo P2P de controladores de *botnet*. Fonte: WANG, 2012

[9] Programa de compartilhamento de arquivos, sobretudo música em formato mp3, em redes P2P, que permitia aos usuários efetuar um download direto da máquina de outros usuários, tudo de forma descentralizada, pois cada usuário conectado exercia tanto função de cliente como servidor.

Com esse modelo desenvolvido, é possível ao *botherder* o controle de toda a *botnet* a partir de qualquer nó (vítima) da rede P2P. O *botherder*, por exemplo, a partir de uma simples atualização em um único nó pode gerar uma reação em cadeia, onde cada outro zumbi dentro da rede P2P também sofre uma atualização, sincronizando um ao outro com as novas configurações injetadas pelo *botherder*.

Schiller (2007) aponta que um ponto negativo nos controladores em P2P é que de qualquer ponto da rede pode ser injetado algum tipo de programação, visando seqüestro da rede, monitoramento ou investigação, podendo-se, inclusive, descobrir toda a rede de *bots*. Já em (DUNHAM; MELNICK, 2009) é acrescentado que para mitigar tais problemas, os criminosos utilizam criptografia e tem seus códigos injetados na rede com assinatura digital.

Wang e outros (2012) esclarecem que apesar da ideia geral de total descentralização nas redes P2P, existem modelos que contam com um servidor central, que mantém metadados, informações de roteamento, processos de pedidos de buscas de arquivos e coordenadas de controle de downloads entre os pares. Todavia, esse serviço central não participa efetivamente dos downloads, mas constitui um ponto de falha.

Para superar esse problema, o autor continua informando que as botnets mais sofisticadas preferem usar arquiteturas P2P puramente descentralizadas, onde cada nó, de fato, cumpre seu papel de servidor, quando do processamento de uma consulta de arquivo por outro par, ou como cliente, quando este solicita uma consulta, ou seja, tudo sem controle ou intermediários.

Então, contando com o apoio dessas redes P2P sem uma autoridade central, que não consegue garantir que arquivos compartilhados não são maliciosos, os *botherders* encontraram um local confiável para armazenar seus softwares maliciosos, e, principalmente, poder espalhá-los.

De tal modo, uma vez contaminado, o nó comprometido prossegue tentando infectar outros nós. E essa contaminação é baseada numa lista que, segundo Wang e outros (2012), contém a relação dos usuários que solicitaram algum contato anterior ou responderam alguma consulta de arquivo. Depois, de um modo geral, o *worm* P2P residirá no mesmo diretório local de compartilhamento de arquivos, podendo ter um nome sugestivo ou popular, e esperará que outros pares o executem e promovam a contaminação.

2.6.4 Outros controladores

Schiller (2007) ainda apresenta outras formas para gerenciar uma rede zumbi, que apesar de tímidas veem tornando-se comum:

- **Comunicadores Instantâneos**: na verdade, ele atua como uma tendência em auxiliar outras botnets com controladores diversos. Da mesma forma, após a contaminação, o worm utiliza as contas de usuários dos programas mensageiros, como MSN, GTalk, ICQ ou Yahoo, para se comunicar com a central. O autor resalta que, ao contrário dos controladores em servidores IRC, os comunicadores instantâneos são controlados por grandes empresas, o que facilita sua detecção e filtragem de tráfego, não sendo eficaz para gerenciamento de grandes redes zumbis.

- **Ferramentas de administração remota**: alguns programas com PCAnywhere, RealVNC, TeamViewer ou ferramentas do próprio sistema operacional como o *Remote Desktop Services* – ou *terminal services* – da Microsoft, tem sido encontrados em computadores comprometidos. Porém, tal técnica precisa de

gestão contínua e exclusiva, estação por estação, o que implica em uma dificuldade gigantesca de administração;

- **Servidores FTP**: apesar de pouco comum na prática, o protocolo também foi experimentando como um canal de controle de *botnets*. Tal tecnologia foi utilizada juntamente com algumas formas de cavalos de tróia bancário, que ficavam em modo *listen* aguardando que o usuário acessasse a zona HTTPS, e, em seguida, além de apresentar um site falso, capturava as credenciais – o *phishing*. Os dados privados roubados, então, são enviados para um servidor FTP sobre domínio do *botherder*.

Por fim, em (SCHILLER, 2007) é esclarecido que independentemente do tipo de mecanismo de controle que uma *botnet* faça uso, a prática comum é que todo C&C faça uso de alguma forma de criptografia, autenticação ou ofuscação para limitar a capacidade dos outros de espionarem as comunicações.

2.7. IRC e Botnets

Charalabidis (2000) define IRC como uma simples forma de comunicação, em tempo real, entre milhares de usuários em todo mundo. Porém, como resalva, ele pode se tornar uma ferramenta extremamente complexa para diversos outros fins.

Essa simplicidade consiste basicamente de dois programas, um programa-servidor que aceita conexões e um programa-cliente que requisitará a conexão ao servidor. Já os servidores podem se conectar um aos outros, aumentando a capacidade de se comunicar com um grande número de usuários através do compartilhamento de canais comuns de conversação.

Após um longo período de popularidade, sobretudo do cliente de IRC para Windows – o mIRC – o protocolo de comunicação encontrou sua decadência por volta de 2003, quando os mensageiros instantâneos se tornam mais populares, oferecendo recursos mais inovadores e atraentes. A partir de então, nos tempos atuais, a utilização do IRC, além da já conhecida aplicação em *botnets,* ficou para fins mais específicos, como para troca de arquivos, para prover suporte aos usuários de algumas distribuições Linux ou compartilhar conhecimento e troca de idéias em alguns projetos de software-livre (RFC, 1993a).

2.7.1 Protocolo IRC

O protocolo IRC é aberto e foi desenvolvido para ser utilizado com o protocolo TCP/IP – mas não necessariamente somente ele –, e, opcionalmente, pode ser adicionado SSL, criando um túnel que encapsula os dados do início ao fim (RFC, 2012a e IRC.ORG,2012).

Ao longo do tempo o protocolo foi definido em algumas RFC, a primeira foi a RFC 1459. Atualmente, a arquitetura foi redefinida na RFC 2810. O gerenciamento de canal está presente na RFC 2811, o cliente na RFC 2812 e, por fim, o lado do servidor foi definido na RFC 2813 (RFC, 2012b).

Os servidores formam a espinha dorsal da estrutura de IRC, sendo o ponto central onde convergem e se conectam os usuários para conversar com os demais usuários, é também o ponto onde outros servidores se conectam, formando toda a rede de IRC.

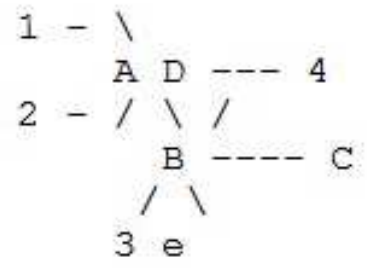

Figura 2.15 – Arquiterura IRC – Servidores A,B,C,D e E, Clientes 1, 2, 3 e 4. Fonte: RFC, 2012b

O protocolo IRC também é baseado em texto, isso permite que visualmente perceba-se o que está ocorrendo, assim, com um cliente telnet ou netcat é possível obervar o comportamento ao nível de protocolo, linha a linha (IRC.ORG, 2012).

2.7.2 Funcionamento básico do IRC

Independentemente do Sistema Operacional usado, o funcionamento padrão do uso de um chat tipo IRC é primeiramente escolher o aplicativo cliente, em seguida, escolher um apelido, ou *nick*, e, por fim, o servidor e seu respectivo canal.

Um servidor pode ter tão somente um canal, como milhares de canais habilitados ao mesmo tempo. Em geral, o nome do canal reflete a natureza das conversas, e cada canal tem seu tópico, que, em geral, é dinâmico e é uma breve descrição do que se pretende no chat, conforme Mutton (2004).

Os nomes de canais são precedidos de uma cerquilha (#), e para acessá-los basta um duplo clique ou o comando /join <nome do canal>. Como pode haver uma infinidade de canais dentro do servidor, o comando /list varrerá o servidor e exibirá todos os canais públicos disponíveis, bem como a quantidade de usuários e sua descrição.

Dentro dos canais, além dos usuários comuns tem os usuários com alguns poderes privilegiados, são os operadores, também chamado de OP ou IRCops, e seu nome vem precedido do símbolo “@”. O operador pode ser o criador do canal ou simplesmente obteve a função de moderador, cuja função principal é manter a organização do chat, banindo, se necessário, usuários que desrespeitam as regras.

2.7.3 IRC *Bots*

Conforme exposto no início do trabalho e também ratificado por Charalabidis (2000), IRC bots são programas ou *scripts* que foram criados para servir como ferramenta de manutenção e gestão de canais, porém nem toda variabilidade de uso tem fins inocentes. É, então, nesse ponto que se teve início o que hoje conhecemos por *botnets*.

Tecnicamente e visualmente, o *bot* é um cliente, e, como tal, ele permanece conectado e listado como se fosse um usuário comum, porém vigiando os eventos e seguindo apenas as instruções que lhe foram atribuídas, mesmo que seu proprietário não esteja online. A diferença básica entre um *bot* e um usuário comum é que aquele passa muito tempo conectado, sem dizer nada ou limitando-se às respostas automáticas.

Os *bots*, como clientes especiais, correm em segundo plano, e são controlados através de comandos pré-configurados por meio de um cliente de IRC comum. Sua resposta pode ser tanto mensagens para outros usuários ou saída de escrita em arquivos. Mas a sua

característica primordial é fornecer alto nível de automação tanto para tarefas tediosas como as que exigem desempenho rápido (WANG et al., 2012).

2.7.4 IRC *Botnets*

Em geral, como todo acontece em todo tipo de tecnologias, existem sempre aquelas pessoas que abusam, explorando todas as formas maliciosas para obter algum tipo de vantagem. Com o modelo IRC também não foi diferente, pois os IRC *bots* foram desenvolvidos até chegar ao conceito de IRC *botnets* (MAYNOR et al, 2006).

Após a exploração bem sucedida, conforme visto no tópico 2.4.1, a *botnet* usa algum protocolo, como FTP, HTTP ou o CSend do próprio IRC, para se transferir para a estação comprometida. O arquivo binário transferido é automaticamente iniciado e logo tenta se conectar com o servidor mestre. Em (HONEYNET PROJECT, 2008a) é explicado que essa conexão, geralmente, se dá com um DNS dinâmico, de modo que caso ocorra alguma falha, o bot é facilmente realocado.

Em seguida, com algum apelido especial que o identifique facilmente, o *bot* tem acesso ao canal do *botherder*, que, via de regra, é acompanhada de senha (HONEYNET PROJECT, 2008b) ou, como nos modelos mais atuais, criptografia (SCHILLER,2007). Abaixo, a figura ilustra uma comunicação típica após uma infecção bem sucedida.

```
<- :ircl.XXXXXX.XXX NOTICE AUTH :*** Looking up your hostname...
<- :ircl.XXXXXX.XXX NOTICE AUTH :*** Found your hostname
-> PASS secretserverpass
-> NICK [urX]-700159
-> USER mltfvt 0 0 :mltfvt
<- :ircl.XXXXXX.XXX NOTICE [urX]-700159 :*** If you are having problems connecting due to ping
timeouts, please type /quote pong ED322722 or /raw pong ED322722 now.
<- PING :ED322722
-> PONG :ED322722
<- :ircl.XXXXXX.XXX 001 [urX]-700159 :Welcome to the ircl.XXXXXX.XXX IRC Network
[urX]-700159!mltfvt@nicetry
<- :ircl.XXXXXX.XXX 002 [urX]-700159 :Your host is ircl.XXXXXX.XXX, running version Unreal3.2-
beta19
<- :ircl.XXXXXX.XXX 003 [urX]-700159 :This server was created Sun Feb  8 18:58:31 2004
<- :ircl.XXXXXX.XXX 004 [urX]-700159 ircl.XXXXXX.XXX Unreal3.2-beta19 iowghraAsORTVSxNCWqBzvdHtGp
lvhopsmntikrRcaqOALQbSeKVfMGCuzN
```

Figura 2.16 – Conexão típica de um bot. Fonte: HONEYNET PROJECT, 2008b

O servidor, então, passa a reconhecer o *bot* como um novo cliente no IRC e, após a troca de alguns pacotes, o *bot* entrará no canal configurado e com a senha pré-estabelecida, como por exemplo “JOIN #canalbot <senha>”.

