INSTITUTO MILITAR DE ENGENHARIA

FERNANDO HENRIQUE DE SOUZA MACEDO

DESENVOLVIMENTO DE DEFESA PROATIVA CONTRA ATAQUES CIBERNÉTICOS DO TIPO SMURF E SYN FLOOD

Projeto Final de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do diploma de Engenheiro de Computação.

Orientador: Cap Anderson Fernandes Pereira dos Santos – D. Sc.

Rio de Janeiro

2010

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

Este exemplar é de propriedade do Instituto Militar de Engenharia, que poderá incluí-lo em base de dados, armazenar em computador, microfilmar ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es) e do(s) orientador(es).

003.5 Macedo, Fernando Henrique de Souza
M141 Desenvolvimento de Defesa Proativa contra Ataques Cibernéticos do Tipo Smurf e SYN Flood / Fernando Henrique de Souza Macedo. – Rio de Janeiro: Instituto Militar de Engenharia, 2010.

44 p. il.

Projeto Final de Curso (Graduação) – Instituto Militar de Engenharia – Rio de Janeiro, 2010

1. Cibernética. 2. Negação de serviço. I. Título. II. Instituto Militar de Engenharia.

CDD 003.5

INSTITUTO MILITAR DE ENGENHARIA

FERNANDO HENRIQUE DE SOUZA MACEDO

DESENVOLVIMENTO DE DEFESA PROATIVA CONTRA ATAQUES CIBERNÉTICOS DO TIPO SMURF E SYN FLOOD

Projeto Final de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do diploma de Engenheiro de Computação.

Orientador: Cap Anderson Fernandes Pereira dos Santos – D. Sc.

Aprovado em 20 de agosto de 2010 pela seguinte Banca Examinadora:

_____

Cap Anderson Fernandes Pereira dos Santos – D. Sc. do IME

_____

Prof[a] Raquel Coelho Gomes Pinto – D. Sc. Do IME

_____

Maj Sérgio dos Santos Cardoso Silva – M. Sc. do IME

Rio de Janeiro

2010

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

CERT.br	Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil
BGP	*Border Gateway Protocol*
DDoS	*Distributed Denial of Service*
DoS	*Denial of Service*
DRDoS	*Distributed Reflection Denial of Service*
FBI	*Federal Bureau of Investigation*
ICMP	*Internet Control Message Protocol*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
IPv6	*Internet Protocol, Version 6*
IRC	*Internet Relay Chat*
OTAN	Organização do Tratado do Atlântico Norte
RFC	*Request for Comments*
RTO	*Retransmission Timeout*
RTT	*Round Trip Time and Timeout*
SRTT	*Smoothed Round Trip Time and Timeout*
TCB	*Transmission Control Block*
TCP	*Transmission Control Protocol*
TFN	*Tribe Flood Network*
TFN2K	*Tribe Flood Network 2000*

RESUMO

Os ataques cibernéticos são uma realidade e, ao adquirirem conotações políticas e militares, motivaram uma preocupação das nações com uma possível guerra cibernética. Assim, vários países estão investindo muitos recursos em defesa contra ataques cibernéticos.

Nesse contexto, os ataques de negação de serviço visam inviabilizar o acesso a recursos através da exploração de vulnerabilidades existentes na infraestrutura de servidores ou mesmo pela sobrecarga do tráfego.

Assim, a detecção e a defesa contra esse tipo de ataque tornam-se um estudo importante e necessário. Este trabalho tem por objetivo desenvolver uma defesa proativa para dois tipos de ataque de negação de serviço: *SYN Flood* e *Smurf* e analisar sua eficiência em um ambiente com um ataque simulado.

ABSTRACT

The cyber attacks are a reality and, having political and military connotations, prompted a concern of nations with a possible cyber war. Thus, many countries are investing many resources in defense against cyber attacks.

In this context, denial of service attacks are designed to denial the access to resources by exploiting vulnerabilities in the infrastructure of servers or even by an overload of traffic.

Thus, detection and defense against such attacks become an important and necessary study. This work aims to develop a proactive defense for two types of denial of service attack: *SYN Flood* and *Smurf* and analyze their efficiency in an environment with a simulated attack.

1 INTRODUÇÃO

O uso da Internet trouxe inúmeros benefícios à humanidade, tornando-se uma ferramenta importante nas estruturas social, econômica e política mundial, porém essa importância tornou-a um alvo em potencial para atingir pessoas ou instituições.

