MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
SEÇÃO DE ENGENHARIA DE COMPUTAÇÃO

VINICIUS RAMOS EDUARDO

RENATO DIAS COSTA

MODELAGEM E SIMULAÇÃO DO TRÁFEGO *BOT*

Rio de Janeiro
2013

INSTITUTO MILITAR DE ENGENHARIA

VINICIUS RAMOS EDUARDO
RENATO DIAS COSTA

MODELAGEM E SIMULAÇÃO DO TRÁFEGO *BOT*

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

Orientador: Sérgio dos Santos Cardoso Silva, M.Sc.

Rio de Janeiro
2013

INSTITUTO MILITAR DE ENGENHARIA

VINICIUS RAMOS EDUARDO
RENATO DIAS COSTA

MODELAGEM E SIMULAÇÃO DO TRÁFEGO *BOT*

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

Orientador: Sérgio dos Santos Cardoso Silva, M.Sc. Aprovada em 27 de Maio de 2013 pela seguinte Banca Examinadora:

Sérgio dos Santos Cardoso Silva, M.Sc. do IME - Presidente

Anderson Fernandes Pereira dos Santos, D.Sc, do IME

Julio Cesar Duarte, D.C, do IME

Rio de Janeiro
2013

RESUMO

As *botnets* são redes formadas por um conjunto de agentes de *software* maliciosos (*bots*) com a capacidade de se infiltrar em um ou mais computadores. Essa invasão permite que *malwares* sejam distribuídos, informações sigilosas sejam acessadas e graves ataques sejam provocados como, por exemplo, o de *DDoS* (distribuido de negação de serviço). A motivação do estudo se deve ao rápido avanço das *botnets* bem como ao fato de que nenhum sistema operacional ou rede de computador comercializado hoje apresenta qualquer mecanismo eficiente de defesa contra essa ameaça.

Neste trabalho serão apresentados os conceitos básicos de classificação e funcionamento de uma *botnet*, assim como algumas das modelagens mais comuns utilizadas para a simulação de ambientes de ataque e defesa associados a essa ameaça. Serão discutidos alguns dos principais desafios conceituais associados ao entendimento global da tecnologia dos exércitos *bot* sem, por outro lado, detalhar os aspectos técnicos intrínsecos à construção efetiva de um ambiente de simulação. Além disso, será estudada uma ferramenta que foi desenvolvida para simular tais ambientes, o *framework Rubot*, associado ao estudo mais aprofundado de simulação para que se possa analizar os dados obtidos pelo *framework*. Para fins de simulação, usou-se a *botnet Nugache* no *Rubot* para coletar dados e tirar conclusões a cerca da eficácia do *framework*.

ABSTRACT

Botnets are networks formed by a set of malicious software agents (bots) with the ability to infiltrate in one or more computers. This invasion allows malware to be distributed; qualified information to be accessed and very severe attacks to be taken place such as a DDoS (Distributed Denial of Service) attack. The motivation of the study is due to the rapid development of botnets as well as to the fact that none of the operating systems or networks commercialized today has any effective defense mechanism against this threat.

This paper will present the basic concepts of classification and operation of a botnet, and some of the mathematical models commonly used for the simulation of attack and defense environments associated with this network. We will discuss some of the key challenges associated with a general, conceptual understanding of bot army technology without, however, detailing technical aspects intrinsic to the actual construction of such simulation environments. Besides this, to emulate those enviroments a framework called Rubot will be studied. To help proccess the data obtained from the framework, a further study on simulation will be made either. For the simulation, the botnet nugache was used with the Rubot to genarate data to make conclusions about the efficience of the framework.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

1 INTRODUÇÃO

Com o aumento do número de computadores conectados à internet no mundo e a crescente diversificação e personalização das atividades nela realizadas, tornou-se cada vez mais frequente a prática de atividades criminosas aproveitando-se desse contexto. Um dos mais recentes e sofisticados meios para a prática desse tipo de atividade é a *botnet*: rede formada por máquinas infectadas por softwares maliciosos, com a particularidade de ser comandada por uma máquina central.

1.1 MOTIVAÇÃO

A importância do estudo das *botnets* se deve, primeiramente, ao fato desse tipo de tecnologia maliciosa ter se desenvolvido de maneira extremamente rápida e já ter se mostrado capaz de provocar ataques de grande porte, atingindo instituições financeiras e públicas, em dimensões nacionais e internacionais. Em conjunto a isso, não há, atualmente, sistemas operacionais ou redes que apresentem qualquer mecanismo de defesa eficaz contra esse tipo de ameaça. O que aumenta bastante a gravidade da ameaça.

Dado o seu rápido desenvolvimento em diversas regiões no mundo, juntamente com a inexistência de sistemas de proteção adequados a seu combate, pode-se concluir que toda a sociedade usuária da Internet é, nos dias atuais, uma potencial vítima de ataques e danos provocados por esse tipo de rede. A ameaça das *botnets* é um problema de escala internacional e, por essa razão, o entendimento de sua dinâmica de funcionamento é de interesse imediato não só a acadêmicos ou profissionais da indústria da computação, mas de toda a população mundial conectada à Internet.

1.2 OBJETIVOS

Esta pesquisa tem, como seu objetivo geral, a realização de um estudo global sobre *botnets* e, em seguida, um estudo sobre simulação de *botnet*. Com o uso do *framework Rubot* e a análise específica da *botnet Nugache*, obter-se-á as informações necessárias para modelar a fase de conexão do ciclo de vida de um *bot*, que é uma das fases mais vuneráveis. Neste etapa, em especial, serão cumpridas etapas do objetivo geral, tendo como base os seguintes objetivos específicos:

- Estudo expecífico da *botnet Nugache*;
- Utilizar o *Rubot* para simular uma Nugache e verificar sua eficácia como simulador.

1.3 JUSTIFICATIVA

O estudo de simuladores de *botnets* ainda é, no presente momento, uma frente de atuação relativamente pouco desenvolvida no meio acadêmico nacional e internacional. Este trabalho busca contribuir para o conhecimento de peculiaridades desse tipo de rede através dessa diretriz de pesquisa e poderá, ainda, ser estendido para um futuro trabalho, visando maior aprofundamento do tema.

1.4 METODOLOGIA

Durante a primeira documentação da pesquisa, adotou-se como foco a exploração dos fundamentos teóricos associados às *botnets*, definindo conceitos basicos de classificação e funcionamento. Posteriormente, na segunda parte da documentação, introduziu-se o modelo de análise estatística para a validação de uma simulação e deu-se início ao estudo específico do funcionamento do *software Rubots*. Para finalizar a documentação, focou-se no estudo da *botnet Nugache* e sua simulação no *framework Rubot*.

Nesse intuito, será discutido primeiramente as definições associadas às *botnets*, criando uma base necessária para o desenvolvimento do trabalho. Em seguida, serão contemplados os desafios associados e os aspectos importantes a serem considerados dentro de uma simulação de *botnet*. Após isso, será introduzido o *framework Rubot* de simulação e análise de *botnets*, que será utilizado como gerador de dados para que possa ser feita a verificação de um principal aspecto: a eficiência do *framework* como simulador de uma *botnet* específica. Tal *botnet* será a *Nugache*, por isso se faz necessário seu estudo mais aprofundado, para que tenha-se uma base teórica quando a simulação for feita através do *Rubot* e sua eficiência, como simulador.

1.5 ORGANIZAÇÃO

Este trabalho está organizado em capítulos.

No segundo capítulo, será feito um estudo conceitual sobre *botnets*. Nele, serão desenvolvidos os seguintes tópicos: definição, identificação dos componentes, características mais comuns, objetivos, ciclo de vida e arquiteturas de uma *botnet*.

No terceiro capítulo, será feito um estudo dos princípios de simulação e modelagem de uma *botnet*. Nele, serão desenvolvidos os seguintes tópicos: análise dos principais desafios de sua simulação, paradigmas fundamentais de simulação de rede e introdução ao modelo estatístico de simulação.

No quarto capítulo, será introduzido uma ferramenta de análise e simulação de *Botnets*, o *Rubot*. Nele, serão desenvolvidos os seguintes tópicos: Apresentação, motivação do uso, arquitetura, funcionalidades e uso futuro.

No quinto capítulo, será introduzido conceitos sobra a *botnet Nugache* e a simulação dela no *framework Rubot*. Nele, serão desenvolvidos os sequintes tópicos: *Nugache*, Simulação da *Nugache* no *Rubot*.

No sexto capítulo, será feita a conclusão da pesquisa e sugestões de trabalhos futuros para a continuação da mesma.

2 *BOTNET*

2.1 DEFINIÇÃO

Botnets são redes formadas por computadores "escravos" que foram infectados, chamados de *bots* (derivada da palavra inglesa *robot*), que são controladas por um ou mais atacantes chamados de *botmasters* (SILVA, 2012). Em outras palavras, *bots* são *malwares*[1] rodando em computadores escravos executando atividades maliciosas pela rede, controlados por um *botmaster* (CHOI, 2009). Normalmente, uma *botnet* tem fins criminosos, como roubar senhas de cartão, disparar *e-mails* de *spam* pela Internet ou realizar ataques *DDoS* (do inglês, Distribuido de Negação de Serviço).

O número de *bots* existentes é muito difícil de ser calculado, pois muitas máquinas passam imperceptíveis aos próprios usuários, porém estima-se que existem *botnets* que controlam em torno de 1 milhão de *bots*, como é o caso da Rustock (FOSSI, 2011). Tal rede, com tamanha dimensão, consegue efetuar ataques quase que indefensáveis, e de grande dimensão (FOSSI, 2011).

O número de estudos destinados à detecção de *botnets* e sua paralização estão aumentando bastante utimamente (FEILY, 2009). Somente agora que pesquisadores, autoridades, usuários e empresas estão dando a devida importância ao assunto.

2.2 COMPONENTES DE UMA *BOTNET*

Para o melhor entendimento do funcionamento de uma *botnet*, se faz necessário a apresentação de seus componentes, como visto na figura 2.1. Vale ressaltar que o que faz o estudo de *botnet* tão amplo e complexo são suas particularidades, porém o que será discutido a seguir é similar em todas a *botnets*. Tais componentes são:

- *Bot*: É o *malware* instalado em uma máquina vunerável, que é controlada remotamente, para realizar atividades maliciosas. Tal malware pode infectar a vítima de inúmeras formas, como acesso a sites infectados ou aberturas de *e-mails* spam.