Maynor et al. (2006) afirmam que uma *botnet* IRC pode ser controlada de várias formas, a mais tradicional é através do tópico do canal. O *bot* receberá o tópico do canal e interpretará o comando. Abaixo, a figura mostra o comando recebido do tópico.

```
<- :ircl.XXXXXX.XXX 332 [urX]-700159 #foobar :.advscan lsass 200 5 0 -r -s
<- :[urX]-700159!mltfvt@nicetry JOIN :#foobar
<- :ircl.XXXXXX.XXX MODE #foobar +smntuk channelpassword
```

Figura 2.17 – Tópico de um canal de uma Botnet IRC. Fonte: HONEYNET PROJECT, 2008b

Esse tópico ordenará ao *bot* que se propague através de alguma vulnerabilidade, nesse caso o processo lsass do Windows, com 200 sessões concorrentes, delay de 5 segundos, por tempo indeterminado, verificação randômica e silenciosa, o que evita maiores tráfegos. Nessa busca descomedida, cada máquina vulnerável encontrada na pesquisa poderá ser automaticamente contaminada, contribuído para o crescimento da rede (MAYNOR et al, 2006).

Outro exemplo citado no Projeto Honeynet (2008b) é o comando "Http.update http:// <servidor>/~ mugenxu /rBot.exe c:\.Msy32awds.exe 1", o qual o *bot* será instruído a fazer o download de um arquivo binário, executando-o, em seguida, de acordo com o parâmetro “1”. Logicamente, o tópico do canal pode também não conter comando algum ou apenas um breve comentário, o que fará que os *bots* fiquem apenas escutando, aguardando novas ordens.

Uma segunda forma comum de controlar os *bots* no IRC é através do envio de mensagens pelo próprio canal. Apesar de efetivo, esse método é identificado através de uma simples visualização no corpo do canal. Os comandos são semelhantes às ordens estabelecidas no controle por tópicos, com o diferencial que qualquer usuário que entrar no chat poderá facilmente monitorar as ações do *botherder*.

Outro método comum, mas contra produtivo, é o envio de mensagens particulares para cada *bot* comandado. Nesse caso, a efetividade está na falta de percepção do que está acontecendo no canal. Assim, ela é mais utilizada quando os *bots* assumem necessidades especiais com comandos personalizados, onde cada nó age como uma entidade separada (MAYNOR et al, 2006).

Dentre os modelos de IRC Botnets que foram desenvolvidos, a RxBot teve seu destaque pelo inúmeros casos de eficiência e contaminação e será analisada no próximo capítulo.

3. A BOTNET RXBOT

A RxBot, também conhecida por Rbot, é um worm escrito em linguagem C++ (PATIL, 2009), foi reportada pela primeira vez em 2003, e até 2006 já contava com mais de 1,9 milhões de máquinas infectadas, figurando em primeiro lugar em seu ranking de contaminação, de acordo com a tabela comparativa da equipe antimalware da Microsoft (2006) .

A Rxbot já foi uma das mais difundidas e complexas *botnet* existente desde 2003, e, atualmente, ele continua impulsionando a funcionalidade principal de diversas outras variantes (SCHILLER, 2007).

3.1 Características da RxBot

Patil (2009) enumera as principais características da família Rxbot:

- Programação modular;
- Seus alvos são os Sistemas Operacionais Microsoft Windows;
- Para comunicação com os bots, há suporte para canais de IRC protegidos por senha;
- Possui recursos para port scanning, packet sniffing e key logging;
- Conta com multi thread, orientação a objetos e códigos polimóficos;
- Utiliza o protocolo SMTP para enviar spam e espalhar suas próprias cópias;
- Utiliza protocolo HTTP para consecução de fraudes e ataques DDos.

3.2 Inicialização

A Rxbot usa métodos diferentes para buscar vulnerabilidades e sistemas para infectar. Ela tenta explorar senhas ou políticas de segurança fracas, através uma lista padrão ou uma lista pré-configurada pelo *botherder* de senhas, pelo método da força bruta. Assim, senhas simples ou em branco são presa fáceis para esta família de worms (SCHILLER, 2007).

Ela também pode realizar um escaneamento nas principais vulnerabilidades de software, inclusive explorando *backdoors* ou portas abertas por outros arquivos maliciosos.

Os nomes dos arquivos maliciosos podem variar muito de uma variante para a próxima, tornando sua identificação difícil. Mas em (SCHILLER, 2007), o autor esclarece que as empresas antivírus não possuem uma nomenclatura padrão, e exibe os principais apelidos utilizados pelos fornecedores:

- McAfee: W32/SDbot.worm.gen.g
- Symantec: W32.Spybot.Worm
- Trend Micro: Worm_RBot
- Kaspersky: Backdoor.RBot.gen
- CA: Win32/RxBot

Consultando os sites oficiais das empresas de antivírus, podemos obter uma lista completa e dinâmica dos arquivos maliciosos e suas variantes.

3.3 Sinais de comprometimento

Após a contaminação, conforme Schiller (2007), a Rxbot é transferida para o diretório %system% – ou c:\windows\system32 – , este arquivo, primeiramente, era batizado de *wuamgrd.exe,* tendo suas variantes utilizado diferentes nomes de arquivos. Já as variantes mais modernas conseguem alterar dinamicamente o nome do arquivo, de modo que cada sistema infectado terá um nome de ficheiro diferente.

Ainda, para dificultar a identificação e o contra ataque, o arquivo é copiado como somente leitura, oculto e tem sua data e *timestamp* alterados, fazendo com que, caso um usuário perceba algo estranho, a aparência será de um arquivo antigo e previamente instalado no sistema operacional.

Para garantir que o bot seja executado automaticamente a cada inicialização do sistema operacional, a Rxbot adiciona algumas entradas nas seguintes chaves do registro do Windows, podendo, inclusive, verificar periodicamente se tais valores não foram alterados ou excluídos, restaurando-os (MICROSOFT, 2012d):

- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices
- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServices

A manutenção do bot depende também da possibilidade do malware está programado para matar alguns processos associados com programas de segurança e antivírus, evitando ser detectado e consequentemente removido. Da mesma forma, ela realiza busca e

tenta encerrar malwares concorrentes. Os processos encerrados são, por exemplo, regedit.exe, msconfig.exe, ntstat.exe, msblast.exe, zonealarm.exe, wincfg.exe, taskmon.exe, sysinfo.exe, pandaavengine.exe, sinsys.exe, entre outros (SCHILLER, 2007).

Apesar da tentativa constante de deixar os bots mais imperceptíveis possíveis, os sistemas operacionais acabam apresentando sintomas que a máquina está contaminada, apresentando alguns erros como prova de que o sistema está comprometido.

Assim, um sistema operacional contaminado com um backdoor:Win32/Rbot ou qualquer de suas variantes, pode exibir caixas de diálogos informando do desligamento, reinicialização do sistema ou um erro LSA Shell, conforme figuras abaixo:

Figura 3.1 – Desligamento. Fonte: MICROSOFT, 2012d

Figura 3.2 – Reinicialiação. Fonte: MICROSOFT, 2012d

Figura 3.3 – Erro LSA Shell. Fonte: MICROSOFT, 2012d

3.4 Funcionalidades

Após o comprometimento, o nó passará a receber os comandos através do IRC, e dentre as diversas ordens, a Microsoft (2012d) enumera as possíveis ações:

- Escanear a rede a procura de computadores sem *patchs*;
- Escanear portas abertas na rede;
- Realizar downloads e executar arquivos remotos;
- Monitorar o tráfego da rede;
- Iniciar serviços de HTTP, socks4 e FTP
- Ativar ou desativar o protocolo DCOM, que é um protocolo responsável pela comunicação entre objetos em sistemas distribuídos da Microsoft;

- Recuperar informações de algumas configurações do computador, como informações de *logon*, contas de usuários, compartilhamentos abertos, sistema de arquivos e conexões de redes;
- Gravar e recuperar os logs das teclas digitadas;
- Obter chaves/licenças de programas ou jogos;
- Redirecionar tráfego TCP;
- Enviar e-mails;
- Manipular processos e serviços do sistema operacional.

Além destes, Muzzi (2010) complementa tal pensamento trazendo à luz a ameaça à privacidade das pessoas, na medida em que informações como senhas ou quaisquer outros dados confidenciais ou íntimos são capturados. E acrescenta:

- Possível comprometimento da produtividade, dificultando realização de tarefas na estação;
- O computador transforma-se em uma plataforma para atividades maliciosas;
- Comprometimento também da produtividade da rede local em que se encontra o nó infectado, visto que aumentará as atividades de tráfego na rede e, consequentemente, a degradação na largura de banda e;
- Reduz o nível de segurança do computador.

3.5 Análise do código da RxBot

O código fonte da Rxbot para realização do presente trabalho foi obtido através de download, em arquivo.zip, em fórum *underground* da internet.

Apresenta-se bem modular, assim, cada característica individual foi escrita em um arquivo separado. Isso permitirá tanto um controle fácil por parte do botherder, como também uma fácil inclusão de módulos ou alteração de serviços indesejados.

O principal arquivo de configuração é o “configs.h”, mostrado na figura 3.4, onde é possível configurar diversos parâmetros para que o Rxbot comece a operar. A variável “botid” é o nome que identificará o bot. “Version” a sua versão, e “passwd” é a senha que o botherder usará para se conectar a cada bot.

O campo “server” conterá o IP ou hostname do servidor de IRC que agirá como comando e controle. O “serverpass” atribui senha ao servidor de controle. O “channel” representa o canal para comunicação no servidor IRC, que é passado para o *bot* através do comando “join”, e, caso necessário, configurado com senha no campo “chanpass”.

Detalhando o código, pode-se verificar que ele também conta com parâmetros secundários, cujas informações de backup são fornecidas em caso de falhas nos serviços principais, tais como IP do novo servidor, porta, canais e suas senhas.