Nesse contexto, os ataques cibernéticos se tornaram um meio de explorar vulnerabilidades de segurança em computadores e em redes, em especial na Internet, de forma a prejudicar um ou mais usuários de computadores. Segundo o Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT.br) (2007), "Um computador (ou sistema computacional) é dito seguro se este atende a três requisitos básicos relacionados aos recursos que o compõem: confidencialidade, integridade e disponibilidade".

Por confidencialidade, entende-se como a propriedade de os dados serem divulgados apenas a pessoas ou instituições autorizados. Integridade é a garantia de que dados não sejam destruídos ou modificados sem autorização. Por fim, disponibilidade é a capacidade de os recursos (sejam dados ou serviços) estarem disponíveis sempre que necessário.

Dentre os tipos de ataques cibernéticos, o ataque de negação de serviço, do inglês *Denial of Service* (*DoS*) *Attack*, é aquele que visa indisponibilizar o acesso a recursos de um sistema a seus usuários, através da sobrecarga (ou inundação) de requisições ao sistema, de forma a tornar a taxa de requisições superior à capacidade de o sistema processá-las, ou através da exploração de uma vulnerabilidade de um sistema ou de um protocolo de comunicação. Os ataques *DoS* também podem ser feitos de forma distribuída, quando há vários computadores sob o comando de um computador mestre realizando um ataque de negação de serviço, sendo conhecido como ataque distribuído de negação de serviço, do inglês *Distributed Denial of Service* (*DDoS*) *Attack*.

Portais robustos e acostumados com um número elevado de acessos, como *eBay*, *Yahoo*, *Amazon* e *CNN* já foram vítimas de ataques *DDoS*, conforme é mencionado por Sandoval e Wolverton (2000).

Assim, cogita-se que esses ataques adquiriram conotações políticas e militares, criando a possibilidade do surgimento de uma guerra cibernética entre entidades políticas. Como exemplo disso pode-se citar o ataque sofrido por vários *sites* americanos e sul-coreanos em 7

de julho de 2009, em que acusaram a Coréia do Norte de ter comandado o ataque. As principais vítimas eram *sites* governamentais desses dois países, mas páginas de entidades financeiras e de órgãos de imprensa também sofreram com os ataques, conforme menciona South (2009). Esse ataque *DDoS* envolveu uma rede com mais de cem mil agentes, que enviavam às vítimas requisições *HTTP GET* e pacotes dos tipos *UDP* e *ICMP Echo*, segundo mencionado por Ollman (2009).

Outro incidente nesse sentido aconteceu entre a Estônia e a Rússia em 2007. Após sofrer três semanas de ataques *DDoS*, a Estônia acusou a Rússia da autoria dos mesmos, gerando uma crise diplomática entre esses dois países, conforme é citado por Castro (2007). Outro ataque *DDoS* que adquiriu conotação política aconteceu em 2009, quando um grupo político denominado *DDoSIran*, que protestava contra as supostas fraudes nas eleições do Irã, utilizou o *Twitter* para convidar internautas a ajudarem em um ataque *DDoS* em sites do governo iraniano, como cita Übergeek (2009).

1.1 OBJETIVOS

Observando a necessidade de defesa contra ataques cibernéticos, surge o problema de como fazer essa defesa. Este trabalho tem como objetivos estudar e desenvolver uma defesa proativa contra ataques *DoS* do tipo *SYN Flood* e *Smurf* em um ambiente simulado, além de estudar a viabilidade do uso de máquinas virtuais para esse fim.

Essa defesa proativa se caracteriza, de forma geral, por um aplicativo que far o reconhecimento da assinatura dos ataques – característica que permite identificar os mesmos – do tipo *SYN Flood* e *Smurf* e tenta inviabilizá-los descartando os pacotes atacantes por um tempo pré-determinado, porém permitindo que a máquina atacante ainda consiga fazer outros tipos de requisições à máquina com a vítima, haja vista que em muitos casos as máquinas atacantes são vítimas de vírus e outras pragas digitais.

A escolha desse tema é justificada por propor uma defesa a alguns tipos de ataques cibernéticos, aspecto que vem ganhando cada vez mais destaque na mídia e que já é objeto de estudos a nível mundial por governos e organizações políticas e militares, como a Organização do Tratado do Atlântico Norte (OTAN), conforme *North Atlantic Treaty Organization* (2007). Além disso, as possibilidades na área de defesa para ataques do tipo

DoS é um campo amplo, devido à diversidade de métodos de ataque, sendo que muitos desses ainda não possuem solução definitiva.