[1] **Malware**:do inglês,*malicious software*; software que destina-se a se infiltrar em um computador alheio com o intuito de causar algum dano ou roubo de informações por meio de um *software*.

- Máquina Vítima: Máquina dos usuários que poder ser infectadas pelos *bots*.
- *Botmaster*: Um ou mais usuários que controlam a *botnet* através de um centro de comando e controle, com fim geralmente malicioso. O *botmaster* também pode obter lucro alugando parte da sua infraestrutura para outros usuários enviarem spams, por exemplo (COOKE, 2005; FOSSI, 2011).
- Centro de Comando e Controle (*C&C*): Infraestrutura que serve para comandar os *bots*, requerindo uma conexão entre o *bot* e o centro. É comandado pelo *botmaster* e permite executar todas as ações que ele desejar.

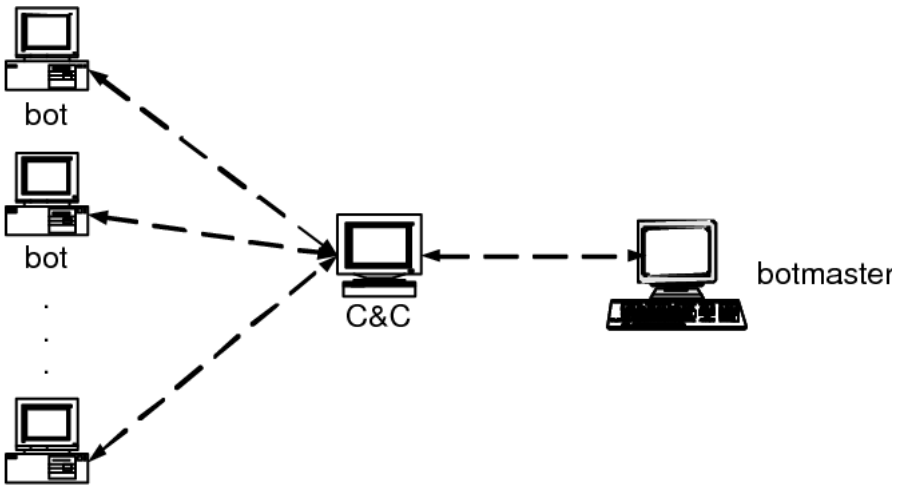

FIG. 2.1: Estrutura básica

2.3 CARACTERISTICAS DESEJÁVEIS EM UMA VÍTIMA

Uma máquina pode ser infectada de inúmeras maneiras, desde emails falsos até sites maliciosos, mas existem critérios que podem ser empregados na procura de máquinas ideais para compor uma *botnet*. Costuma-se procurar aquelas com vulnerabilidade, geograficamente espaçadas e com conexão de alta velocidade à Internet.

Utilizando conexão rápida, consegue-se fazer ataques DDoS mais eficientes (COOKE, 2005), outro motivo seria que com o uso da Internet de alta velocidade, o usuário da máquina escrava pode não perceber que parte de sua banda está sendo utilizada para uma atividade maliciosa (SILVA, 2012).

Máquinas que estão geograficamente distantes ajudam no fato de ser mais difícil de serem localizados e desconectadas pelos orgãos competentes, tendo em vista que

sua jurisdição se restringe a seu país e não existe uma agência internacional que faz a fiscalização.(FEILY, 2009) diz que *botnets* podem abranger vários países dificultando tanto sua detecção quanto sua paralisação.

Outra característica procurada são máquinas com uso do sistema operaciona MS Windows, já que a maioria dos *malwares* rodam somente nesse sistema operacional (SILVA, 2012), devido principalmente às suas vulnerabilidades e ao número maciço de usuários.

2.4 OBJETIVOS DE UMA *BOTNET*

As *botnets* possuem certos objetivos, algumas são usadas para mais de um, outras são especificas. Os principais e mais usuais são::

- Coleta de informações: Existem *bots* especializados em capturar informações que o usuário digitou, recebeu ou passou, como teclas pressionadas, arquivos, senhas e outros. Essas *botnets* podem ser usadas para roubar senhas de cartão de crédito, de *e-mail* e de qualquer outra informação.

- Ataques *DDoS*: São *botnets* usadas para realizar ataques de negação de um serviço ou de uma rede.

- Repositório *Malware*: É quando alguns *bots* dentro da *botnet* são responsáveis pelo armazenamento dos binários que instalam o(s) *malware*(s) nas máquinas recém infectadas. Esses repositórios evitam conexões desnecessárias ao centro de comando e controle ou ao próprio *botmaster*. Essas conexões oferecem um grande risco para o *botmaster* pois facilita a detecção da *botnet* (SILVA, 2012).

- Conteúdo ilegal: São os *bots* designados para armazenar conteúdo ilegal, como as informações coletadas por outros *bots*. Outro uso comum é a pirataria de mídias, como videos e músicas, que podem ser armazenados nesses *bots*.

- Anonimato e impunidade: Como mencionado anteriormente, toda *botnet* tenta preservar o anonimato do seu *botmaster* e garantir que ele não seja punido por seus atos. Quando se usa uma infinidade de máquinas espalhadas pelo mundo para realizar ataques, é muito difícil rastrear a origem dos comandos, ou seja, o real invasor.

2.5 CICLO DE VIDA

O ciclo de vida de um *bot* é composto por cinco fases, que são identificadas de formas diferentes dependendo da literatura, porém, na maioria dos casos, elas são representadas como na figura 2.2:

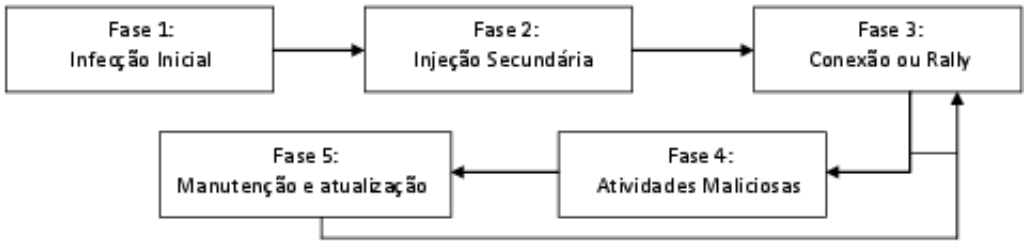

FIG. 2.2: Ciclo de Vida de uma *Botnet*

A primeira fase é a Infecção Inicial que é a primeira invasão na máquina vítima e pode ser feita de diversas maneiras, por exemplo, através de downloads de *malwares* sem solicitação, abertura de *spam*, dispositivos móveis infectados ou anexos infectados em emails (ZHU, 2008; FEILY, 2009). Essa fase é essencial e necessária, porém não garante que a máquina se torne um *bot* (SILVA, 2012). Para tal é necessário ocorrer a segunda etapa, conhecida como Injeção Secundária. Nessa segunda fase, a máquina previamente infectada procura os binários em um repositório pré-definido. Após baixa-los e executá-los ela se transforma em um *bot* ou *zombie*.

Após o *bot* ter "nascido", ele precisa, em algum momento, se conectar a um centro de Comando e Controle (*C&C*), que varia dependendo da arquitetura empregada na *botnet*. Esse ponto é a terceira etapa, conhecida como Conexão ou *Ralling* (FEILY, 2009), onde o *bot* se conecta ao *C&C* para receber instruções e atualizações do que fazer nos próximos ataques. Essa fase ocorre várias vezes, pois é nela que o *botmaster* se assegura que o *bot* está conectado a *botnet*.

A fase de Conexão é considerada a fase mais vulnerável dentro do ciclo de vida do *bot*, devido à necessidade de contato com o *C&C*. Nessa fase pode se descobrir muito sobre a *botnet*, inclusive identificar o *C&C* e maneiras de pará-lo (SILVA, 2012). Pode ocorrer da segunda e da terceira fase acontecerem simultaneamente, ou seja, os repositórios de

binários podem estar no próprio *C&C*, por isso, alguns autores juntam essas duas fases em uma só.

Nesse ponto, após feita a conexão, o *bot* está pronto para exercer sua função primária, esperar comandos para executar ativadades maliciosas (ZHU, 2008; FEILY, 2009). Essa fase se chama Ativade Maliciosa, e nela a troca de mensagens entre o *C&C* e o bot aumenta, pois, são recebidas as instruções do tipo de atividade que a máquina vai realizar como por exemplo: roubo de informações, ataque DDoS, disparo de *e-mails* de spam ou procura de potenciais novos *bots* (IANELLI, 2005; ZHU, 2008). Caso o *bot* não realize nenhum ataque, por exemplo, seja somente um repositório de binários, ele precisa continuar a se conectar com o *C&C* caso surja a necessidade da mudança de atividade. É dai que surge o *loop* visto na figura 2.2.

A útima fase do ciclo é a manutenção e atualização do *bot*, pois o *botmaster* deseja manter seu exército em pleno funcionamento (SILVA, 2012). Essas atualizações podem ter como objetivos evitar novas técnicas de identificação, novos *malwares* ou realizar a migração para outro *C&C* (ZHU, 2008; FEILY, 2009). Após a manutenção, é necessária refazer a conexão para receber novas instruções.

2.6 TIPOS DE ARQUITETURAS

O centro de comando e controle é o canal que une os *bots* de forma que virem uma só *botnet*. Existem algumas arquiteturas que permitem tal união, tais como: centralizada, descentralizada e híbrida.

2.6.1 CENTRALIZADA

A arquitetura centralizada se assemelha bastante a uma arquitetura cliente-servidor utilizada em redes de computadores (SILVA, 2012). Nesse modelo, todos os *bots* estabelecem conexão com um ou mais *C&C* que são responsáveis por enviar instruções e realizar atualizações nos *malwares*.

Essa arquitetura tem como vantagem a baixa latência, pois um bot está sempre conectado a pelo menos um *C&C*. Outra seria a facilidade com que o *botmaster* gerencia a *botnet*, pois ele tem controle dos *C&C*, logo também de todos os *bots*.