Outros parâmetros de destaque são: o local onde ficará armazenado o log de gravação do teclado (“keylogfile”), o nome do arquivo executável instalado na máquina na vítima (“filename”), a string que fará parte do apelido da vítima (“nickconst”) e os canais de resposta dos keylogger, exploits e sniffers.

```
char botid[] = "dang";              // bot id
char version[] = "[\x03\x34\2piabot\2\x03 reloaded 0.6] \2((\2by koncool\2))\2";              //
Bots !version reply
char password[] = "changeme";          // bot password
char server[] = "127.0.0.1";       // server
char serverpass[] = "";            // server password
char channel[] = "#a";             // channel that the bot should join
char chanpass[] = "b!";  // channel password
char server2[] = "127.0.0.1";      // backup server (optional)
char channel2[] = "#a";            // backup channel (optional)
char chanpass2[] = "b";            // backup channel password (optional)
char filename[] = "winlolx.exe";// destination file name
char keylogfile[] = "winabc.sys";                  // keylog filename
char valuename[] = "Windows LoL Layer";        // value name for autostart
char nickconst[] = "BoT|";                 // uses rndnick
char szLocalPayloadFile[]="winsys.dat";// Payload filename
char modeonconn[] = "-x";                  // Can be more than one mode and contain both + and -
char exploitchan[] = "#a";                 // Channel where exploit messages get redirected
char keylogchan[] = "#a";                  // Channel where keylog messages get redirected
char psniffchan[] = "#a";                  // Channel where psniff messages get redirected
```

Figura 3.4 – Parte do código fonte do módulo configs.h

Patil (2009) faz uma breve análise sobre as funções dos principais módulos da Rxbot:

- **Rbot.ccp**: módulo do código fonte mais importe, onde reside o núcleo de toda atividade maliciosa;
- **Autostart.cpp**: garante a funcionalidade de início automático, fazendo as devidas alterações no registro do sistema operacional. Para tanto, basta alterar a variável “AutoStart”, do módulo configs.h, para o valor “true”;
- **Capture.cpp**: programação responsável pela captura de dados, telas e imagens da webcam;
- **Cdkeys.cpp**: módulo que busca em várias entradas de registro, capturando informações sobre softwares licenciados;

- **Findpass.cpp**: este módulo busca senhas armazenadas no sistema operacional, inclusive na memória, armazenando e enviado para o botherder;

- **Processes.cpp**: parte fundamental que fica responsável por matar os processos indesejados para a atividade maliciosa. Além dos processos pré configurados, o botherder pode adicionar novos processos de acordo com os programas antivírus, firewall ou processos de segurança que rodam na máquina da vítima.

3.6 Técnicas de ofuscação

Como já visto, é prática comum proteger os arquivos maliciosos executáveis visando a não identificação pelos softwares antivírus. E isso é conseguido através de técnica de ofuscação, como empacotamento e criptografia de arquivos.

Empacotar o arquivo comprime o tamanho do cliente *bot* e, assim, facilita seu deslocamento e propagação pela internet, além de dificultar o escaneamento da assinatura dos arquivos.

Já a utilização de criptografia nos arquivos contaminados permite a não detecção pelos programas de antivírus, visto que este não consegue acesso completo ao arquivo infectado.

Patil (2009) demonstrou ambas as técnicas para proteger o arquivo e burlar a segurança do computador, servindo para demonstrar que mesmo o usuário possuindo um sistema operacional, programas antivírus e de segurança, é passível de sofrer um ataque.

Para conseguir mascarar o arquivo malicioso existem diversos softwares gratuitos e, com apenas alguns cliques, é possível transformar e ocultar a sua real finalidade, como o “pepack”, “Poisen Ivy Crypter”, “themida”[10], entre outros.

[10] Os programas pepack, poison ivy cryptr estão disponíveis no mundo underground para download, não sendo encontrado em sites oficiais deste. O themida pode ser encontrado no http://www.oreans.com/.

4. SIMULAÇÃO E TESTES

4.1 Escolha da Máquina Virtual

A fim de verificar as ferramentas existentes no mercado para a simulação da plataforma, foi realizada uma breve abordagem com os principais aspectos referente ao tema. Assim, foram explorados além dos conceitos iniciais, as vantagens na utilização e os principais fabricantes de VMs.

4.1.1 Conceitos de Máquinas Virtuais

Primeiramente, conforme a definição de Anjos e Nicolao (2007), uma máquina virtual – VM – é uma cópia com seu sistema devidamente isolada de uma máquina real ou uma duplicata eficiente e isolada de uma máquina real (LAUREANO,2006).

A virtualização de sistemas apareceu na década de 60, através dos *mainframes* IBM que simulavam máquinas reais, disponibilizando recursos computacionais aparentemente dedicados aos usuários. Logicamente, devido ao grande custo, essa implementação ficou restrita aos sistemas com grande capacidade de computação e alto desempenho (ANJOS;NICOLAO, 2007).

Ainda sobre o surgimento da virtualização de sistemas, Carissimi (2008) complementa que apesar da programação Java ter trazido nos últimos anos a noção de máquina virtual, os antigos *mainframes*, mesmo de um único fabricante, possuíam seu próprio sistema operacional, tornando-se a principal razão para o surgimento das máquinas virtuais e permitindo que softwares legados fossem executados nos *mainframes*.

Foi possível, então, executar ou migrar uma aplicação de uma plataforma para outra, onde na plataforma alvo haveria a versão da máquina virtual, que ofereceria uma camada de software de uma ambiente completo muito próximo de uma máquina física (CARISSIMI, 2008).

A essência da virtualização consiste em estender ou substituir um recurso ou interface, de modo a imitar um comportamento (LAUREANO, 2006), assim, cada máquina virtual terá seu próprio sistema operacional, aplicativos, serviços de rede, inclusive interconexão entre elas através de switches, roteadores e firewalls virtuais (CARISSIMI, 2008). A figura 4.1 ilustra genericamente a tecnologia de máquinas virtuais.

Conforme explica Laureano (2006), a diversidade de projetos de hardware e software, bem como de empresa, ocasionou, ao longo do tempo, uma variedade grande de plataformas operacionais, e, consequentemente, uma incompatibilidade entre si, ou seja, aplicações para uma plataforma operacional não funciona em outro tipo de plataforma.

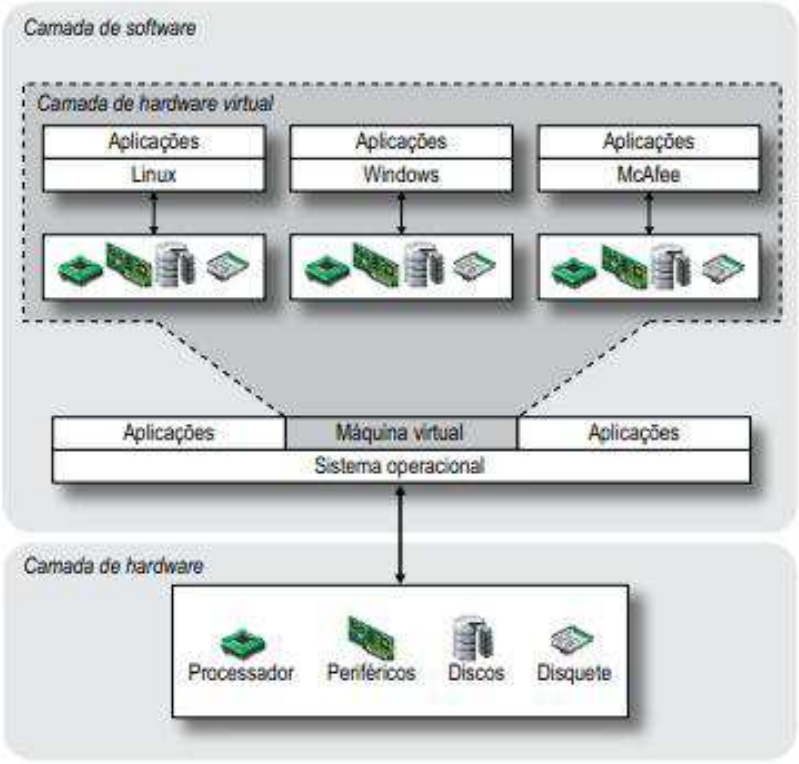

Figura 4.1 – Diagrama básico de uma máquina virtual . Fonte: ANJOS;NICOLAO, 2007

Continua o autor que com o surgimento das máquinas virtuais foi possível contornar o problema, pois com a virtualização, uma camada é criada para compatibilizar diferentes plataformas.

Atualmente, com o desenvolvimento de processadores, vários fabricantes desenvolvem soluções com baixo custo para consolidação de servidores em empresas (ANJOS;NICOLAO, 2007), oferecendo confiabilidade, isolamento e escalabilidade (MATTOS, 2008).

Outros pontos que merecem destaque na contextualização de máquinas virtuais são o *Virtual Machine Monitor* (VMM), ou Monitor de Máquina Virtual ou ainda *Hypervisor* e os modos de usuário e de supervisor.

VMM é um componente de software que hospeda as máquinas virtuais, sendo o responsável direto pela virtualização, controle dos recursos compartilhados, como processadores, memória, armazenamento e dispositivos de entrada e saída e pelo escalonamento de qual máquina terá preferência de execução (MATTOS, 2008).

Já os modos são as formas de como o computador pode operar. No modo de usuário – ou espaço de aplicação – é o modo no qual geralmente as aplicações comuns são executadas, não sendo possível executar instruções com maiores privilégios. Ao contrário, tem-se o modo de supervisor que por sua vez tem controle total sobre a CPU, executando instruções não-privilegiadas ou privilegiadas (MATTOS, 2008).

Um sistema operacional, por exemplo, trabalha no modo de supervisor, porém ao passar o controle da CPU para a aplicação de um usuário, o bit de controle de modo tem que ser setado para o modo de usuário.

Assim, de modo análogo aos sistemas operacionais, Mattos (2008) abaliza que o Monitor de Máquina Virtual é executado no modo de supervisão, enquanto cada máquina virtual é executada em modo de usuário, e caso tente executar uma instrução privilegiada, um pedido de interrupção é gerado e o controle passa para o VMM.

4.1.2 Vantagens na utilização

A utilização de VM oferece diversas vantagens, como:

- **Isolamento do sistema visitante**[11]: apesar de as máquinas virtuais compartilharem os recursos físicos de um único computador, elas permanecem totalmente isoladas uma das outras, como se realmente fossem estações físicas separadas. O isolamento ocasiona uma melhor disponibilidade e segurança de aplicativos em execução em um ambiente virtual que os executados em um ambiente tradicional (VMWARE, 2012);

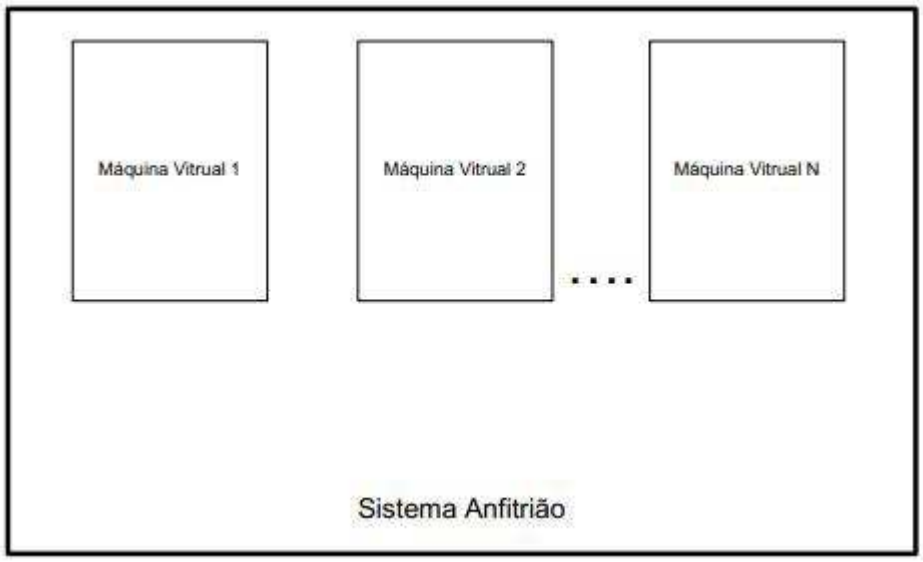

Figura 4.2 – Isolamento de máquinas virtuais. Fonte: ANJOS; NICOLAO, 2007

- **Flexibilidade de alocação de recursos**: uma única máquina física pode comportar mais de dois tipos diferentes de sistemas operacionais, o que possibilita a alocação de recursos de hardware sob demanda (ANJOS;NICOLAO,2007);
- **Custos**: de acordo com Mattos (2008), a redução de custos pode variar de 29% a 64%, através da consolidação de pequenos servidores dentro de outros mais poderosos;

[11] Dentro do ambiente virtualizado tem-se o sistema operacional hospedeiro – que é o sistema nativo e é executado diretamente sobre o hardware físico – e o sistema operacional visitante, convidado ou também conhecido por G*uest Operation System* – que é executado em cima do hardware virtualizado

- **Segurança**: através das VMs, pode se definir qual o melhor ambiente para executar cada serviço, com diferentes requerimentos de segurança, ferramentas diversas e o sistema operacional mais adequado. E, usando uma máquina virtual para cada serviço, uma possível vulnerabilidade de um serviço não prejudica as demais (MATTOS, 2008).