Os ataques em questão forma escolhidos por serem ataques de negação de serviço bem definidos e conhecidos entre os ataques *DoS*.

2 GUERRA CIBERNÉTICA

A possibilidade de uma guerra cibernética já é algo real e há uma preocupação cada vez maior das nações e de organizações políticas e militares em desenvolver métodos de ataque e de defesa de natureza cibernética.

Além dos incidentes ocorridos na Estônia e no Irã, outros acontecimentos pelo mundo têm chamado a atenção para essa possibilidade. Pilkington (2008) menciona que um painel realizado pelo Congresso Americano alerta que a China estaria desenvolvendo forças capazes de promover um ataque cibernético que poderia atrasar ou prejudicar o envio de tropas militares pelo mundo. Segundo esta mesma reportagem, "*A habilidade da China para promover uma guerra cibernética é agora 'tão sofisticada que os Estados Unidos podem ser incapazes de contra-atacar ou mesmo detectar os esforços', alerta o relatório*" (tradução nossa).

O *Federal Bureau of Investigation* (*FBI*), a polícia federal americana, classificou os ataques cibernéticos como a terceira maior ameaça mundial, atrás apenas dos ataques nucleares e das armas de destruição em massa, como é citado por Presse (2009).

A OTAN também tem se mostrado preocupada com essa possibilidade, conforme pode ser visto em *North Atlantic Treaty Organization* (2007), que menciona uma preocupação com um terrorismo cibernético, citando cinco razões pelas quais essa modalidade é uma opção viável para os terroristas:

a) Os métodos cibernéticos são mais baratos que os métodos terroristas tradicionais;
b) O rastreamento de terroristas cibernéticos é mais difícil e confere a eles um maior anonimato;
c) O terrorismo cibernético possibilita uma base de alvos muito maior;
d) Os ataques podem ser conduzidos à distância; e
e) Os métodos cibernéticos podem afetar um maior número de pessoas que os métodos já conhecidos.

Assim, para discorrer melhor sobre esse assunto, segue a necessidade da definição do termo "guerra cibernética". Uma definição mais técnica e que independe de como esses ataques são realizados segue abaixo:

> "Guerra Cibernética é o subconjunto da guerra de informações que envolve ações realizadas pelo mundo cibernético. O mundo cibernético é qualquer realidade virtual delimitada por uma coleção de computadores e redes. Há vários mundos cibernéticos, porém o mais relevante para a guerra cibernética é a Internet e as redes relacionadas que compartilham mídia com a Internet. A definição militar mais próxima para o nosso termo, guerra cibernética, é uma combinação de ataque a redes de computadores e defesa de redes de computadores, e, possivelmente, operações de informações especiais." (PARKS, DUGGAN, 2001, tradução nossa).

Cabe ressaltar que os ataques cibernéticos envolvidos em uma guerra cibernética não necessariamente visam atingir alvos militares e políticos. Dutra (2007, p. 3), menciona que "bancos, usinas elétricas, empresas de telefonia e telecomunicações, sistemas de transporte e logística, serviços de emergência e segurança pública, entre outros são alvos em potencial". Serviços essenciais como esses, em caso de longos períodos de indisponibilidade, podem vir a prejudicar um país gravemente. Por causa disso, a defesa contra ataques cibernéticos também necessita de um planejamento, tornando-se mais crítica.

De acordo com Mirkovic e Reiher (2007), há duas estratégias básicas para os mecanismos de defesa contra ataques cibernéticos: a prevenção e a reação. Os mecanismos de prevenção utilizam-se de métodos como a contabilidade de recursos, que controla o acesso dos usuários aos recursos baseando-se na confiança do mesmo, e a multiplicação de recursos, caso em que os recursos, como a largura de banda, são tão abundantes que o ataque não funciona. A grande vantagem desse método é que clientes legítimos não são afetados pelo ataque.

A reação é uma estratégia de um mecanismo de defesa utilizada quando o ataque ocorreu e foi bem sucedido. A preocupação nesse caso é detectar os ataques e ter uma resposta rápida aos mesmos. Os mecanismos realizam a detecção pelos padrões (assinaturas) dos ataques, pelo modelo da anomalia ou através de mecanismos de terceiros não envolvidos diretamente no ataque, como roteadores, que utilizam-se de métodos como o *IP traceback*[1]. Serviços mais

[1] ***IP traceback*** – É todo e qualquer método capaz de determinar de forma confiável a origem de um pacote na Internet. Muito utilizado para se determinar a origem de um ataque de negação de serviço.

críticos normalmente possuem as duas estratégias combinadas para garantir o máximo de segurança possível.