Como desvantagem, o próprio servidor *C&C* é um ponto central de falha (MICRO, 2006), ou seja, uma vez detectado, fica simples de desativar a *botnet* inteira, e uma vez achado um *C&C*, os possíveis outros *bots* podem ser identificados com maior facilidade.

Um dos protocolos mais utilizado nessa arquitetura é o *Internet Realy Chat*(IRC), que funciona, basicamente, com o *botmaster* criando vários canais IRC pelos quais os *bot* se conectam. Uma grande vantagem disso é a flexibilidade dada a *botnet*, podendo o *botmaster* destinar uma parte específica da sua rede para um ataque, ou gerenciar vários ataques simultâneos. A principal desvantagem do emprego do protocolo IRC é o fato dele ser raramente utilizado nos dias atuais, ou seja, mesmo que seja muito flexível, é de fácil detecção, pois uma vez que o usuário da máquina infectada perceber que está utilizando esse protocolo, já desconfiará que algo está errado (MICRO, 2006; WANG, 2007). Para contornar tal desvantagem, ultimamente tem sido utilizado o *HyperText Transfer Protocol* (HTTP), protocolo de aplicação amplamente utilizado e mais difícil de ser detectado.

2.6.2 DESCENTRALIZADA

Atualmente, as técnicas de detecção de *botnets* tornaram-se bastante sofisticadas, surgindo assim a necessidade do aumento da flexibilidade e robustez dessas redes. Uma saída foi descentralizar o comando da *botnet*, ou seja, nao utilizar um servidor *C&C* central. Esse tipo de arquitetura descentralizada é bastante difícil de ser descoberta (SILVA, 2012), e mesmo que seja, ainda é difícil sua total desarticulação, pois com a ausência do *C&C*, mesmo que uma boa quantidade de *bots* sejam desativados, não existe um canal central que controle todos os *bots* (SILVA, 2012), tornando mais difícil descobrir a origem do ataque, ou seja, o *botmaster*.

Essa arquitetura é comumente gerenciada através de uma variedade de protocos *Peer-To-Peer*(P2P) e funciona como uma rede de sobreposição.

2.6.3 HÍBRIDA

Arquiteturas híbridas são aquelas que utilizam elementos tanto da centralizada, quanto da descentralizada. Uma possível forma de implementar tal estrutra seria através de protocolos P2P utilizando *superpeers*, onde alguns *peers* são escolhidos temporariamente como "*C&C*". Estes nós possuem características de servidor, ficando a cargo do controlador da *botnet* selecionar quais serão os *superpeers*, que podem variar dependendo da situação.

Em (WANG, 2007) detalha-se uma *botnet* híbrida como possuindo dois tipos de *bots*, os servidores e os clientes. Nesta arquitetura, o cliente não pode receber conexões e é configurado com endereço IP dinâmico, podendo estar localizado atrás de *firewalls* que

não possuem conexão global com a Internet.

Já os servidores possuem tanto o lado cliente quanto o servidor, têm endereço IP estático e são os únicos candidatos a ter seu endereço IP na lista de *peers* para serem conectados. Esse tipo de arquitetura funciona muito bem, pois os clientes só podem se conectar com *peers* servidores, já que só eles entram na lista de *peers*. Essa conexão passa por uma encriptação, tornando-a segura, e periodicamente, os clientes se conectam a todos os servidores presentes em sua lista de *peers* para atualizações, assim como é feito na arquitetura centralizada.

3 INTRODUÇÃO À SIMULAÇÃO DE *BOTNETS*

Antes de iniciar considerações específicas sobre a simulação de uma *botnet* e, em particular para este trabalho, analisar a eficiência do *software Rubot* para uma tentativa de reproduzir o comportamento de *botnets*, é importante definir alguns dos principais fundamentos de simulação computacional para permitir que as futuras análises nesta pesquisa sejam coerentes e o mais precisas possível.

Uma simulação pode ser considerada, por definição, como uma imitação do funcionamento de um processo ou sistema no mundo real, ao longo do tempo (BANKS, 2001). A tentativa de unir um modelo bem definido – e matematicamente tratável – a uma descrição razoavelmente fiel da realidade nos leva, em geral, à adoção de um modelo simplificado desta. Com avanço da computação, porém, tem se tornado cada vez mais simples tomar um modelo tão fiel quanto possível e, então, basear-se num estudo de simulação para analisá-lo (ROSS, 2002).

3.1 TIPOS DE SIMULAÇÃO

Nesta seção, apresentaremos tipos importantes de simulação de redes em geral, analisados em publicações acadêmicas, apresentando suas características básicas e seus paradigmas. Os tipos mais importantes de simulação de redes são baseados em:

- Agente: neste, leva-se em consideração que cada elemento do sistema apresenta características individuais, que são capazes de perceber o ambiente ao seu redor, tomar decisões e de interagir com outros elementos. Um projeto bem desenvolvido em simulação baseada em agente requer aplicações de conceitos de inteligência artificial (KOTENKO, 2008);

- Componente: na simulação baseada em componentes, são utilizados diferentes softwares, os quais podem ser dispostos em hierarquia ou não, funcionando como blocos independentes para a simulação do ambiente de interação (KOTENKO, 2012);

- Eventos Discretos: nesta simulação, o estado do sistema é representado por uma sequência cronológica de eventos discretos. Considera-se, nesta simulação, que os

eventos acontecem de maneira estritamente sequencial, isto é, jamais ocorrerão dois eventos simultaneamente e que a ocorrência de um evento sempre altera o estado do sistema (KOTENKO, 2012);

- Nível de pacotes: neste tipo, considera-se que a comunicação entre os elementos do sistema é feita por meio de unidades fundamentais de informação, ou pacotes. Desta maneira, a interação entre dois elementos num intervalo de tempo pode ser resumida a um conjunto finito de pacotes contendo informações elementares (DAINOTTI, 2006).

- Nível de fluidos: nesta forma, considera-se que as informações trafegam entre os elementos do sistema não por meio de unidades individuais, mas sim de maneira fluida e contínua. Em situações em que há altas taxas de transmissão, um modelo em nível de pacotes se torna inviável, tornando mais recomendável o tratamento de dados por meio de abstração de fluidos (AGUIAR, 2008.).

É importante observar que os tipos de simulação acima não são excludentes, de modo que, a priori, é possível a criação de um modelo global que envolva todos eles. Modelos envolvendo combinações dos tipos acima já foram, com efeito, realizados em trabalhos acadêmicos, como em (KOTENKO, 2012).

3.2 ANÁLISE DE SIMULAÇÃO

Um estudo de simulação tem como objetivo avaliar a qualidade de um modelo proposto pra um fenômeno real. Numa simulação computacional, todo modelo deve ser abstraído para um conjunto finito de variáveis, o qual traduzirá o comportamento do fenômeno simulado. Para que diferentes situações possíveis sejam simuladas é necessário, portanto, que diferentes valores sejam definidos para as variáveis de simulação. Fenômenos reais como, por exemplo, infecção de máquinas por *bots*, chegada e saída de clientes numa fila de atendimento ou variação do preço de um ativo no mercado de ações frequentemente envolvem fatores imprevisíveis ao observador e, por essa razão, simulações computacionais comumente envolvem atribuição de valores aleatórios a suas variáveis de simulação.

3.2.1 VARIÁVEIS PSEUDOALEATÓRIAS

Até o presente momento, não há entre pesquisadores um conceito unânime para número aleatório, de modo que não se pode afirmar se há ou não um computador capaz

de gerar números aleatórios por definição. Porém, com o uso de técnicas computacionais, é possível gerar variáveis de maneira determinística, porém cujo caráter é aparentemente imprevisível para um observador comum. Essa forma de variável é classificada como variável pseudoaleatória.

O bloco fundamental na construção de um estudo de simulação computacional é a geração de números pseudoaleatórios uniformemente distribuídos no intervalo real entre 0 e 1 (ROSS, 2002). A partir dessa geração, pode-se utilizar métodos para obter a geração de números com distribuição qualquer, dentro de um intervalo qualquer.

Uma técnica comumente utilizada para isso é a do Gerador Congruente Linear. Esse método consiste na construção de uma sequência de naturais $\{x_0\}$ a partir de escolha de quatro números naturais convenientes a, c, m e x_0 onde x_0 é o primeiro termo e $x_0 < m$, e da seguinte equação de recorrência:

$$x_{n+1} = (ax_n + c) \bmod m$$

A partir desta sequência, define-se uma sequência de números reais $\{y_n\}$ por meio da equação $y_n = \frac{x_n}{m}$. Desse modo, temos que $\{y_n\}$ pode ser vista como uma sucessão de variáveis pseudoaleatórias uniformemente distribuídas entre 0 e 1 e pode, portanto, ser utilizada para a geração de uma variável aleatória com essa característica (ROSS, 2002).

Para a geração de variáveis pseudoaleatórias obedecendo a uma distribuição qualquer, pode-se utilizar o Método da Transformada Inversa, no qual para gerar uma variável aleatória discreta X com distribuição de massa $p_j, j > 0$ (isto é, $P\{X = x_i\} = p_i$, com $j = 0, 1, \ldots$ e $p_1 + p_2 + \ldots = 1$), utiliza-se o seguinte procedimento:

a) Gera-se uma variável U uniformemente distribuída entre 0 e 1 pelo método do Gerador Congruente Linear e, caso $U < p_0$, define-se $X = x_0$. Caso contrário:

b) Gera-se, da mesma forma, uma variável aleatória U, e encontra-se o menor número natural j tal que tal que $p_0 + p_1 + \ldots + p_{j-1} < U < p_0 + p_1 + \ldots + p_j$ e, ao encontrá-lo, define-se $X = x_j$ e retorna-se ao passo a), até que tenha sido gerada uma quantidade satisfatória de valores de X e o procedimento é finalizado.