Ademais, Laureano (2006) aponta outros benefícios da utilização de máquinas virtuais em sistemas computacionais, tanto de mainframes como para computadores pessoais:

- Auxílio no ensino prático de sistemas operacionais e programação, pois possibilita a execução de vários sistemas para comparação no mesmo equipamento;
- Poder ter várias máquinas virtuais, cada uma com um sistema operacional diferente sem a necessidade de particionar o disco rígido;
- Simulação de alteração e falhas de hardware;
- Portabilidade de aplicações;
- Facilidade de gerenciamento, migração e replicação de computadores, aplicações ou o próprio sistema operacional;
- Facilidade de treinamento e demonstração de produtos.

4.1.3 Propriedades das Máquinas Virtuais

Algumas características são desejáveis para o funcionamento ideal das VMs, conforme trabalho de Laureano (2006):

- **Isolamento**: a vantagem do isolamento deve ser observada à medida que um processo de uma VM não deve interferir no funcionamento de outro processo, nem comprometer o desempenho de outras máquinas virtuais (MUZZI, 2010);

- **Inspeção**: o *hypervisor* tem a capacidade de acesso e controle sobre todas as informações dos processos das VMs, como estado da CPU, memória, eventos e outros;
- **Interposição**: o *hypervisor* também pode intercalar ou acrescentar instruções em algumas operações, como execução de instruções com privilégio. Assim, instruções do tipo “espera” podem ser inseridas na máquina virtual enquanto o monitor processa instruções privilegiadas (MUZZI, 2010);
- **Eficiência**: por não interferir em outras máquinas virtuais ou aplicações, instruções consideradas inofensivas podem ser executadas diretamente no nardware;
- **Gerenciabilidade**: como são instâncias distintas, a administração de uma VM independe da outra VM, deixando o controle simples e centralizado;
- **Compatibilidade de software**: devido à abstração fornecida pela VM, todo software escrito para uma determinada plataforma, será também executado em uma VM que virtualize aquela mesma plataforma;
- **Encapsulamento**: nas VMs, o monitor pode controlar totalmente os processos, gerando inclusive *checkpoints* do sistema para possíveis recuperações após uma falha (MUZZI, 2010);
- **Desempenho**: mesmo que uma camada de software seja acrescida a um sistema e afete o desempenho deste, o benefício proporcionado ainda será válido.

4.1.4 Principais fabricantes

Atualmente, têm-se diversas soluções disponíveis para virtualização, tanto para uso comercial, gratuito, baseada em software livre ou já integrada ao sistema operacional. Os principais fabricantes são:

4.1.4.1 VMware

O VMware tornou-se uma das mais populares soluções para virtualização, oferecendo infra-estrutura completa tanto para desktops como *data centers*. As soluções são divididas em três categorias básicas: gerenciamento e automação, infra-estrutura virtual e plataformas para virtualização (MATTOS, 2008 e VMWARE, 2012).

Na virtualização de plataformas, assunto relacionado ao presente trabalho, suas aplicações são voltadas tanto para servidor como desktops. Entre os produtos fornecidos pela VMware, pode-se encontrar o VMware Workstation, Fusion, Player, voltados para usuários domésticos, e VMware ESX Server, Server, vSphere Hypervisor (ESXi) para maiores corporações (VMWARE, 2012).

O VMware é executado da mesma forma que um programa comum, em um sistema operacional hospedeiro, ficando responsável pela abstração de processador, memória, armazenamento e recursos de rede (COELHO;CALZAVARA;LUCIA, 2010).

Muzzi (2012) explica que a arquitetura do VMware trabalha com virtualização no nível de processador,em que as instruções privilegiadas a serem executadas são capturadas e

virtualizadas pelo VMM, enquanto as outras instruções são executadas no próprio processador hospedeiro.

Outro ponto interessante ilustrado pelo autor é que os recursos de hardware também são virtualizados. Assim, o suporte para a utilização para os diversos dispositivos é fornecido pelo próprio sistema operacional hospedeiro. Para a máquina ter acesso ao dispositivo real, é instalado um *driver* de dispositivo, o *VMDriver*. Na rede, por exemplo, a placa será colocada em modo promíscuo, recebendo todos os quadros ethernet, e criando uma ponte (*bridge*), que será responsável por encaminhar tais quadros para o sistema hóspede.

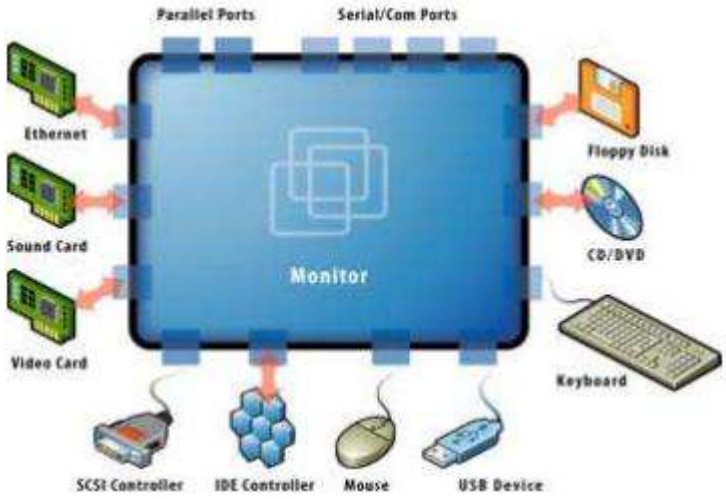

Figura 4.3 – Virtualização de dispositivos no VMware. Fonte: MATTOS, 2008

A figura 4.3 representa uma máquina virtual usando arquitetura VMware, com os dispositivos hardware virtualizados, como placa de rede, som, vídeo, USB e demais hardwares.

4.1.4.2 VirtualBox

VirtualBox é um programa virtualizador da Oracle, de propósito geral, voltado para hardware x86, para uso em servidores ou desktops. Está disponível gratuitamente como software de código aberto sob os termos da GNU – General Public License –, suporta como anfitriões os sistemas operacionais Windows, Linux, Macintosh e Solaris, e como convidado, a família Windows (NT, 2000, XP, Server 2003, Vista e 7), DOS/Windows3.x, Linux, Solaris e Open Solaris, OS/2 e OpenBSD (VIRTUALBOX, 2012).

O programa apresenta arquitetura modular e como característica principal o gerenciamento através de uma interface bem definida, facilitando a administração de várias máquinas virtuais ao mesmo tempo, conforme figura 4.4.

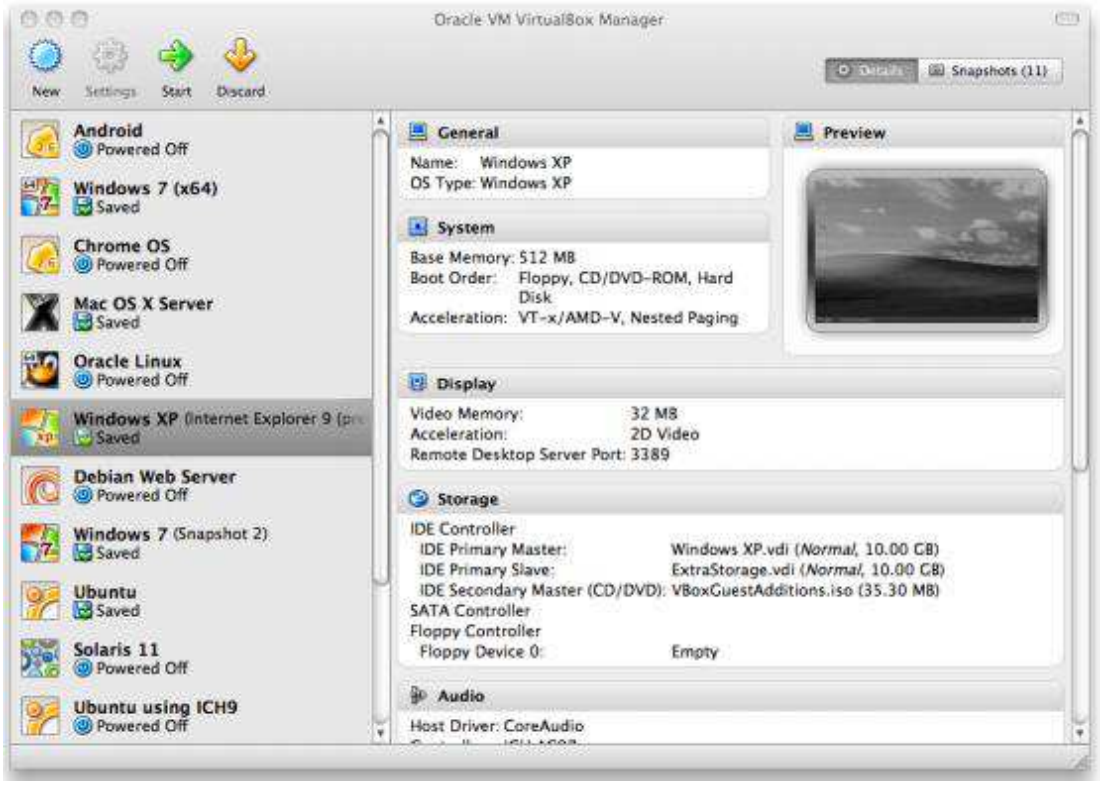

Figura 4.4 – Interface de gerenciamento do VirtualBox. Fonte: VIRTUALBOX, 2012

Muzzi (2010) destacou que as ferramentas do VirtualBox também são de fácil compreensão, além de serem constantemente modificadas e testadas e as principais são:

- **VirtualBox GUI**: interface gráfica administrativa, já visto na figura acima;
- **VBoxManage**: interface de linha de comando para controle completo a partir do sistema hospedeiro. Ele além de suportar todos os recursos da interface gráfica, possui recursos extras não disponíveis no modo gráfico (VIRTUALBOX, 2012);
- **VBoxSDL**: alternativa que apresenta uma interface gráfica simplificada através de recursos intencionalmente limitados, destinados a mostrar somente o essencial para execução da máquina virtual;
- **VBoxHeadless**: sua função principal é oferecer acesso remoto, não havendo a necessidade de apresentar uma interface gráfica visível na estação hospedeira.