Há diversos tipos de ataques cibernéticos, como a tentativa de ganhar acesso não autorizado a sistemas ou dados ou a disseminação de pragas digitais. Entretanto, o escopo desse trabalho relaciona-se apenas com os ataques do tipo *DoS*.

3 ATAQUES DE NEGAÇÃO DE SERVIÇO (*DOS*)

Os ataques *DoS* surgiram em meados da década de 90, em uma época em que não existiam serviços de banda larga disponíveis e o poder computacional era muito limitado se comparado aos parâmetros atuais. Originalmente, esses ataques aproveitavam-se de uma vulnerabilidade do protocolo *TCP*/*IP* que permitia o envio de pacotes via *broadcast*[1] de rede para uma máquina, o que gerava um número cada vez maior de pacotes do atacante até a vítima final. Dessa forma, eram necessários apenas alguns poucos pacotes para provocar a queda de várias máquinas pelo caminho, sem que muitas vezes a máquina a ser atacada ficasse indisponível. Porém essa vulnerabilidade foi facilmente corrigida desabilitando-se a recepção de mensagens *broadcast* em roteadores.

Há diversos tipos de ataques *DoS* que exploram vulnerabilidades distintas. Muitos dos ataques, em especial o *SYN Flood* e o *Smurf*, baseiam-se em protocolos normatizados pelo *Internet Engineering Task Force* (*IETF*). As definições desses protocolos são definidas em memorandos chamados Request for Comments (*RFC*). Os *RFCs* não descrevem apenas os protocolos de rede, mas também descrevem métodos, inovações, pesquisas e comportamentos relativos a redes.

Com o surgimento do serviço de banda larga, na época limitado a universidades, centros de pesquisa e outras corporações, e com a elevação do poder computacional, os *crackers*[2] passaram a utilizar outro método para os ataques *DoS*. Esse método baseava-se no envio da maior quantidade possível de pacotes às vítimas, sobrecarregando as mesmas. Em muitos casos, os *crackers* utilizavam-se de conexões presentes em universidades, onde normalmente existiam conexões com maior largura de banda.

Com o passar do tempo, as conexões de Internet tornaram-se mais homogêneas e os métodos de defesa evoluíram ao ponto de dificultar o sucesso de ataques *DoS*, conforme é citado por Mirkovic, Dietrich, Dittrich e Reiher (2004).

Para burlar esses problemas, *crackers* passaram a utilizar-se de várias máquinas para realizar um ataque distribuído, surgindo assim, no final da década de 90, os ataques *DDoS*.

[1] ***Broadcast*** **–** Forma de encaminhamento de pacotes que possibilita o envio da informação para todos os computadores da rede.
[2] O termo "*cracker*", aqui utilizado, refere-se a um *hacker* com intenções maliciosas.

A seguir, são detalhados alguns tipos de ataque *DoS*.

3.1 ATAQUES DISTRIBUÍDOS DE NEGAÇÃO DE SERVIÇO (*DDoS*)

Os ataques *DDoS* são ataques de negação de serviço executados por várias máquinas simultaneamente, fato que torna esses ataques de difícil detecção e defesa.

Na maioria dos casos, apesar da grande quantidade de máquinas realizando o ataque, este costuma ser comandado por um único computador. A forma tradicional de se conseguir esse controle se dá através de *worms*, que são programas capazes de se replicar automaticamente e que não dependem de outros programas para se propagarem. Há outras formas menos usuais que utilizam robôs (*bots*) para encontrar máquinas que possam ser controladas em redes, como as que se utilizam do protocolo *Internet Relay Chat* (*IRC*).

Entretanto, há a possibilidade de um ataque *DDoS* em que várias máquinas realizem um ataque combinado por cada usuário. O ataque coordenado pelo grupo *DDOSIran*, conforme citado anteriormente por Übergeek (2009), é um exemplo de ataque *DDoS* em que o ataque não é controlado por apenas uma máquina.