Há ainda o Método da Aceitação-Rejeição, que parte do princípio que se é capaz de gerar uma variável pseudoaleatória discreta Y com distribuição $q_j, j > 0$, por algum método conhecido e que quer-se gerar outra variável X com distribuição $p_j, j > 0$. Nesse caso, determina-se o menor real c tal que $p_j \leq cq_j$ para todo j = 0,1,... e segue-se o seguinte algoritmo:

a) Simular um valor para Y (método conhecido) e para U (Método Congruente Linear);

b) Se $U < p_Y/cq_Y$, define-se $X = Y$ e, o algoritmo encerra. Caso contrário, voltar para a).

Tanto o Método da Transformada Inversa quanto o Método da Aceitação-Rejeição foram apresentados para o caso discreto, mas possuem também análogos para o caso contínuo, utilizando a função densidade de probabilidade da variável aleatória a ser considerada, ainda utilizando, porém, o Método Congruente Linear como auxiliar (ROSS, 2002).

3.2.2 SIMULAÇÃO DE EVENTOS DISCRETOS

Em modelos de processos ou de fenômenos da realidade, é bastante comum que sua evolução ao longo do tempo envolva estruturas lógicas complexas, de modo que nem sempre é evidente como manter o controle dessa evolução e determinar os parâmetros de interesse no sistema. Uma abstração baseada no conceito de eventos discretos foi criada no intuito de auxiliar o acompanhamento de um modelo ao longo do tempo, além de determinar os valores de interesse no sistema. A abordagem de simulação baseada nessa abstração é chamada de abordagem de simulação de eventos discretos (ROSS, 2002).

Uma simulação de eventos discretos é fundamentalmente constituída por dois elementos: variáveis e eventos. Normalmente, são utilizados os seguintes 3 tipos de variáveis:

- Variável de tempo t: representa o tempo de simulação que se passou;
- Variáveis de contagem: registram quantas vezes um evento ocorreu desde o início da simulação até o tempo de simulação t;
- Variável de estado: descreve o estado do sistema no instante t.

Sempre que um evento ocorrer, os valores das variáveis acima serão atualizados e quaisquer parâmetros de interesse no sistema serão tratados como valores de saída da simulação e serão coletados. Deverá ser mantida ainda uma lista com a relação dos eventos da simulação e os instantes de simulação em que irão ocorrer. Os atos de atualizar as variáveis de tempo, de contagem e armazenar os parâmetros de interesse são, portanto, o que se pode denominar "acompanhamento" do sistema à medida que este evolui no tempo (ROSS, 2002).

Como exemplo de simulação de eventos discretos e utilizando-se de métodos quaisquer de geração de números pseudoaleatórios, pode-se tomar a simulação de um sistema de atendimento de clientes com um único atendente. Nesse sistema, tem-se que o tempo entre a chegada de 2 clientes e o tempo de atendimento de um cliente são variáveis aleatórias cada uma com distribuição de probabilidade específica. Haverá um tempo T fixo após o qual não será mais permitida a entrada de novos clientes, e tem-se interessados em determinar o tempo médio de atendimento de um cliente e o tempo médio até que o último cliente saia. As variáveis do sistema serão as seguintes:

- Variável de tempo: t, inicializada como 0;
- Variáveis de contagem: N_C é o número de chegadas ao tempo t e N_S é o número de saídas ao tempo t, e ambas inicializadas como 0;
- Variável de estado: n é o número de clientes no sistema ao tempo t, e será inicializada como 0.

Neste exemplo, haverá dois tipos de eventos: chegadas e partidas. Portanto, a cada chegada ou saída de cliente, N_C, N_S e t serão atualizadas e os valores de interesse serão armazenados. A lista de eventos conterá o tempo da próxima chegada e da próxima saída. Para determinar quantas vezes deveremos executar essa simulação para uma estimativa confiável dos parâmetros de interesse, é necessário um estudo especial de análise estatística, que será visto na próxima subseção.

3.2.3 ANÁLISE ESTATÍSTICA

Um estudo de simulação geralmente envolve a determinação de algum parâmetro m conectado a um modelo estocástico particular. Nas simulações deste estudo, deverá existir uma variável aleatória de saída X, tal que m é o valor esperado para X. Dessa forma, a cada execução da simulação, independentemente da anterior, registra-se um valor para X_i de modo que, após n simulações, tem-se que a variável aleatória $\overline{m} = (\sum_{i=1}^{n} X_i)/n$ é um estimador para o parâmetro m. Neste caso, a variável $\overline{m}$ é dita a média da amostra. Outra variável importante é variância da amostra $\overline{v}^2$: $\overline{v}^2 = (\sum_{i=1}^{n}(X_i - m)^2)/(n-1)$, que é um estimador para a variância da população $(\sigma)^2$.

Para estabelecer um critério para um valor adequado de n para se obter um estimador confiável para m, pode utilizar-se uma aproximação para valores altos de n. Sabendo que o valor esperado de $\overline{p}$ $(E[\overline{p}])$ é igual a p e que a variância de $\overline{p}$ $(Var[\overline{p}])$ é igual a $\frac{\sigma^2}{n}$,

podemos utilizar o teorema do limite central para afirmar que, para valores altos de n temos, aproximadamente:

$$\frac{\sqrt{n}(\overline{p}-p)}{\sigma} \sim N(0,1)$$

Como também é natural deduzir que nem sempre tem-se o valor de σ em mãos, deve-se substituí-lo por seu estimador $\overline{v}$, de modo que tem-se, também para valores altos de n, de maneira aproximada, que:

$$\frac{\sqrt{n}(\overline{p}-p)}{\overline{v}} \sim N(0,1)$$

Defina-se agora as variáveis a e z_a , tais que $0 < a < 1$ e $PZ < z_a = a$, onde Z é uma variável aleatória com distribuição normal padrão. Dessa maneira, tem-se que:

$$P\{-z_{\frac{a}{2}} < Z < z_{\frac{a}{2}}\} = 1 - a$$

Como considerou-se que a variável aleatória $(\sqrt{n}(\overline{p}-p))/\overline{v}$ também segue uma distribuição normal padrão, tem-se, consequentemente, que:

$$P\{-z_{\frac{a}{2}} < \frac{\sqrt{n}(\overline{p}-p)}{\overline{v}} < z_{\frac{a}{2}}\} = 1 - a \text{ , donde :}$$

$$P\{\overline{p} - \frac{(z_{\frac{a}{2}})\overline{v}}{\sqrt{n}} < p < \overline{p} + \frac{(z_{\frac{a}{2}})\overline{v}}{\sqrt{n}}\} = 1 - a$$

Dessa maneira, encontrou-se um método estatístico para determinar o quão confiável é o estimador $\overline{p}$ para a estimativa de p, dado o número execuções feitas da simulação em questão, uma vez que podemos interpretar a última equação acima como "o valor de p está entre $\overline{p} - z_{\frac{a}{2}}\overline{v}/\sqrt{n}$ e $\overline{p} + z_{\frac{a}{2}}\overline{v}/\sqrt{n}$ com $(1-a) \times 100\%$ de certeza".

Pode-se, ainda, considerar uma situação distinta, em que o número n de execuções da simulação pode ser estendido tanto quanto se queira, e que quer-se determinar um valor satisfatório para n a partir do qual não precisaremos mais realizar novas execuções. Uma solução é escolher valores para a e L, e gerar execuções da simulação até que a probabilidade de p estar entre $\overline{p} - z_{\frac{a}{2}}\overline{v}/\sqrt{n}$ e $\overline{p} + z_{\frac{a}{2}}\overline{v}/\sqrt{n}$ é maior ou igual a $1-a$ e que o comprimento do intervalo de confiança $2(z_{\frac{a}{2}})\overline{v}/\sqrt{n}$ é menor que L (ROSS, 2002).

3.3 DESAFIOS PRÁTICOS DE SIMULAÇÃO

A presente seção tem por objetivo discorrer sobre os desafios associados à simulação de ataques *botnets*, apresentando um modelo estatístico que visa oferecer uma noção quantitativa do crescimento das populações infectadas pelos ataques.

Um dos principais desafios associados à criação de um modelo de simulação o esforço para que este seja o mais genérico possível. A dificuldade dessa tarefa se deve simplesmente à extensa variedade de tipos de *botnet* e de seus meios de propagação. Já que a implementação de um sistema totalmente completo e abrangente é inviável em curto e médio prazo, é necessário que seja definido um modelo sintético para as formas de atuação dessas redes. Assim, a simulação reproduzirá apenas alguns aspectos dentre todos os existentes em *botnets* reais. Essas restrições incluem especificar quais interfaces do hospedeiro poderão ser atacadas, de que maneira cada uma será atacada, quais técnicas de ataque serão adotadas e quais combinações de ataques serão consideradas as mais perigosas.

3.4 MODELOS EXISTENTES PARA SIMULAÇÃO DE *BOTNETS*

As cinco principais fases associadas à atividade de uma *botnet* são contaminação primária, contaminação secundária, conexão, ataque e atualização. Serão apresentadas nesta seção alguns dos modelos já existentes para uma a simulação de algumas dessas fases.

3.4.1 CONTAMINAÇÕES PRIMÁRIA E SECUNDÁRIA

Em (STYTS, 2008) é proposta a utilização de um modelo epidemiológico já consolidado na comunidade científica, o qual se utiliza do diagrama da figura 3.1 para esquematizar o grupo de elementos-fonte da infecção e os caminhos possíveis para a contaminação. Ao se construir uma simulação de uma *botnet*, pode-se tomar como base a lógica desse esquema para, através de algum modelo estatístico a ser elaborado, determinar o padrão de crescimento da população contaminada.

Na figura 3.1, o identificador "M" se refere à classe dos bebês com imunidade passiva herdadas da mãe. Aplicando esse conceito no contexto das *botnets*, a classe M representaria a classe dos computadores que não foram infectados por qualquer malware, mas que podem ser explorados para a manifestação de atividades *bot* (infecção primária).