4.1.4.3 Xen

Antes de apresentar este fabricante, é necessário saber que na virtualização propriamente dita, toda a infra-estrutura completa de hardware subjacente é virtualizada, não sendo necessário modificar o sistema operacional convidado para que este consiga ser executado sobre o VMM. Já a paravirtualização é uma técnica na qual uma arquitetura similar, mas não idêntica à arquitetura real, é apresentada ao sistema operacional emulado (ANDRADE, 2006), conforme ilustração.

Nesse caso, conforme Laureano (2006), o sistema virtualizado (convidado) sofre modificações para que as interações com o monitor de máquinas virtuais sejam mais eficientes, aumentando a performance das máquinas virtuais.

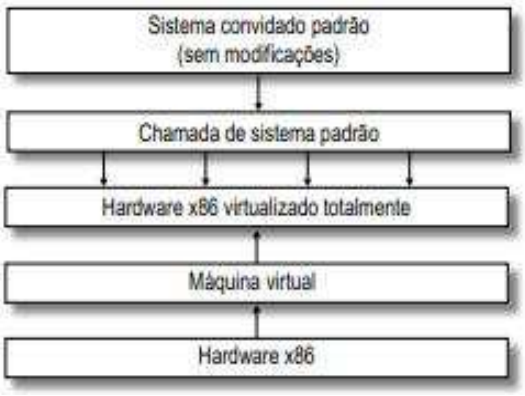

Figura 4.5 – Virtualização total. Fonte: LAUREANO, 2006

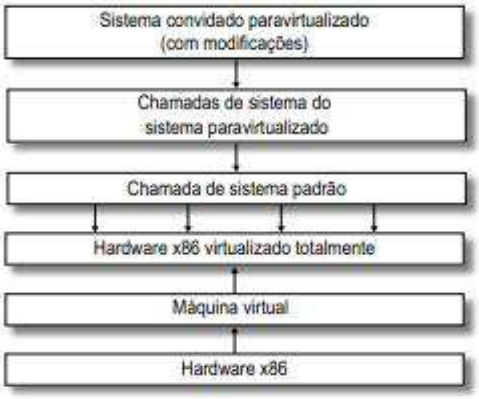

Figura 4.6 – Paravirtualização. Fonte: LAUREANO, 2006

Deste modo, Muzzi (2010) e Mattos (2008) citam que o Xen é um dos mais populares exemplos de paravirtualização, onde o sistema operacional visitante não tenta executar tarefas protegidas diretamente na CPU, mas sim entregá-las ao monitor de máquina virtual.

Xen é uma camada de software desenvolvida pela comunidade Xen.org, padrão *open source*, que roda diretamente no hardware do computador, permitindo que nele sejam executados vários sistemas operacionais convidados (XEN, 2012).

Muzzi (2010) informa que o Xen tem uma arquitetura mais específica que os demais fabricantes, onde utilizam os conceitos de *domínio* e *hypervisor*. O domínio é a forma como são batizadas as máquinas virtuais no Xen, e podem ser domínio 0 ou privilegiada ou domínio U ou não privilegiada. Já o hypervisor controla os recursos de comunicação, memória e processamento.

Carissimi (2008) esclarece que o hypervisor Xen não possui *drivers* de dispositivos, não sendo capaz de suportar nenhuma interação com os sistemas hóspedes, e assim temos que:

> É necessário que exista um sistema inicial para ser invocado pelo *hypervisor*. Esse sistema inicial é o domínio 0. As outras máquinas só podem ser executadas depois que ele for iniciado. As máquinas de domínio U são criadas, iniciadas e terminadas através do domínio 0. O domínio 0 é uma máquina virtual única que executa um núcleo Linux modificado e que possui privilégios especiais para acessar recursos físicos de entrada e saída, interagindo com as demais máquinas virtuais (domínio U) (CARISSIMI, 2008).

Mattos (2008) afirma que, de início, a escolha por paravirtualização era justificada pelo acréscimo em desempenho, sendo bem melhor que a virtualização total. Porém, com o desenvolvimento de tecnologias como AMD-V e Intel VT, a virtualização padrão passou a obter resultados tão bons ou até melhores que a paravirtualização.

4.1.4.4 Microsoft Virtual PC

A Microsoft também possui uma grande variedade de produtos voltados para virtualização, inclusive soluções para *datacenters*, mas o que se destaca para o presente trabalho é o software Virtual PC.

O Virtual PC é a máquina virtual da família Windows, podendo ser configurada para executar outros sistemas operacionais, e, segundo a Microsoft, citada por Carissimi (2008), sua principal função é o desenvolvimento e testes de software para múltiplas plataformas.

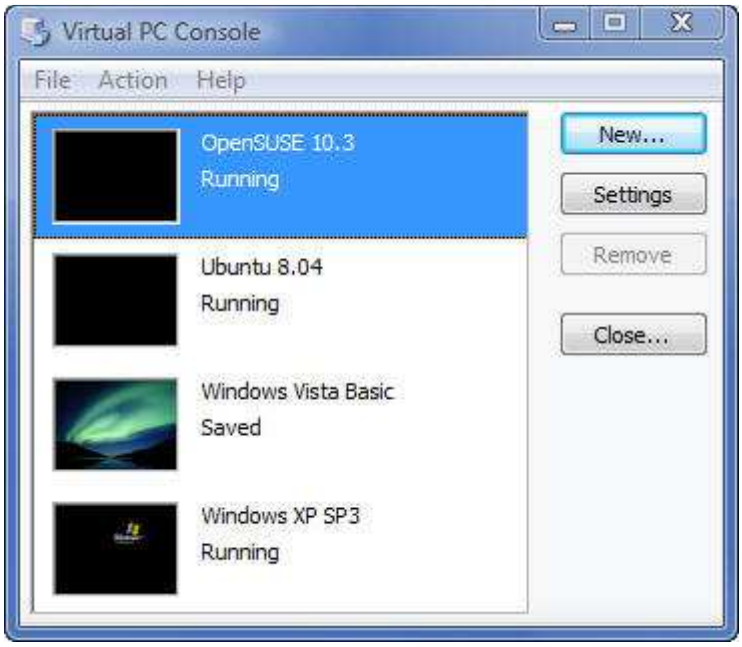

Figura 4.7 – Tela de gerenciamento do Virtual PC da Microsoft. Fonte: CARISSIMI, 2008

Como as outras máquinas virtuais, o Virtual PC permite interconectar logicamente diferentes máquinas, pois possui seu próprio endereço MAC e IP, tem também servidor NAT e *switches* virtuais.

Muzzi (2010) afirma que as máquinas virtuais estão sendo cada vez mais utilizadas, provendo ambientes mais otimizados, um menor número de máquinas reais, uma maior organização do parque tecnológico e um menor consumo de energia. E, como método de simulação de *botnets*, é o ambiente quase perfeito, pois fornece o confinamento necessário e vital para a realização de testes com segurança.

4.2 Criação da plataforma virtual

Assim, diante das características apresentadas por cada fabricante no tópico anterior, como facilidade de gerenciamento, robustez, exportação e importação de appliances, descrição em XML, segurança, portabilidade, consumo de memória, entre outros, decidiu-se por realizar os testes através do VirtualBox.

Para o presente trabalho, a simulação foi realizada utilizando-se o sistema de virtualização VirtualBox. Um servidor IRC foi instalado em uma máquina virtual baseada em Linux, e, por apresentar simplicidade no manuseio, o Ubuntu 12.04 foi escolhido, já as estações foram simuladas com Windows XP, no esquema da figura 4.8.

Figura 4.8 – Esquema de rede da plataforma virtual

Para deixar a plataforma de simulação completamente isolada e portável, inclusive para esta apresentação, decidiu-se pela não utilização de servidores e canais de chat já existentes na internet. Para tanto, foi instalado e testado em uma máquina virtual o IRCD-Hybrid (IRCD-HYBRID, 2012), provendo um serviço somente local.

Ainda sobre a compartimentação, a rede montada deve permite acesso somente entre as estações virtuais. No VirtualBox, cada máquina criada teve sua configuração de rede alterada para “rede interna”, mantendo-se o nome padrão da rede, conforme figura abaixo:

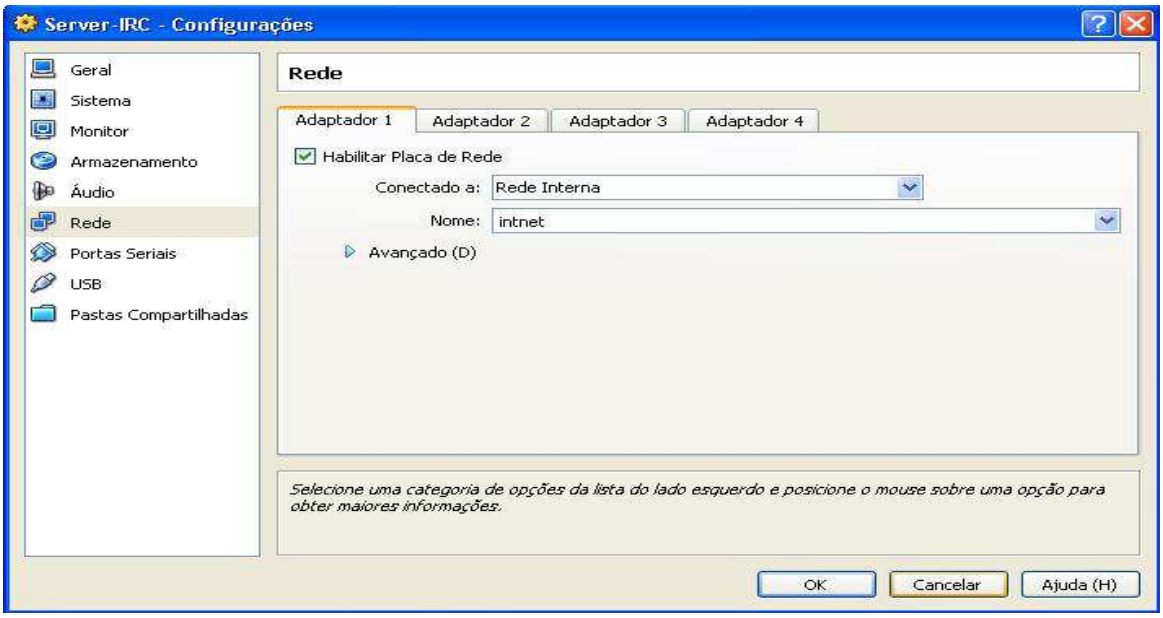

Figura 4.9 – Configuração de rede local no VirtualBox. Fonte: VIRTUALBOX, 2012

4.3 Instalação e configuração do serviço de IRC

Existem diversos programas para prover serviço de IRC, inclusive para plataformas Windows, e com uma infinidade de recursos. Devido à simplicidade e robustez, o IRCD-Hybrid foi configurado para realização dos testes.