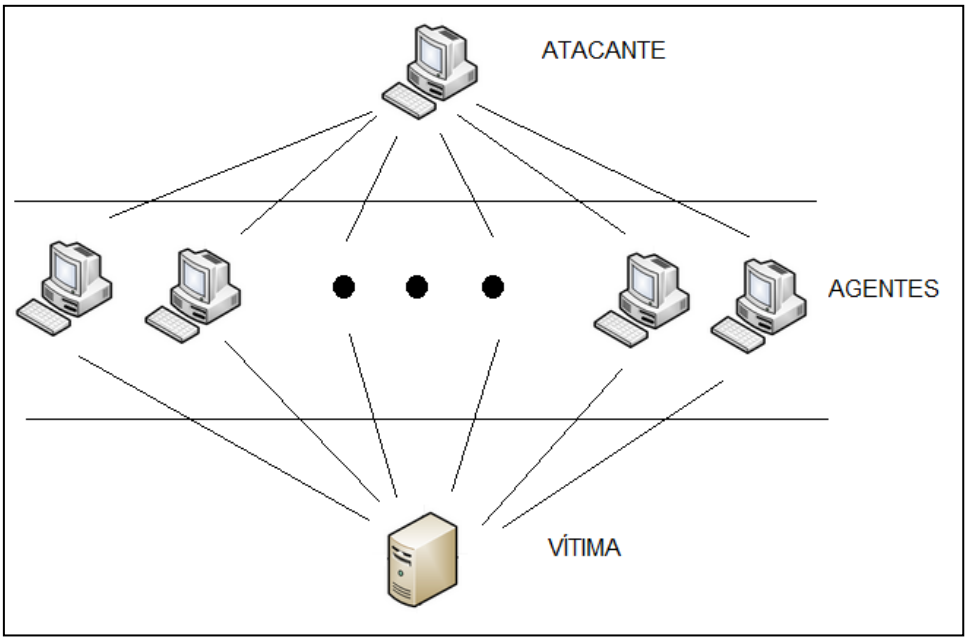

FIG. 3.1 – Hierarquia clássica de um ataque *DDoS*

Nos casos em que há um controlador por trás do ataque, há uma hierarquia entre o atacante e os agentes (também chamados de zumbis ou escravos), que são as máquinas

controladas que enviarão os pacotes que farão a inundação. A hierarquia clássica é apresentada na FIG. 3.1.

Para garantir um ataque mais eficaz através de um volume maior de tráfego na máquina da vítima e para dificultar o rastreamento do atacante, há outra arquitetura para o ataque, semelhante à anterior, com a presença de mestres, que entram na hierarquia entre o atacante e os agentes. Dessa vez, o atacante envia a ordem de ataque aos mestres, que por sua vez, repassa-a para os agentes. A hierarquia com a presença de mestres é representada na figura FIG. 3.2.