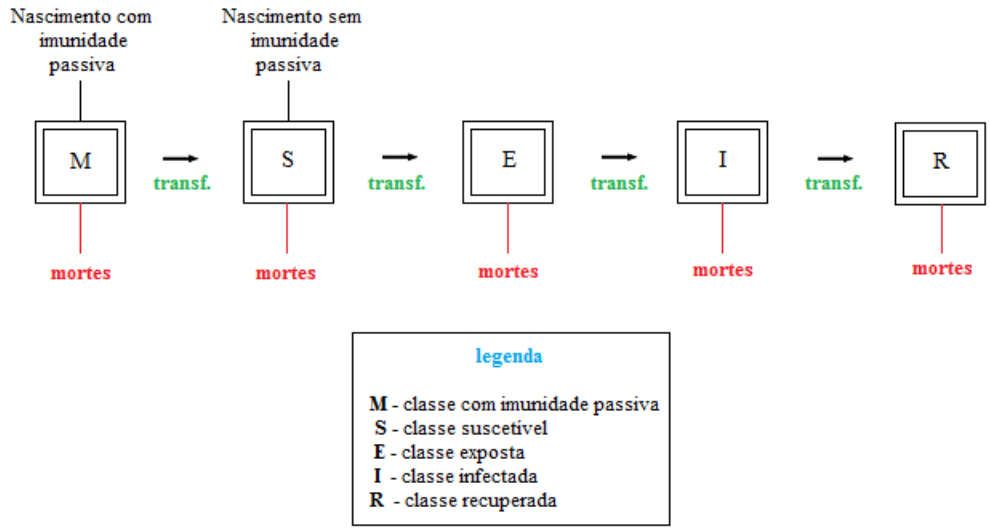

FIG. 3.1: Simulação, adaptada de (BANKS, 2001)

A classe S representa os bebês que não apresentam imunidade passiva, o que, no contexto das *botnets*, representa os computadores que são infectados e que podem ser utilizados para transmitir atividades *bot* (infecção secundária).

A classe E representa a classe dos bebês (computadores) que foram infectados, não são capazes de transmitir a infecção e nos quais a infecção não foi ainda detectada. A classe I é o conjunto dos bebês (computadores) que foram infectados, são capazes de transmitir a infecção e nos quais a infecção não foi detectada. Por fim, a classe R representa os computadores que foram infectados, tiveram sua infecção detectada e tiveram sua infecção (*bot*) removida.

O modelo deve ainda levar em consideração certas possibilidades anômalas ou menos comuns. Uma delas é a de um computador passível de contaminação não ser infectado por não ter sido exposto a um *bot* específico que caracterizaria a infecção. Outra delas é a de um computador ser infectado por um conjunto de *bots* os quais, isoladamente, não seriam capazes de provocar infecção.

3.4.2 CONEXÃO, ATAQUE E ATUALIZAÇÃO

Ao contrário das fases de contaminação primária e secundária, as fases de conexão, ataque e atualização não possuem, até então, propostas de modelos formais de funcionamento. Existe, entretanto, um *framework* já implementado para a emulação das fases de conexão, ataque e atualização de *botnets* - o *Rubot* - que será analisado a partir da próxima seção.

4 *RUBOT*

O *Rubot* é definido pelo seu desenvolvedor como um *framework* para análise e simulação de *botnets* que permite a pesquisadores implementem suas próprias redes com o intuito de realizar experimentos nelas (LEE, 2009). Seu código foi feito na linguagem *Ruby* junto com a utilização de *scripts* escritos em *Shell Script* para automatizar alguns processos.

Para se validar um modelo teórico são necessários dados, que podem ser obtidos a partir de simulações ou de situações reais. Para situações reais é difícil coletar dados pois essas redes são construidas para serem imperceptíveis, além disso, existem alguns riscos intrísecos, como ter sua máquina infectada durante a coleta. Também se deve atentar para questões legais, já que pode-se capturar um trafégo de alguma rede privada sem a devida autorização. Para a simulação é necessário uma ferramenta que simule tais redes, e é ai que o *Rubot* se faz necessário. O motivo para o uso específico do *Rubot* se deve ao fato dele ser o único *framework* conhecido e publicado que se tem acesso, e mesmo assim, seu acesso é restrito, tendo que ser solicitado ao seu desenvolvedor.

Estando de posse do código do *Rubot* pôde-se realizar algumas análises e experimentos. Primeiro foi realizado uma análise do seu código, verificando quais *botnets* estavam implementadas e quais delas poderiam ser usadas como referência para o estudo. Após a análise, selecionou-se uma *botnet* específica para testar sua execução, no caso, a *Nugache*. Sua escolha foi baseada na facilidade de execução e no melhor entendimento do seu funcionamento. O experimento foi realizado com sucesso utilizando máquinas virtuais simulando *bots* e uma física simulando o *botmaster*, seguindo um *script* que auxilia na população da *botnet*.

Ainda não se sabe se ele simula *botnets* de forma eficiente ou mesmo fiel ao modelo teórico, ou se é somente uma gerador de pacotes aleatório, por isso esse trabalho está sendo feito. O que não se pode negar é o fato de ser uma ferramenta inovadora, que se comprovada eficaz, pode mudar o rumo dos estudos de simulação de *botnets*.

4.1 ARQUITETURA

O *Rubot* é um *framework* feito na linguagem *Ruby* que contém um núcleo responsável por carregar a configuração, inicializar os componentes e realizar a comunição entre eles, já os componentes são responsáveis pelo comportamento do *bot* (LEE, 2009). O núcleo é representado por uma classe chamada de *Engine*, que possui duas subclasses, a *Host* e *Bot* que não diferem em nada do núcleo, são somente apelidos usados para o melhor entedimento. As classes *Host* e *Bot* simplesmente provêm serviços para os componentes, como passagem de messagens (LEE, 2009). Os componentes são agrupados em quatro categorias: Vulnerabilidades, Ataques, Atualização e Comunicação do CC. Para melhor ilustrar a arquitetura do *Rubot*, na figura 4.1 será mostrado o diagrama UML dos objetos do *framework*.

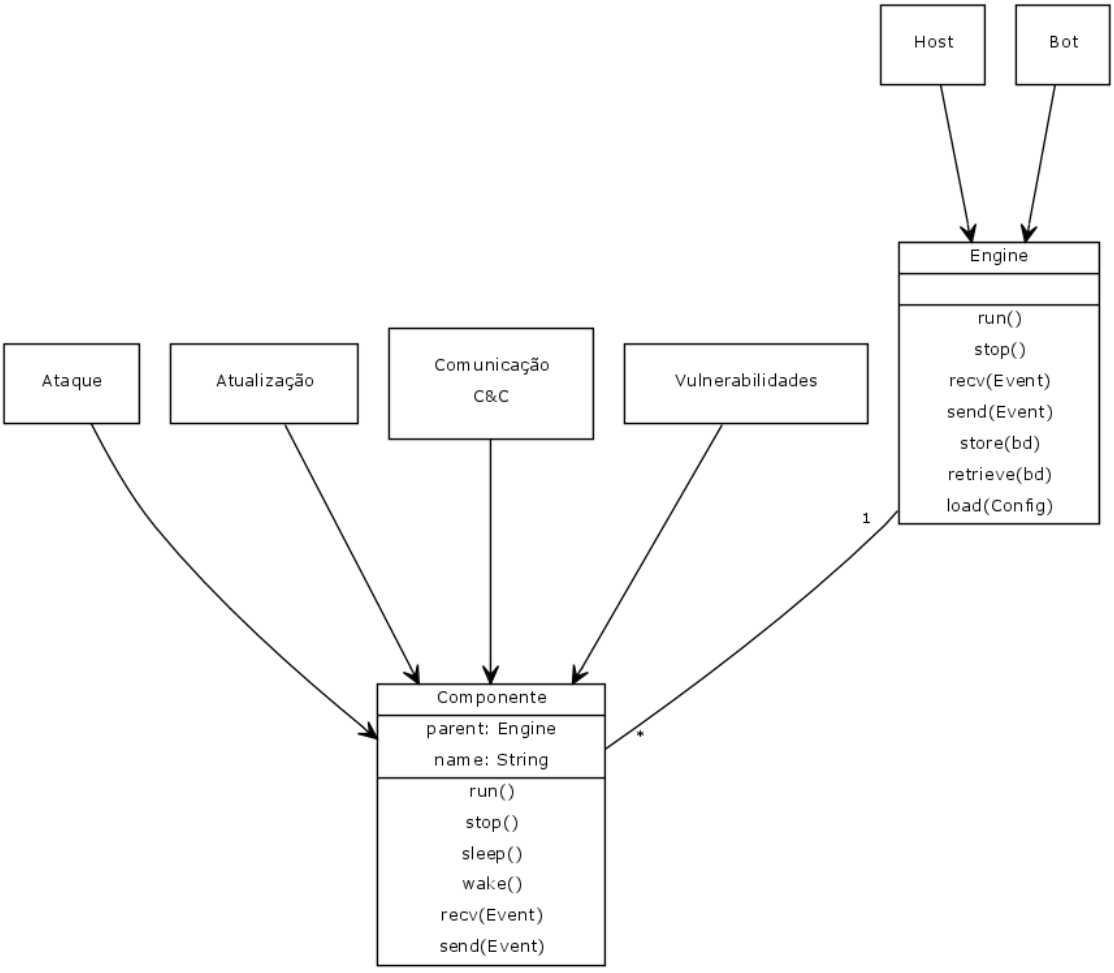

FIG. 4.1: Diagrama UML do *Rubot*, adaptada de (LEE, 2009)

4.2 CARACTERÍSTICAS

O *Rubot* possui algumas características, entre as principais está o fato de todos os componentes de uma *botnet* estarem em pacotes separados, que podem ser usados por um pesquisador para construir a *botnet* que ele desejar (LEE, 2009). Tal tarefa porém demanda um maior tempo e requer um maior entendimento do código, que é pobremente documentado, logo não será abordado nesse estudo, o que nos leva à utilização das *botnets* já implementadas pelo desenvolvedor e seus *scripts* automatizadores.

4.2.1 *BOTNETS* IMPLEMENTADAS

O *Rubot* (LEE, 2009) vem com alguns experimentos (é o nome dado pelo autor para as botnets implementadas) documentados, que variam de *worms* a *botnets* tradicionais, porém o que tem-se é teoria, nada foi testado. Entre todos os experimentos, os mais bem implementadas são:

- *Nugache*: Uma das primeiras *botnets* P2P(*peer-to-peer*) a despertar interesse nos pesquisadores de segurança da informação (LEE, 2009), seu uso é bem simples e não requer muita infraestrutura;

- *Storm*: Uma complexa *botnet*, necessita de bastante infraestrutura de rede, como um *proxy*, um servidor *web* e um nó subcontrolador, também é uma *botnet* descentralizada.