Esse aplicativo, disponível apenas para plataformas Linux, versão estável 7.3.1 ou versão beta 8.0.0 (IRCD-HYBRID, 2012), foi instalado a partir do comando:

- sudo apt-get install ircd-hybrid

Como a simulação roda em rede local, o arquivo de configuração do hybrid (/etc/ircd-hybrid/ircd.conf) foi alterado conforme figura abaixo, e reiniciado com <sudo invoke-rc.d ircd-hybrid restart>:

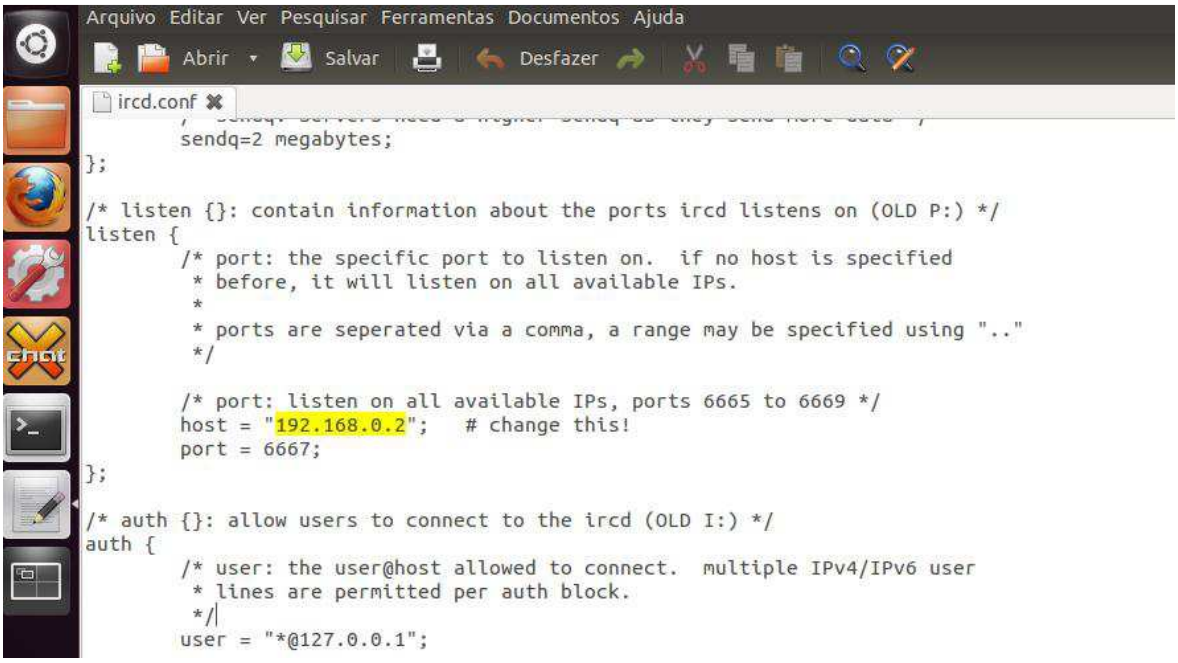

Figura 4.10 – Arquivo de configuração do IRCD-Hybrid. Fonte: IRCD-HYBRID, 2012

4.4 Compilação do bot

Conforme já explicado, o código dá Rxbot foi escrito em linguagem C++, então para adequá-lo ao presente estudo foi necessário alterar algumas linhas do arquivo configs.h através do Visual C++. As principais alterações dizem respeito ao servidor local, porta e canal para cada *bot* se conectar.

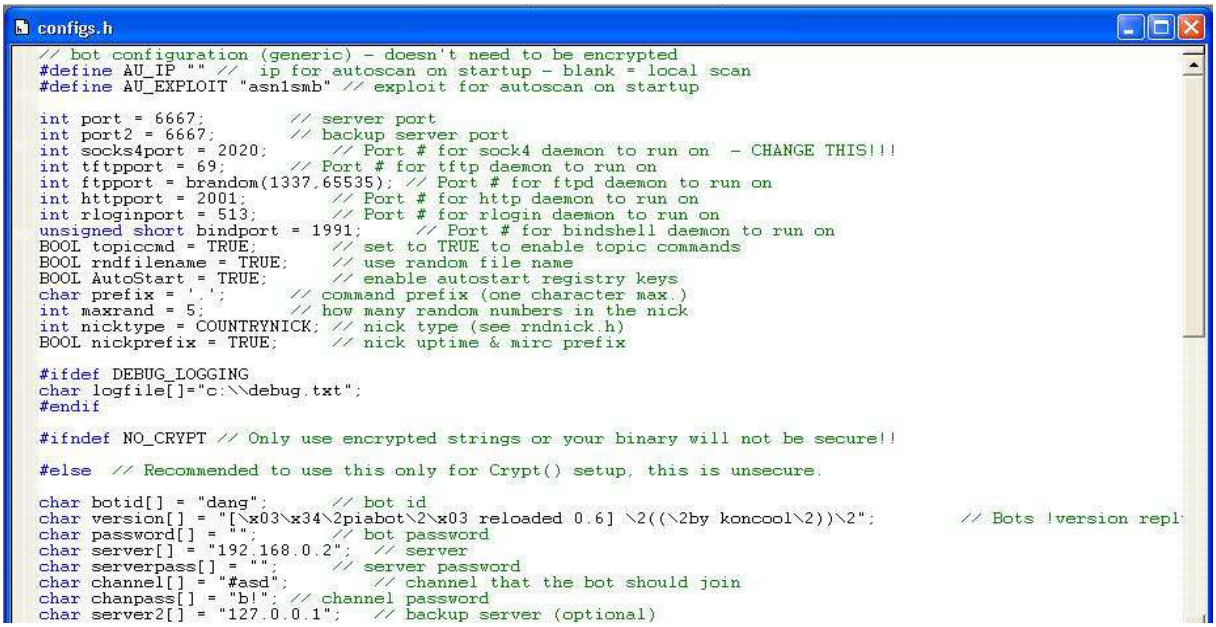

Figura 4.11 – Arquivo de configuração da RxBot

Para a integração e compilação do novo código é necessária a instalação da Plataforma SDK da Microsoft, além da inclusão das seguintes linhas no diretório do Visual C++:

- C:\Program Files\Microsoft Platform SDK\;
- C:\Program Files\Microsoft Platform SDK\Bin\;
- C:\Program Files\Microsoft Platform SDK\Include\;
- C:\Program Files\Microsoft Platform SDK\Lib\

Após a compilação, o arquivo executável virótico estará disponível na pasta "debug" do diretório que contém os códigos maliciosos, com o nome de "rbot.exe".

4.5 Análise do malware

Para se ter uma ideia do grau de risco e contaminação, o arquivo criado foi submetido ao antivírus online Jotti (2012), ferramenta que permite verificar arquivos suspeitos diretamente em aproximadamente 20 fabricantes de antivírus.

Como resultado, foi verificado que 18 dos programas antivírus identificaram o arquivo como malicioso, com diversos nomes de variantes, conforme figura:

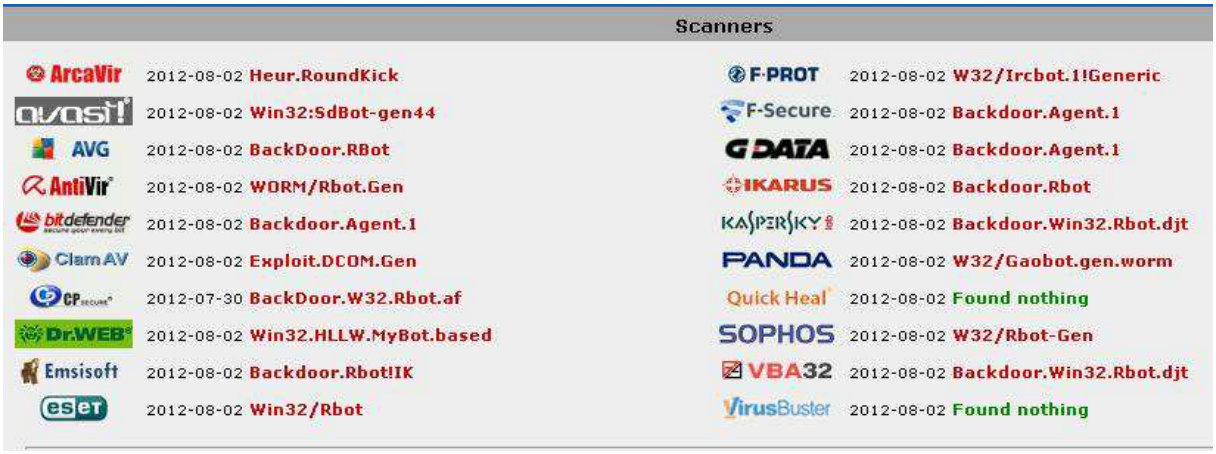

Figura 4.12 – Verificação de vírus no arquivo "rbot.exe"

Já com um identificador de arquivos, o CFF Explorer, foi possível identificá-lo realmente como um arquivo executável e escrito em linguagem C++, conforme figura abaixo.

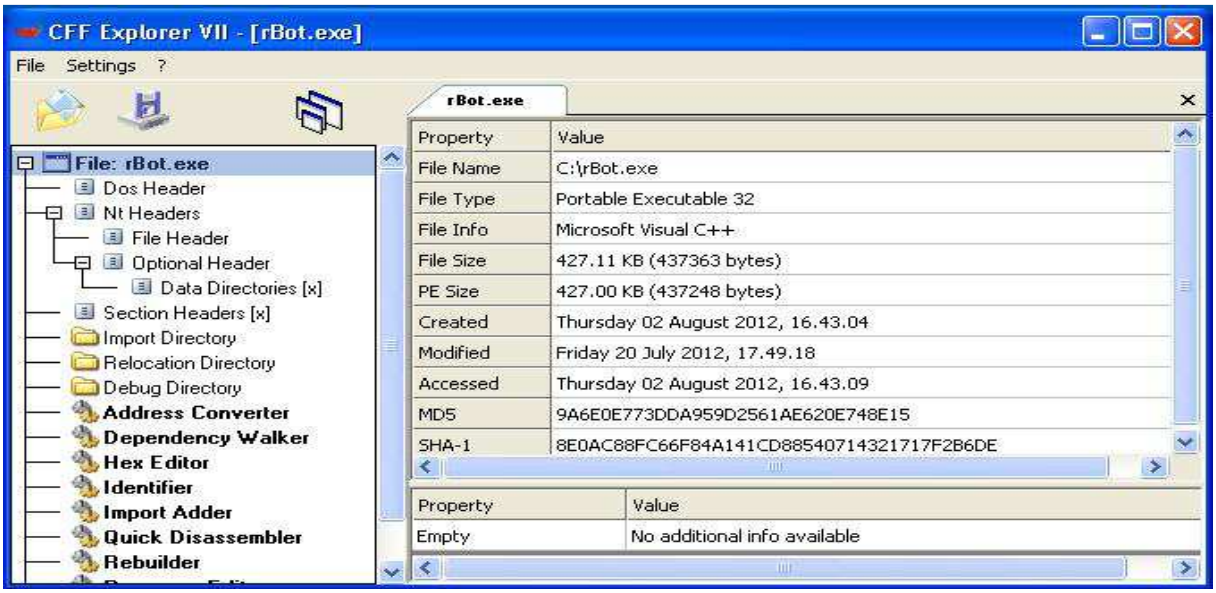

Figura 4.13 – Identificação do arquivo malicioso rbot.exe

Por fim, uma breve análise com um programa disassembler, IDA, foi possível obter alguns rastros que seriam deixados pelo criador do código malicioso, como por exemplo, os IPs ou canais configurados pelo *botherder*, conforme visto na figura 4.13.