4.2.2 *SCRIPTS* AUTOMATIZADORES

O *framework* possui *scripts*, em alguns experimentos, que podem automatizar especificamente a população da *botnet*, facilitando assim uma tarefa muito dispendiosa. O que eles fazem, basicamente, é popular a *botnet*, ou seja, criar toda a estrutura de forma automatizada. Nem todos os experimentos implementados possuem os *scripts*, ele é opcional e somente é uma ajuda, ou seja, pode-se criar a estrutra manualmente sem probelmas nenhum. Por exemplo, a *botnet Nugache* vem com um *script* que entra em todas as máquinas definidas dentro dele e executam o *malware* específico da *Nugache*, por final, ele faz com que o *botmaster* comece a enviar pacotes para uma máquina também pré-definida.

O *Rubot* não possue uma sistema de coleta de informação, cabendo ao pesquisador procurar a solução que lhe atenda, sendo uma das melhores formas de se coletar dados

a utilização de um *sniffer* como o *TCPDump*, que coleta o fluxo de mensagens de uma determinada interface de rede.

5 EXPERIMENTO COM A *NUGACHE* NO *RUBOT*

Nesse capítulo, será apresentada a *botnet p2p Nugache* e a análise de seu funcionamento no *framework Rubot*. Para que se possa ter uma entendimento da *botnet*, na subseção 5.1, será feita uma fundamentação teórica sobre seu funcionamento. Na subseção 5.2, usando a teoria obtida sobre a *Nugache*, será feita uma análise da simulação feita no *Rubot*, citando semalhanças, diferenças e funcionalidades que faltam ao *framework*. Com isso teremos fundamentação para dizer em que aspectos o *Rubot* é um bom simulador e em quais ele deixa a desejar.

5.1 NUGACHE

A *Nugache* é uma *botnet peer-to-peer*, ou seja, descentralizada, onde suas máquinas infectadas se conectam a um número predefinido de *peers* (GRIZZARD, 2007). Ela foi desenvolvida em abril de 2006 e ataca somente computadores que utilizam o sistema operacional MS Windows (TYAG, 2011). Inicialmente foi chamada, erroneamente, de *'bot tcp/8'*, pois só utilizava a porta 8 para comando e controle, através do protocolo IRC (STOVER, 2007). Posteriormente isso deixou de existir e a *Nugache* passou a utilizar portas altas aleatórias para a comunicação, evitando assim sua detecção. Um dos seus principais propósitos é o DDoS, mas ela também é capaz de executar outras tarefas, como roubo de senhas e corrompimento de dados. A *Nugache* também respeita a característica de *botnets peer-to-peer*, e não utiliza o serviço DNS, mas em raras ocasiões ele é usado, que é quando se utiliza o protoclo IRC.

5.1.1 IMPLEMENTAÇÃO

A implementação da *Nugache* é bastante simples. Um *peer* infecta outro com um binário, esse binário pode comprometer o *Windows Registry* com uma lista de peers para ele aderir a *botnet*. Outra forma de obter essa lista é através de um procedimento de *bootstraping* que vem codificado dentro do próprio binário (STOVER, 2007). Uma vez de posse da lista de peers, a máquina infectada adere a rede, se conectando a todos os *peers* da lista. Isso gera uma arquitetura que lembra um grafo, o que a torna bastante escalável, tendo em vista que uma vez conectado a rede, o *peer* procura outras máquinas

para infectar (DAGON, 2007).

A *Nugache* emprega uma arquitetura P2P, mas possui alguns *peers* que funcionam como 'servidores', que são capazes tanto de receber quanto enviar mensagens. Determinar quais serão esses *peers*, e modifica-los na medida do tempo é uma tarefa importantíssima que dificulta bastante a detecção da rede, pois são eles os responsáveis pelo repasse das mensagens. Os servidores, além de escutar os nós da sua lista de *peers*, irão enviar mensagens para todos aqueles *peers* que os possuirem nas suas respectivas listas. Já os *peers* que somente escutam os nós da lista são chamados de clientes.

5.1.2 INFECÇÃO E COMUNICAÇÃO

A infecção inicial ocorre quando um *peer* servidor infecta a máquina vítima com a lista de *peers*, após receber a lista, a máquina infectada já está apta a aderir a *botnet*, procedimento que se inicia com a troca de chaves RSA para que a partir delas possa ser gerada uma criptografia do modo *Rjindael* 256-bit, que são chaves de sessões (existe uma para cada conexão entre *peers*) (STOVER, 2007).

Essa criptografia é uma das principais características da *Nugache*, e que de fato dificulta as tentativas de detecção do canal de controle da rede (TYAG, 2011). Uma vez que a criptografia está funcionando, o *peer* de fato entrou na rede e pode, através de um protocolo interno, executar outras funcionalidades, como definir se ele será um servidor ou atualizar a lista de peers.

Essa ultima tarefa é de suma importancia para o bom funcionamento da rede, e é feita através de um simples "PING" e "PONG" onde os *peers* servidores informam todos os novos peers conectados a rede, atualizando a lista dos nós necessários.

A comunicação entre os nós da rede se faz pelo canal criptografado, porém a *Nugache* possua a capacidade de se comunicar usando o protocolo IRC, porém seu uso é muito incomum, e raramente foi visto sendo usado na prática (STOVER, 2007).

5.1.3 ATUALIZAÇÃO

Quando a máquina vitima recebe o binário da *Nugache*, ele vem com um número que representa a versão do *malware*. Ao se conectar na rede, ele compara as versões com o *peer* ao qual se conectou para verificar a necessidade de atualização. Quando ocorre do nó cair, ao se reconectar com a rede o procedimento é repetido, ou seja, a atualização só ocorre caso o nó esteja a muito tempo inativo ou tenha saído da rede por motivos

desconhecidos e tenha em seguida, se reconectado.

5.1.4 DETECÇÃO

A detecção da *Nugache* é bastante difícil e o principal motivo é que a troca de chaves RSA que geram a criptografia *Rjindael* são dinâmicas, dificultando a descoberta de quando essa troca foi feita. É por esse motivo que se faz necessário um estudo mais aprofundado e o desenvolvimento de novas técnicas e assinaturas IDS (do inglês, Sistema de detecção de intrusos) para tentar detectar essa *botnet*. Mesmo tendo-se posse de alguns tráfegos reais dela, ainda não se tem dados suficientes para executar tal tarefa.

5.2 EXPERIMENTO DA *NUGACHE* COM O *RUBOT*

No início desta pesquisa, teve-se como um dos objetivos a avaliação do *Rubot* segundo o critério de ser um simulador confiável para o comportamento - ou para algum aspecto do comportamento - de uma *botnet*. Para uma avaliação mais consistente, julgou-se necessária a aquisição de conhecimentos básicos em análise estatística de simulações computacionais e, em particular, conhecimento de modelagens já existentes para *botnets* e de uma *botnet* específica (*Nugache*).

Realizou-se 4 experimentos com a utilização de seis máquinas interconectadas pela rede interna do IME, cada uma executando o *Rubot* e o *sniffer TCPDump*. O *software Wireshark* foi utilizado para análise posterior do tráfego gerado.

No *Rubot*, foi utilizado o módulo já existente da *botnet Nugache*. Os experimentos 1, 2, 3 utilizaram a arquitetura I (figura 5.1) e o experimento 4 utilizou a arquitetura II (figura 5.2), todos contendo as mesmas máquinas.

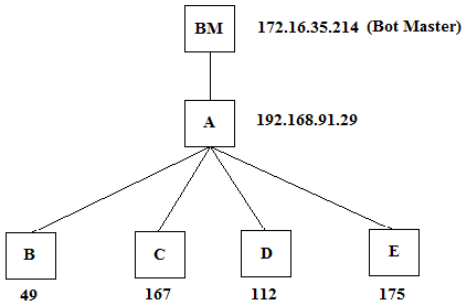

FIG. 5.1: Arquitetura I

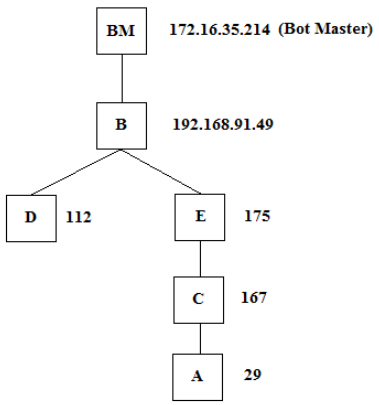

FIG. 5.2: Arquitetura II

Nas figuras 5.1 e 5.2, cada quadrado representa uma máquina identificada por uma letra e seu endereço IP. A máquina identificada por BM representa a máquina com o *botmaster*. Na figura 5.1, as máquinas B, C e D tiveram seus endereços IP abreviados para o quarto octeto pois os 3 primeiros octetos são idênticos aos da máquina A. A mesma abreviação foi feita para as máquinas D, E, C e A, cujos endereços IP possuem os 3 primeiros octetos iguais aos de B (figura 5.2).

O procedimento básico do experimento, para todos os quatro tráfegos, foi o seguinte:

a) O usuário utilizador do *Rubot* acessa cada uma das máquinas via SSH. Uma vez tendo acesso a uma máquina, ele executa o binário do *Rubot* e, assim, pode construir a topologia da rede. Essa tarefa pode ser automatizada fazendo uso de um *script* escrito em *shell script* já existente dentro do *Rubot* porém, neste trabalho, a topologia foi criada manualmente;

b) Cada comando é enviado à máquina destino via protocolo *ssh*, recebido pela máquina destino e repassado à instância do *Rubot*, a qual de se encarrega de executar a instrução;

c) Caso a *botnet* ainda não esteja inicializada, a instrução consistirá em definir a máquina destino como primeira a ser inserida na *botnet*. Nos outros casos, consistirá no envio de algum tipo de mensagem para uma ou mais máquinas já inseridas na *botnet*;

d) Todo envio de mensagem entre dois *hosts* é registrado por ambos através do *TCPDump*;

e) Ao fim de cada tráfego, os arquivos *.pcap* gerados pelo *TCPDump* em cada máquina são coletados, representados em diagrama de fluxo (com uso do *Wireshark*) e posteriormente analisados.

No experimento 1, o qual segue a arquitetura I (figura 5.1) procedeu-se da seguinte maneira :

a) Define-se, manualmente, a máquina A como a primeira a ser inserida na *botnet*;

b) Introduz-se BM à *botnet*, de forma que A ouve BM. Este passo também é feito manualmente;

c) Introduz-se, manualmente, as máquinas B, C, D e E na *botnet*. As máquinas B, C, D e E, devem ouvir A, de modo a definir a arquitetura da rede;

d) BM envia um pacote à *botnet*;

e) Retira-se B e, logo em seguida, reinsere-se B, ambos manualmente;

f) BM envia um segundo pacote à *botnet*.

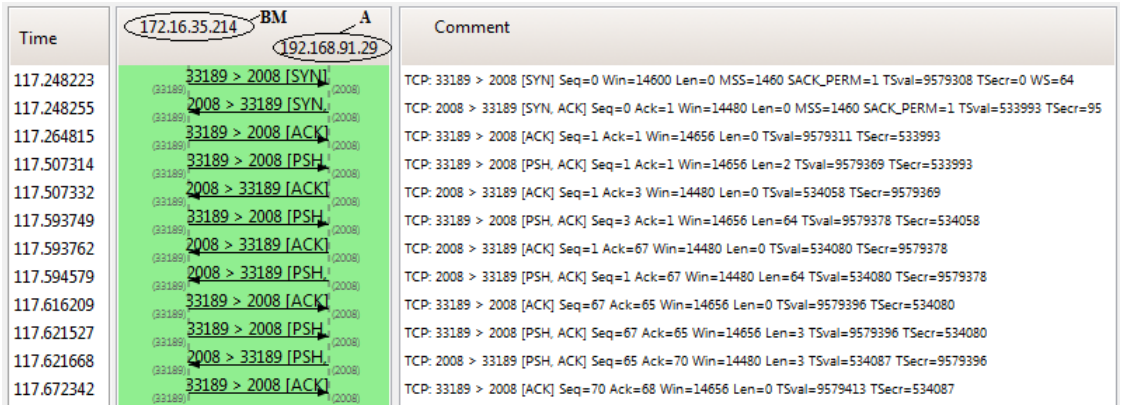

FIG. 5.3: Tráfego 1: Introdução do BM

A troca de mensagens associada à introdução de BM na *botnet* é vista na figura 5.3. A natureza das mensagens durante a introdução de outras máquinas ocorre de maneira idêntica nos quatro tráfegos, de modo que não há necessidade de exibir cada caso. O mesmo ocorre com as mensagens de envio de pacote (figura 5.4) e de retirada de B da *botnet* (figura 5.5).

Nota-se, na figura 5.3, que a quantidade de mensagens associadas à introdução de uma máquina à *botnet* é relativamente alta em relação as demais trocas de mensagens,

como por exemplo, a troca para executar um comando, que são somente duas, bem menos que que as doze mostradas. Pode-se supor que esse número elevado de mensagens esteja relacionado à criptografia realizada pelo *Rubot*, porém não se tem evidências explícitas no diagrama (figura 5.3). Além disso, ao introduzir uma máquina na *botnet*, a porta na qual ela irá responder às mensagens é elevada e é escolhida aleatoriamente pelo *Rubot*, comportamento este também previsto em (STOVER, 2007).

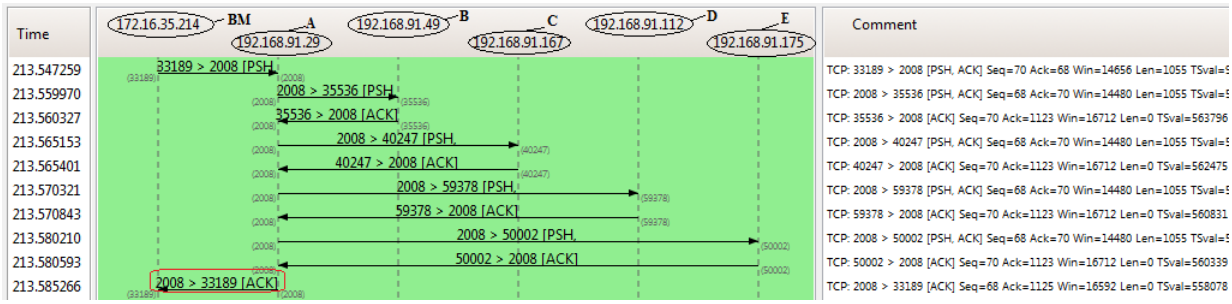

FIG. 5.4: Tráfego 1: Envio do primeiro pacote

Na figura 5.4, nota-se que a quantidade de mensagens associadas ao envio de um pacote é pequena, consistindo apenas de uma *flag* PSH para envio e outra ACK para confirmação. Em particular nesse tráfego, observa-se que a máquina A, mais central, recebe uma *flag* PSH e só envia a confirmação ACK de volta a BM após enviar o pacote para todas as máquinas às quais está conectada (em destaque na figura 5.4). No envio de pacote, o intervalo de tempo entre cada um dos envios de A para seus nós filhos é sempre pequeno, donde se conclui que a distribuição de pacotes para nós filhos ocorre sempre que possível, sem qualquer espera.

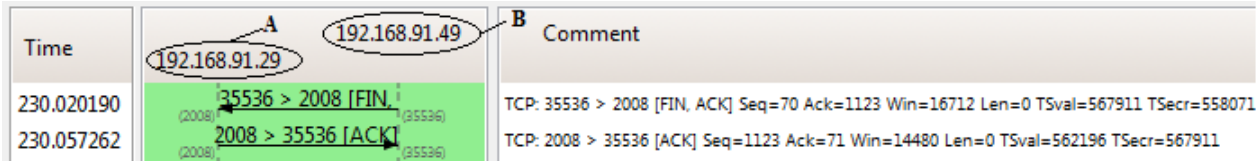

FIG. 5.5: Tráfego 1: Retirada da mÃ¡quina B

Na figura 5.5, observa-se que a troca de mensagens associada à retirada da máquina B é relativamente simples e consiste apenas em uma *flag* FIN partindo da máquina que se deseja retirar e uma *flag* ACK de confirmação partindo da máquina diretamente ligada à primeira.

Na figura 5.6, é exibido o envio do segundo pacote, logo após a máquina B ser reinserida na *botnet*. Em termos de natureza das mensagens, o envio do segundo pacote

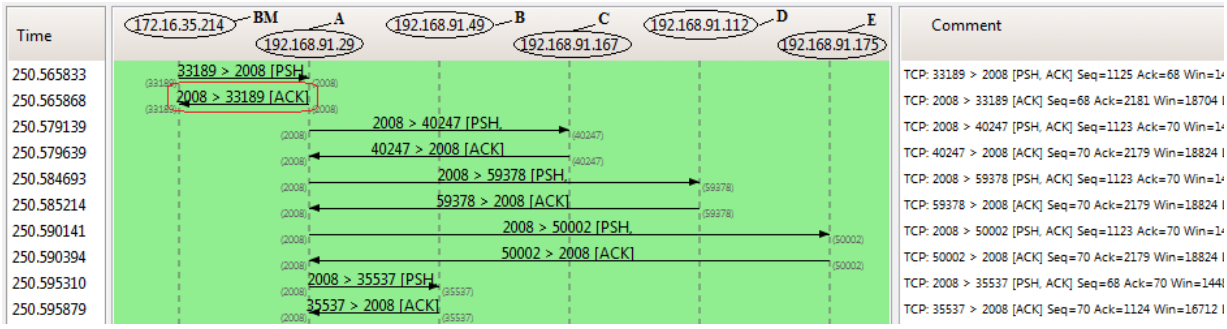

FIG. 5.6: Tráfego 1: Envio do segundo pacote

é idêntico ao envio do primeiro, porém a ordem em que eles foram mandados mudou e, além disso, dessa vez A envia uma confirmação ACK para BM (mensagem destacada na figura 5.6) antes de enviar o pacote para as outras máquinas.

A mudança de ordem se explica pela retirada e reinserção de B, que fez com que B fosse para o fim da fila de *peers* armazenada em A. O envio prematuro do ACK indica que o BM nem sempre possui a garantia de que A envia o pacote para todas as máquinas a ela conectadas.

No experimento 2, que também segue a arquitetura I (figura 5.1), procedeu-se da seguinte forma:

a) Seguiu-se exatamente os passos a), b), c) e d) do procedimento para o experimento 1;

b) Retira-se A;

Nesse experimento, todas as mensagens associadas à inserção de máquinas na *botnet*, envio do primeiro pacote e retirada de máquina se comportaram de maneira inteiramente análoga ao tráfego 1 porém, logo após A (máquina mais central), ser retirada da *botnet*, BM passou a receber erros ao tentar enviar mensagens. Isso indica que o canal de transmissão entre BM e a *botnet* era de fato estabelecido exclusivamente pela conexão BM-A e, ao ser desfeito, a comunicação se tornou impossibilitada.

No experimento 3, que ainda segue a arquitetura I (figura 5.1), procedeu-se da seguinte forma:

a) Seguiu-se exatamente os passos a), b), c) e d) do procedimento para o experimento 1;

b) Insere-se e, logo em seguida, reinsere-se BM;

c) Envia-se o segundo pacote.

Nesse experimento, a inserção, envio do primeiro pacote e retirada de máquina se comportaram de maneira inteiramente análoga aos experimentos 1 e 2. Com a retirada e seguida reinserção de BM, a *botnet* voltou a se comportar da mesma forma de anteriormente, e o envio do segundo pacote ocorreu normalmente, indicando que a saída do *botmaster* não implica na dissolução da *botnet*.

Por fim, no experimento 4, agora obedecendo a arquitetura II (figura 5.2), procedeu-se da seguinte forma:

a) Define-se B como a primeira máquina a ser inserida na *botnet*;

b) Introduz-se as máquinas D e E na *botnet*, de modo que ambas ouçam B;

c) Introduz-se C ouvindo E, e A ouvindo C, definindo a arquitetura da rede;

d) BM envia um pacote à *botnet*.