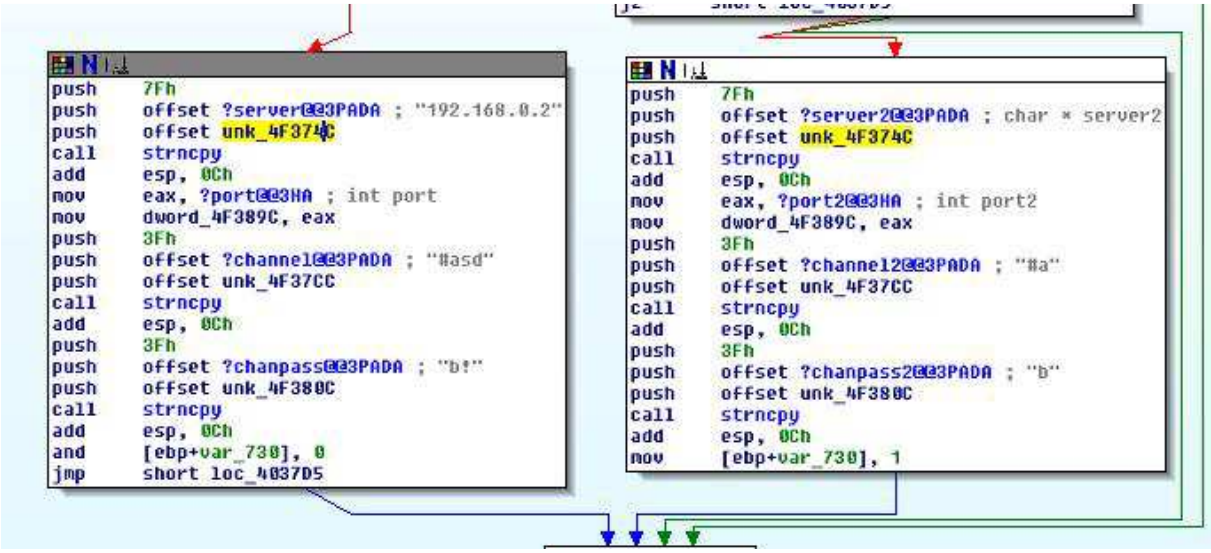

Figura 4.14 – Engenharia reversa no arquivo executável

4.6 Testes

A fim de verificar e demonstrar algumas funcionalidades da botnet, procedeu-se com alguns testes que iniciaram com a contaminação de um nó, reconhecimento e execução de comandos e algumas modalidades de ataque, conforme se segue.

4.6.1 Contaminação

A contaminação inicial ocorreu com um duplo clique no arquivo executável “rxbot.exe”. Ressalta-se que na prática a contaminação, em geral, é através do recebimento de arquivos, de emails, cavalos de Tróia, entre outros, e, mesmo assim, os arquivos são disfarçados com técnicas de ofuscação, conforme visto em tópico acima.

Após a contaminação, o *botherder* é logo avisado no canal de chat pré-estabelecido. Também, depois de contaminado, a cada reinicialização da máquina do usuário, o *botherder* recebe a mensagem que o nó contaminado está online. Na figura 4.15 é possível observar o zumbi conectado ao C&C, que está na estação Linux.

Figura 4.15 – Aviso ao botherder que usuário está online

Já figura 4.16, podemos verificar as conexões que foram estabelecidas entre o C&C e os *bots*, através do comando netstat –a ou netstat –na | grep 667.

4.6.2 Envio e execução de comandos

A RxBot permite o envio de alguns comandos, tanto para obtenção de informações, como para execução de atividades maliciosas:

Comando	Sintaxe	Descriçao
Execute	.execute <visibilidade> <arquivo>	Executar um programa na vítima, 0, invisível e 1 visível
Capture	.capture screen <arquivo>	Capturar a tela da vítima
Get	.get <arquivo>	Enviar um arquivo via DCC
Delete	.delete <arquivo>	Deletar um arquivo
Keylog	.keylog on|off	Ativar ou desativar o monitoramento de teclas
Driveinfo	.driveinfo	Informações sobre espaço total, livre e usado do disco
Findfile	.findfile <curinga> <diretório>	Pesquisar arquivos

Who	.who	Informação de quem está logado
Netinfo	.netinfo	Retornar informação de IP e rede
Open	.open <arquivo>	Semelhante ao “execute”, podendo abrir web browser
Sysinfo	.sysinfo	Retorna informações sobre o sistema
Findpass	.findpass	Decodificar e exibir as credenciais do administrador Windows
Getcdkeys	.getcdkeys	Busca e retorna chaves de programas instalados
Killproc	.killproc <nome do processo>	Killar um processo da lista
Procs	.procs	Listar os processos em memória
Readfile	.readfile <arquivo>	Ler o conteúdo de um arquivo
Download	.download <url> <destino><ação>	Realiza download, grava em um diretório, podendo executá-lo

Tabela 4.1 – Comandos da RxBot

4.6.3 Ataques

A RxBot possui suporte para alguns tipos de ataques, como *icmpflood*, *pingflood*, *synflood*, *tcpflood* e *udpflood*. Atualmente, com a expansão da largura de banda e os constantes *patches*, tem se tornado mais difícil realizar um ataque dessa modalidade.

4.6.3.1 PingFlood

Pingflood, conforme já visto, é um ataque de negação de serviço que sobrecarrega o sistema da vítima através do envio insistente de pacote ICMP *Echo Request*, deixando o alvo impossibilitando de responder todas as requisições.

No teste em questão, com o envio de 1000 pacotes, e apenas duas máquinas contaminadas e bombardeando o alvo, IP 192.168.0.5, nada de anormal foi percebido a não ser um fluxo constante de pacotes no analisador Wireshark.

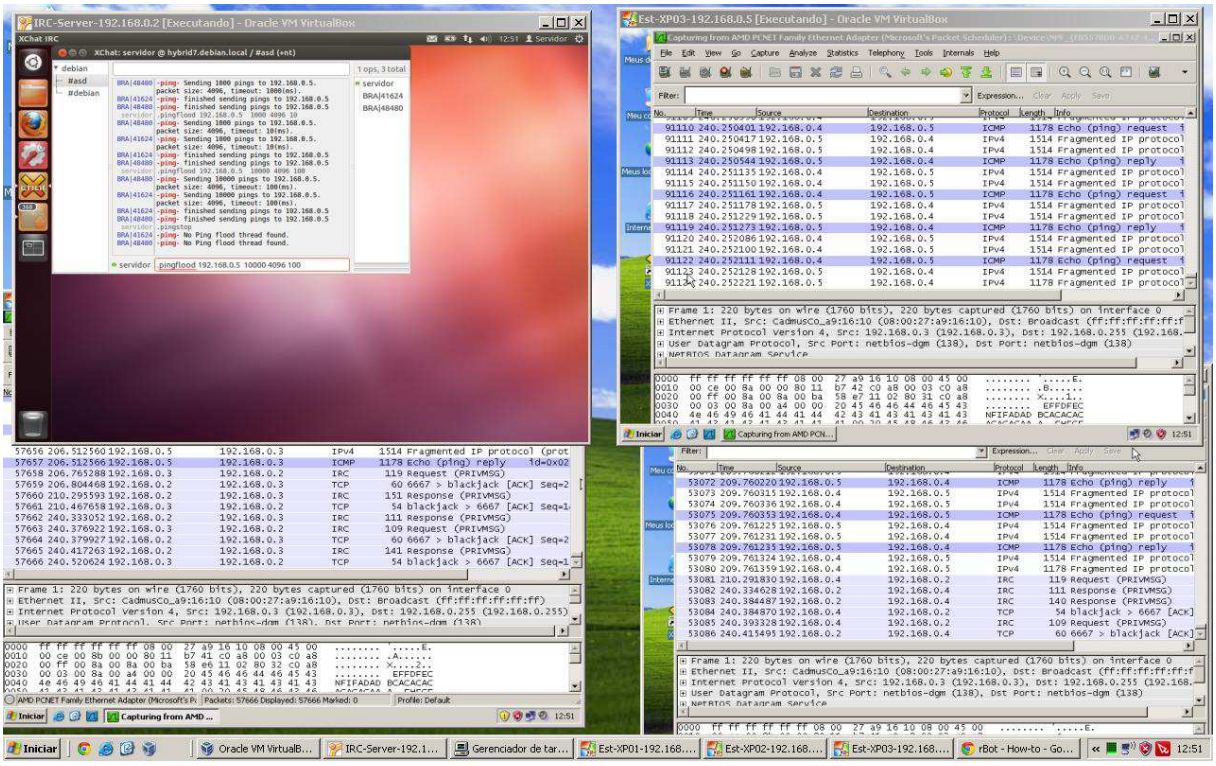

Figura 4.16 – Demonstração do PingFlood no ambiente de simulação

Aumentando o número de pacotes para 10.000, a estação vítima mostrou-se sobrecarregada, apresentando lentidão e não conseguido processar o recebimento dos pacotes a tempo.

4.6.3.2 UDPFlood

Da mesma forma que o ataque anterior, o UDPFlood é capaz de deixar indisponíveis os serviços de uma rede. Um ataque dessa modalidade inicia-se com o envio de um grande número de pacotes UDP.

O protocolo UDP não é orientado a conexão, não fornecendo garantia de entrega do pacote, assim, o ataque consiste simplesmente em enviar pacotes randômicos para várias portas do serviço da vítima.

Ao receber os pacotes, a vítima, com o IP 192.168.0.5, tentou verificar qual aplicação está aguardando por aquele pacote na porta associada. Porém, como não existe, de fato, nenhuma aplicação aguardando, a vítima emite um pacote de resposta ICMP para a coleção de bots, no caso os IPs 192.168.0.3 e 192.168.0.4. O comportamento foi semelhante ao ataque ping.

5. CONCLUSÃO

Nos últimos anos, temos observado o desenvolvimento progressivo e a proliferação assustadora das *botnets* ou redes zumbis. Redes de computadores vêm sendo alvo constante de ataques de hackers ou grupo de hackers, com a finalidade, quase sempre, de cometer atividades maliciosas.

As estações vítimas são infectadas por um código malicioso, deixando-as agir automaticamente e inconscientemente sobre o controle de seu dono, o *botherder*.

Esse trabalho apresentou uma simulação de *botnets* através do uso de máquinas virtuais. Foi possível observar, além das noções introdutórias sobre o tema, os atores envolvidos, a dinâmica de criação, com suas formas de contaminação e propagação, as maneiras possíveis de controlar os *bots*, as finalidades da utilização das *botnets* no mundo, bem como o caso particular da RxBot.

A plataforma de simulação permitiu primeiramente realizar todo o trabalho com somente uma máquina física, evitando maiores custos ou necessidade de aquisição de computadores e equipamentos de redes. Também se conseguiu um perfeito isolamento, onde cada VM funcionou como se realmente fosse uma estação física. Ofereceu ainda disponibilidade, segurança nos procedimentos (visto que foi manipulada certa quantidade de vírus), flexibilidade com uma rápida recuperação após falhas, portabilidade e um eficiente gerenciamento das estações envolvidas, o servidor e os nós vítimas.

Em seguida, com a plataforma, pode-se verificar o funcionamento, na prática, do ciclo de vida de uma *botnet*, desde a manipulação e compilação do código malicioso que

originou o vírus, a exploração das vítimas, o estágio da contaminação e sua propagação e a maneira de controlar as diversas máquinas contaminadas.

Nos testes aplicados, foi visto como os recém nós infectados se comunicavam com o *botherder* ou como estes são avisados após um nó já contaminado ficar online na coleção de *bots*.