Neste experimento, todas as mensagens básicas de inserção e envio ocorreram de maneira inteiramente análoga ao ocorrido nos experimentos 1, 2 e 3. O novo fato deste experimento é a introdução da arquitetura II (figura 5.2), que possui um nível a mais que a arquitetura I (figura 5.1), e o fato de o pacote ter chegado a todas as máquinas indica que a rede criada não possui estrutura hierárquica, o que é também previsto em (STOVER, 2007).

5.3 ANÁLISE DOS EXPERIMENTOS COM O *RUBOT*

Independentemente de cada um dos 4 experimento há, primeiramente, dois pontos importantes a serem observados. O primeiro é o fato de que tanto a atividade de inserção de um *host* na *botnet* quanto a de envio de pacote entre *hosts* são iniciadas manualmente ou através de algum *script* a ser escrito pelo usuário e executado na máquina do botmaster. Isso leva à conclusão de que o *Rubot* não é um simulador computacional, já que toda iniciativa de introduzir/retirar uma máquina ou trocar mensagens parte exclusivamente de um ser humano, seja realizando um comando manual ou introduzindo um *script* independente do *Rubot*.

O segundo ponto é o fato de que, segundo Stover, o funcionamento de uma *botnet Nugache* prevê a utilização de criptografia nas mensagens. Apesar de essa característica não poder ser explicitada nos diagramas de fluxo em cada tráfego, foi-se observada no

código fonte do *Rubot* a utilização de algoritmos de criptografia como RSA e *Rijndael*, também previstos em Stover.

Outra observação importante sobre os tráfegos descritos acima é o fato de que, ao analisá-los utilizando o *Wireshark*, aplicou-se filtros para que apenas mensagens do protocolo TCP fossem exibidas. Apesar dessa restrição, observou-se que a comunicação entre as máquinas da *botnet* é independente do protocolo DNS, o que impossibilita a identificação do tráfego da *botnet* baseada em DNS e está de acordo com o comportamento de uma *Nugache*, segundo Stover.

Em última análise, conclui-se que o *Rubot* não é propriamente um simulador de uma *botnet*, de modo que validações estatísticas de simulação computacional não se aplicam à análise de seu funcionamento. Em contrapartida, o *Rubot* é capaz de emular os estados dos *hosts* assim como a maneira como ocorre a comunicação entre eles, desde que o próprio usuário insira comandos (inserção, remoção ou envio de pacote) de forma manual ou mesmo através de um *script* por ele desenvolvido.

Por fim, considerando aspectos como criptografia, estrutura não hierárquica, escolha de portas de escuta e independência do protocolo DNS, pode-se afirmar que, no caso da *botnet Nugache*, o *Rubot* é capaz de realizar uma emulação de maneira fiel à realidade.

6 CONCLUSÃO

É evidente que as *botnets* são, na atualidade, a mais séria ameaça à segurança de dados na Internet. Seu poder já se mostrou capaz de por em risco um país inteiro, ameaçando o funcionamento de diversas instituições como bancos, ministérios e o parlamento, como se pôde observar no ataque ocorrido em abril de 2007 na Estônia, Leste Europeu.

A falta de sistemas operacionais com mecanismos de defesa eficientes contra esse tipo de ameaça deve ser encarada como um alerta aos industriais e acadêmicos na área de produção de software.

Novas pesquisas em simulação e defesa contra esse tipo de ameaça devem ser feitas o quanto antes já que, muito embora as *botnets* tenham já alcançado resultados expressivos, seu potencial malicioso pode ainda ser bastante explorado, o que poria em sério risco o funcionamento da Internet, afetando, em última análise, todo o planeta.

Um dos aspectos principais a serem analisados nesse estudo é a criação de um modelo de simulação para um ambiente de ataque e defesa associados às *botnets*. Esta concepção envolve diversos desafios, primeiramente pelo fato de que esse tipo de rede já atinge um alto grau de flexibilidade e versatilidade, o que torna difícil o desenvolvimento de um sistema totalmente genérico. Ainda assim, qualquer modelo simplificado, envolvendo apenas os aspectos mais fundamentais de uma simulação é ainda muito complexo e não pode ser feito em curto ou médio prazo.

Neste trabalho foram apresentados os conceitos básicos associados à classificação das *botnets*, seus principais componentes, seus objetivos mais comuns, suas etapas de ciclo de vida e seus tipos de arquitetura, além de alguns dos aspectos a serem analisados quanto a seus modelos de simulação.

Para que a tarefa de modelagem possa ser feita com mais coerência, utilizou-se de alguma ferramentas, uma delas foi o aprofundamento do estudo de simulação, conhecimento que permite validar ou não um modelo segundo padrões pré-estabelecidos. O *framework Rubot* foi outra ferramenta usada, ele está sendo inovador no que tange a simulação e análise de *botnets*, único em sua área. Entretando, ainda não foi testado suficientemente para que se comprove ser um simulador fiel de *botnets*, e é ai que o estudo aprofundado de simulação se faz presente.

A *botnet Nugache* foi escolhida para ser testada no *Rubot* gerando dados necessários

para comprovar sua eficácia como simulador, mas antes foi feito uma fundamentação teórica da *botnet* e assim tirar métricas para tal comprovação. Após a coleta de dados, foi atestado que o *Rubot* não simula a *botnet* como um todo, sendo necessário auxilio de scripts para essa tarefa. Constatou-se também que, apesar de não simular toda a *botnet*, ele consegue simular a troca de menssagens com bastante proximidade, seguindo parâmetros essenciais visto na teoria, e que com alguns ajustes, o *framework* conseguiria sim, simular o cenária inteiro.

Após a análise dos experimentos realizados com o *framework Rubot*, conclui-se que este já apresenta bastantes funcionalidades úteis, principalmente no que se refere na forma como os *bots* se conectam. Tendo isso em vista, sugere-se como trabalhos futuros o desenvolvimento de outro *framework*, ou a melhoria do *Rubot*, que simule uma *botnet* por completo, ou seja, todas as suas fases do ciclo de vida. Também é sugerido, partindo dessa simulação, o desenvolvimento da modelagem de uma *botnet* real.

7 REFERÊNCIAS BIBLIOGRÁFICAS

AGUIAR, A. S. C., FERNANDEZ, M. P., SILVA, J. L. C. e BEZERRA, J. M. Análise comparativa de simuladores de redes baseados em pacotes versus simuladores utilizando abstração de fluidos. *26 Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, 2008.

BANKS, J., II, J. S. C., NELSON, B. L. e NICOL, D. M. *Discrete-Event System Simulation*. Prentice Hall, 2001. ISBN 0-13-088702-1.

CHOI, H., LEE, H. e KIM, H. BotGAD: detecting botnets by capturing group activities in network traffic. Em *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE*, COMSWARE '09, pág. 2:1–2:8, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-353-2.

COOKE, E., JAHANIAN, F. e MCPHERSON, D. The zombie roundup: understanding, detecting, and disrupting botnets. Em *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop*, pág. 6–6, Berkeley, CA, USA, 2005. USENIX Association.

DAGON, D., GU, G., LEE, C. P. e LEE, W. A taxonomy of botnet structures. pág. 8. Georgia Institute of Technology, 2007.

DAINOTTI, A., PESCAPÉ, A. e VENTRE, G. A packet-level characterization of network traffic. *11th International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks*, 2006.

FEILY, M., SHAHRESTANI, A. e RAMADASS, S. A survey of botnet and botnet detection. Em *Emerging Security Information, Systems and Technologies, 2009. SECURWARE '09. Third International Conference on*, págs. 268 –273, junho 2009.

FOSSI, M., EGAN, G. Y., HALEY, K., JOHNSON, E., MACK, T., ADAMS, T., BLACKBIRD, J., LOW, M. K., MAZUREK, D., MCKINNEY, D. e WOOD, P. Symantec internet security threat report - trends for 2010. Technical Report Volume 16, Symantec, abril 2011.

GRIZZARD, J. B., SHARMA, V., NUNNERY, C., KANG, B. B. e DAGON, D. Peer-to-peer botnets: Overview and case study. Usenix Association, abril 2007.

IANELLI, N. e HACKWORTH, A. Botnets as a vehicle for online crime - CERT. *CERT© Coordination Center*, dezembro 2005. Carnegie Mellon University.

KOTENKO, I., A., K. e SHOROV, A. Agent-based modeling and simulations of botnets and botnet defense. *Conference on Cyber Conflict Proceedings*, 2008.

KOTENKO, I., KONOVALOV, A. e SHOROV, A. Discrete-event simulation of botnet protection mechanisms. Em *Discrete Event Simulations - Development and Applications*, pág. 146–168. Eldin Wee Chuan Lim (Ed.), 2012.

LEE, C. P. Framework for botnet emulation and analysis. Atlanta, GA, USA, 2009.

MICRO, T. Taxonomy of botnet threats. Technical report, Trend Micro White Paper, 2006.

ROSS, S. M. *Simulation*. Academic Press, Departament of Industrial Engineering and Operations Research, University of California, Berkley, California, 2002.

SILVA, S., SILVA, R., PINTO, R. e SALLES, R. Botnets: a survey. *Computer Networks*, 2012.

STOVER, S., DITTRICH, D., HERNANDEZ, J. e DIETRICH, S. Analysis of the storm and nugache trojans - p2p is here. pág. 32. Login, dezembro 2007.

STYTS, M. R. e BANKS, S. B. Challenges of modeling botnets for military and security simulations. *SimTecT 2008 Simulation Conference: Simulation – Maximising Organizational Benefit*, 2008.

TYAG, A. K. e AGHILA, G. A wide scale survey on botnet. *International Journal of Computer Applications*, 34, novembro 2011.

WANG, P., SPARKS, S. e ZOU, C. C. An advanced hybrid peer-to-peer botnet. Em *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pág. 2–2, Berkeley, CA, USA, 2007. USENIX Association.

ZHU, Z., LU, G., CHEN, Y., FU, Z. J., ROBERTS, P. e HAN, K. Botnet research survey. Em *Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International*, págs. 967 –972, 2008.