Além disso, uma série de comandos foi enviada pelo C&C, inclusive não somente da estação servidora, mas também de qualquer estação que ele venha se conectar. Dentre os principais comandos, foram testados: execução de arquivos, captura e recebimento de telas, ativação de *keylogger* e recebimento das teclas digitadas, envio e deleção de arquivos, informações sobre estações logadas e seu sistema operacional, informações sobre a rede, busca de arquivos, escâner de rede encerramento de processos.

Os testes prosseguiram com alguns ataques do tipo DDOS, buscando, na medida do possível, um alinhamento da plataforma à realidade, tendo o tráfego entre nós e controlador observado num analisador de rede.

Diante do exposto, com o mundo em freqüente transformação, avanço constante da tecnologia da informação e uso de computadores para os mais diversos fins, vemos a importância do presente estudo, viabilizando meios para compreensão crítica de um grande problema que afeta milhares de redes e usuários de computadores.

REFERÊNCIAS BIBLIOGRÁFICAS

AMOROSO, Edward G. Cyber Attacks - Protecting National Infrastructure. United States of America: Elsevier, 2011.

ANDRADE, M. T. Um estudo comparativo sobre as principais ferramentas de virtualização. Trabalho de conclusão de curso – Centro de Informática, Universidade Federal de Pernambuco, 2006.

ANDRESS, Jason; WINTERFELD, Steve. Cyber Warfare – Techiniques, Tatics and Tools for Security Practitioners. Massachusetts: Elsevier, 2011.

ANJOS, J.C.S, NICOLAO M. Sistemas virtualizados, laboratórios de inteligência artifical e computação em grid, legere sistemas dinâmicos pesquisa e desenvolvimento. Departamento de Sistemas de informação – Ulbra/Guaíba, 2007.

ANTISPAM.BR. Comitê Gestor da Internet no Brasil – CGI.br, 2011. Disponível em: < http://www.antispam.br >. Acesso em: 09 jan. 2012.

BAMBENEK, J e KLUS, A. Botnets and Proactive System Defense. Em: W. Lee e C. Wang (eds.), *Botnet Detection – Coutering the Largest Security Threat*, Spring. New York, 2010.

CARISSIMI, A. Virtualização: da teoria à soluções. Instituto de Informática – UFRGS. 26º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. Porto alegre, 2008.

CHARALABIDIS, A.The book of IRC – The ultimate guide to Internet Relay Chat. United States of America: No Starch Press, 2000.

CISCO. Botnets; The New Threat Landscape. Disponível em <.http://www.cisco.com/en/US/solutions/collateral/ns340/ns394/ns171/ns441/networking_solutions_whitepaper0900aecd8072a537.pdf>. Acesso em: 11 dez. 2011.

COELHO, F.A, CALZAVARA, G.S., LUCIA, R . Virtualização: VMware e Xen. Rio de Janeiro: GTA/POLI/UFRJ, 2010. Disponível em: < http://www.gta.ufrj.br/grad/09_1/versao-final/virtualizacao/index.html>. Acesso em: 03 jul. 2012.

DENNING, Dorothy E. Cyberterrorism – Testimony before the Special Oversight Panel on Terrorism - Commitee on Armed Services. Georgetown University. Maio, 2000. Disponível em: <http://www.cs.georgetown.edu/~denning/infosec/cyberterror.html>. Acesso em: 21 set. 2011

DUNHAM, K e MELNICK, J. Malicious Bots – Na Inseide look into the Cyber Criminal Underground of the Internet. United States of America: CRC Press, 2009.

FINANCIAL SERVICES TECHNOLOGY CONSORTIUM - FSTC - The Technology Policy Division of the Financial Services Roundtable. Understanding and Countering the Phishing Threat. Disponível em: < www.bits.org/publications/members/fraud/escams1205.pdf >. Acesso em: 25 mar. 2012.

HOWARD, R. et al. Cyber Fraud – Tactis, Techniques and Procedures. United States of America: CRC Press, 2009.

IRC.ORG. Internet Relay Chat – Technical Documents. Disponível em <http://www.irc.org/techie.html>. Acesso em: 07 jun. 2012.

IRCD-Hybrid. High Performance Internet Relay Chat. Disponível em: <http://www.ircd-hybrid.org/downloads.html>. Acesso em: 11 jul. 2012.

Jotti – Verificador de Malwares. Disponível em <.http://virusscan.jotti.org/>. Acesso em: 02 ago. 2012.

LAUREANO, M. Máquinas virtuais e emuladoras. São Paulo: Novatec editora, 2006.

MATTOS, D. M. F. Virtualização: VMware e Xen. Rio de Janeiro: GTA/POLI/UFRJ, 2008. Disponível em: < http://www.gta.ufrj.br/grad/08_1/virtual/artigo.pdf>. Acesso em: 27 jun. 2012.

MAYNOR, D. et al. Emerging Threat Analysis from mischief to Malicious. Canadá: Syngress, 2006.

MCAFEE. McAfee Threats Report: First Quarter 2011. Disponível em: < http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q1-2011.pdf>. Acesso em: 02 jan. 2012.

MCAFEE. McAfee Threats Report: Third Quarter 2011. Disponível em: < http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q3-2011.pdf>. Acesso em: 02 jan. 2012.

MICROSOFT – Malware Protection Center Threat Research and Response. Disponível em: < http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?Name=_Backdoor%3aWin32%2fRbott>. Acesso em: 18 jun. 2012.

MICROSOFT. Microsoft Security Intelligence Report. Volume 11. Disponível em: <http://download.microsoft.com/download/0/3/3/0331766E-3FC4-44E5-B1CA-2BDEB58211B8 /Microsoft_Security_Intelligence_Report_volume_11_English.pdf>. Acesso em: 03 jan. 2012.

MICROSOFT. MSRT – Windows Malicious Software removal Tool – Progress Made and Tren Observerd, a paper from the Microsoft Antimalware Team, 2006. Disponível em: <http://download.microsoft.com/download/5/6/d/56d20350-afc8-4051-a0df-677b28298912/msrt%20-%20progress%20made%20lessons%20learned.pdf>. Acesso em: 16 jun. 2012.

MICROSOFT. Relatório de Inteligência de Segurança Microsoft. Volume 10. Disponível em: < http://www.microsoft.com/downloads/details.aspx?FamilyID=821e0433-5b9d-422d-8d78-ab641ee6e132&displayLang=pt-br>. Acesso em: 01 out. 2011.

MICROSOFT. Security Intelligence Report – Featured Articles, 2011. Disponível em: <http://www.microsoft.com/security/sir/story/default.aspx#!botnetsection>. Acesso em: 01 out. 2011.

MICROSOFT. The Oficial Microsoft Blog - Cracking Down on Botnets. Fev, 2010. Disponível em: <http://blogs.technet.com/b/microsoft_blog/archive/2010/02/25/cracking-down-on-botnets.aspx>. Acesso em: 29 set. 2011.

MUTTON, P. IRC Hacks – 100 Industrial-Strenght Tips & Toos. United States of America: O´Reilly, 2004.

MUZZI, Fernando Augusto Garcia. *Análise de botnet utilizando plataforma de simulação com máquinas virtuais visando detecção e contenção.*2010. 144 f. Tese (Doutorado) – Escola Politécnica da Universidade de São Paulo.

NAKAMURA, E e GEUS, P. Segurança de redes em ambientes Cooperativos. Brasil: Novatec Editora, 2007.

NATIONAL VULNERABILITY DATABASE. Search CVE and CCE Vulnerability Database. Disponível em: < http://web.nvd.nist.gov/view/vuln/search>. Acesso em: 18 dez. 2011.

PATIL, Esha. Analysis of RxBot.2009. Tese (Mestrado) – Faculty of the Departamento of Computer Science San José State University.

POSLUNS, J e SJOUWERMAN, S. Inside the Spam Cartel: Trade Secretes from the Dark Side. United States of America: Syngress, 2004.

RFC – Request for Comments 1459 – Internet Relay Chat Protocol. Disponível em <.http://tools.ietf.org/html/rfc1459 >. Acesso em: 05 jun. 2012.

RFC – Request for Comments 2813 – Internet Relay Chat Protocol. Disponível em <.http://tools.ietf.org/html/rfc2813 >. Acesso em: 05 jun. 2012.

RFC – Request for Comments 2810 – Internet Relay Chat Protocol. Disponível em <.http://tools.ietf.org/html/rfc2810>. Acesso em: 05 jun. 2012.

RFC – Request for Comments 2811 – Internet Relay Chat Protocol. Disponível em <.http://tools.ietf.org/html/rfc28101>. Acesso em: 05 jun. 2012.

SAMPAIO, Fernando G. Ciberguerra - Guerra Eletrônica e Informacional - Um novo Desafio Estratégico. Escola Superior de Geopolítica e Estratégia. Abril, 2001. Disponível em: <http://www.defesanet.com.br/esge/ciberguerra.pdf>. Acesso em: 22 set. 2011.

SCHILLER, C.A. et al. Botnet – The killer web app. Syngress, 2007.

STINSON, E e MITCHELL, J. C. Characterizing Bots, Remote Cntrol Behavior. Em: W. Lee e C. Wang (eds.), *Botnet Detection – Coutering the Largest Security Threat*, Spring. New York.

STRAYER, W. T. et al. Botnet Detection Based on Network Beavior. Em: W. Lee e C. Wang (eds.), *Botnet Detection – Coutering the Largest Security Threat*, Spring. New York, 2010.

SYMANTEC. Secure Response - Internet Security Threat Report - Latin America. Disponível em: < http://www.symantec.com/business/threatreport/topic.jsp?id=lam&aid=lam_countries_of_botnet_spam_origin>. Acesso em: 10 set. 2011.

SYMANTEC. Secure Response - Internet Security Threat Report 2010 in Review. Disponível em: <http://www.symantec.com/business/threatreport/topic.jsp?id=threatreport&aid=executive_summary>. Acesso em: 10 set. 2011.

TANENBAUM, Andrew S. Redes de Computadores – Tradução Vandenberg D.de Sousa. Rio de Janeiro: Editora Campus, 2003.

THE HONEYNET PROJET. Know your Enemy: Phishing. Agosto, 2008. Disponível em: <http://www.honeynet.org/papers/phishing/>. Acesso em: 26 mar. 2012.

THE HONEYNET PROJET. Know your Enemy: Tracking Botnets. Agosto, 2008. Disponível em: <http://www.honeynet.org/papers/bots/>. Acesso em: 12 jun. 2012.

TZU, Sun. A Arte da Guerra. São Paulo: Saraiva, 1998.

Unidade de Repressão a Crimes Cibernéticos, Departamento de Polícia Federal. Manual de investigação de crimes cibernéticos. Brasília: ANP, 2011.

VirtualBox. Oracle VM VirtualBox. Disponível em: <http://www.virtualbox.org>. Acesso em: 03 jul. 2012.

VMware. Soluções de virtualização e infra-estrutura em nuvem. Disponível em: <http://www.VMware.com/br/solutions/>. Acesso em: 03 jul. 2012.

VMWARE. Virtualization Basics – Transform your Business with virtualization. Disponível em: < http://www.VMware.com/virtualization/virtual-machine.html>. Acesso em: 27 jun. 2012.

WANG, P. et al. Peer-to-peer Botnets: The Next Generetion of Botnet Attacks. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.153.6675&rep=rep1&type=pdf >. Acesso em: 28 mai.2012.

Xen. What is a Xen Hypervisor. Disponível em: <http://xen.org/files/Marketing/WhatisXen.pdf>. Acesso em: 05 jul. 2012.