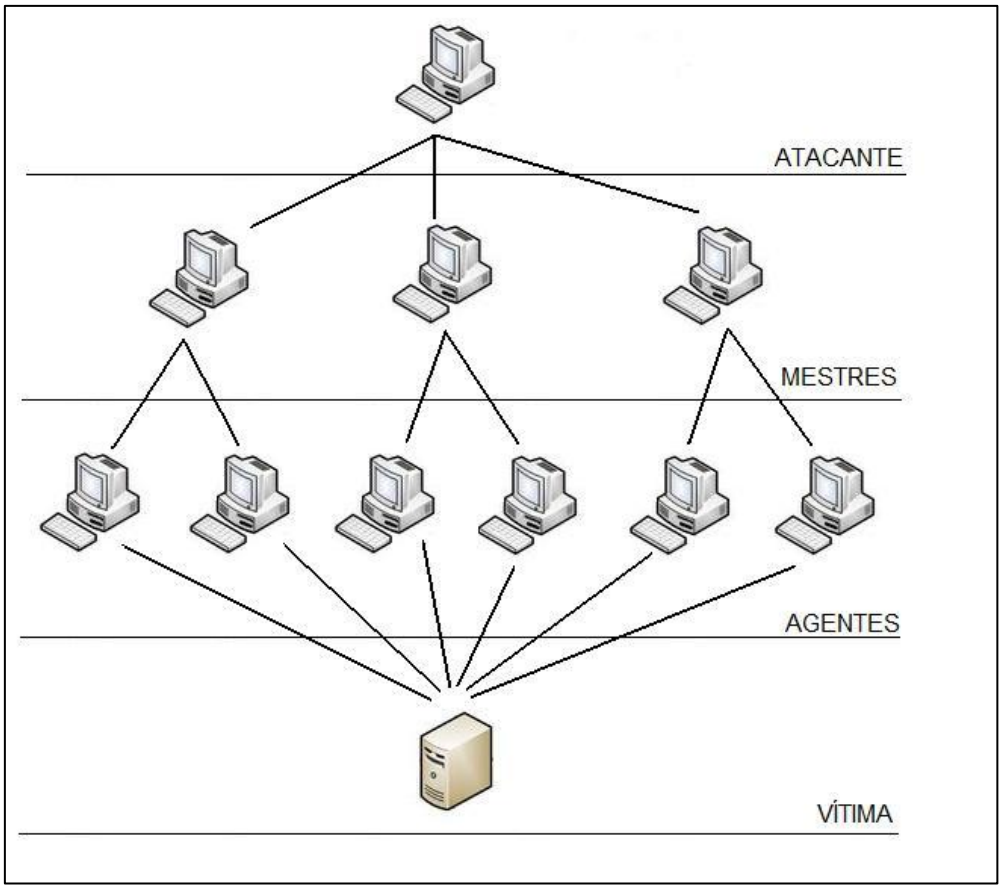

FIG. 3.2 – Hierarquia de um ataque *DDoS* com a presença de mestres

Os ataques *DDoS* são tão eficientes que foram desenvolvidas ferramentas para a criação desses ataques, que passaram a ser distribuídas em sites e fóruns *hackers*. Dentre as mais famosas, pode-se mencionar as ferramentas *Trinoo*, *Tribe Flood Network* (*TFN*) e seu sucessor *Tribe Flood Network 2000* (*TFN2K*), *Stacheldraht* (arame farpado, traduzindo do alemão), *Shaft*, *Mstream* e *Trinity*, conforme menciona Duarte (2009).

3.2 ATAQUES DISTRIBUÍDOS DE NEGAÇÃO DE SERVIÇO POR REFLEXÃO (*DRDoS*)

Os ataques distribuídos de negação de serviço por reflexão tratam-se de uma variante do *DDoS* tradicional, incluindo-se nessa estrutura os refletores. Nesse tipo de ataque as máquinas agentes enviam pacotes para os refletores com o endereço *IP* da vítima como o endereço *IP* de origem. Assim, os refletores responderão aos pacotes requisitantes para o *IP* da vítima, causando uma inundação na vítima. Em outro cenário, os refletores detectam o ataque e bloqueiam requisições cujo *IP* de origem coincida com o *IP* da vítima, o que impediria a vítima de acessar serviços localizados nos refletores, o que também caracteriza uma negação de serviço.

Gibson (2002) descreve um ataque *DRDoS* sofrido pelo site *www.grc.com* em janeiro de 2002. O ataque, inicialmente assemelhou-se a um ataque *DDoS* caracterizado por uma inundação de pacotes *TCP SYN-ACK*. Entretanto, a observância dos pacotes maliciosos mostrou que mais de dois mil roteadores do núcleo da infraestrutura da *Internet* eram os responsáveis pela geração dos pacotes. A porta de destino utilizada nos pacotes do ataque era 179, que correspondia à porta destinada ao *Border Gateway Protocol* (*BGP*). Basicamente, o *BGP* é responsável pela troca de informações entre roteadores que permitam o estabelecimento de trajetórias no tráfego de pacotes por toda a *Internet*, conforme define Rekther, Li e Hares (1995).

Então, como os roteadores em questão não poderiam se tratar de agentes, como ocorre em um ataque *DDoS*, Gibson concluiu que o atacante estava utilizando a técnica de *IP spoofing* para falsificar o endereço de origem dos pacotes maliciosos enviados, de forma que os roteadores respondiam as requisições para a vítima. Assim, os roteadores funcionavam como "refletores", configurando o ataque como DRDoS.

3.3 SYN FLOOD

O ataque *SYN Flood* é um ataque de negação de serviço que se baseia no modelo utilizado pelo protocolo *Transmission Control Protocol* (*TCP*) para estabelecer uma conexão e explora o tamanho máximo da tabela de conexões na máquina – o *backlog* – da vítima.

O protocolo *TCP* é um protocolo da camada de transporte que fornece conexão confiável entre dois pontos, definido no *RFC* 793. A FIG. 3.3 descreve o cabeçalho do protocolo *TCP*.

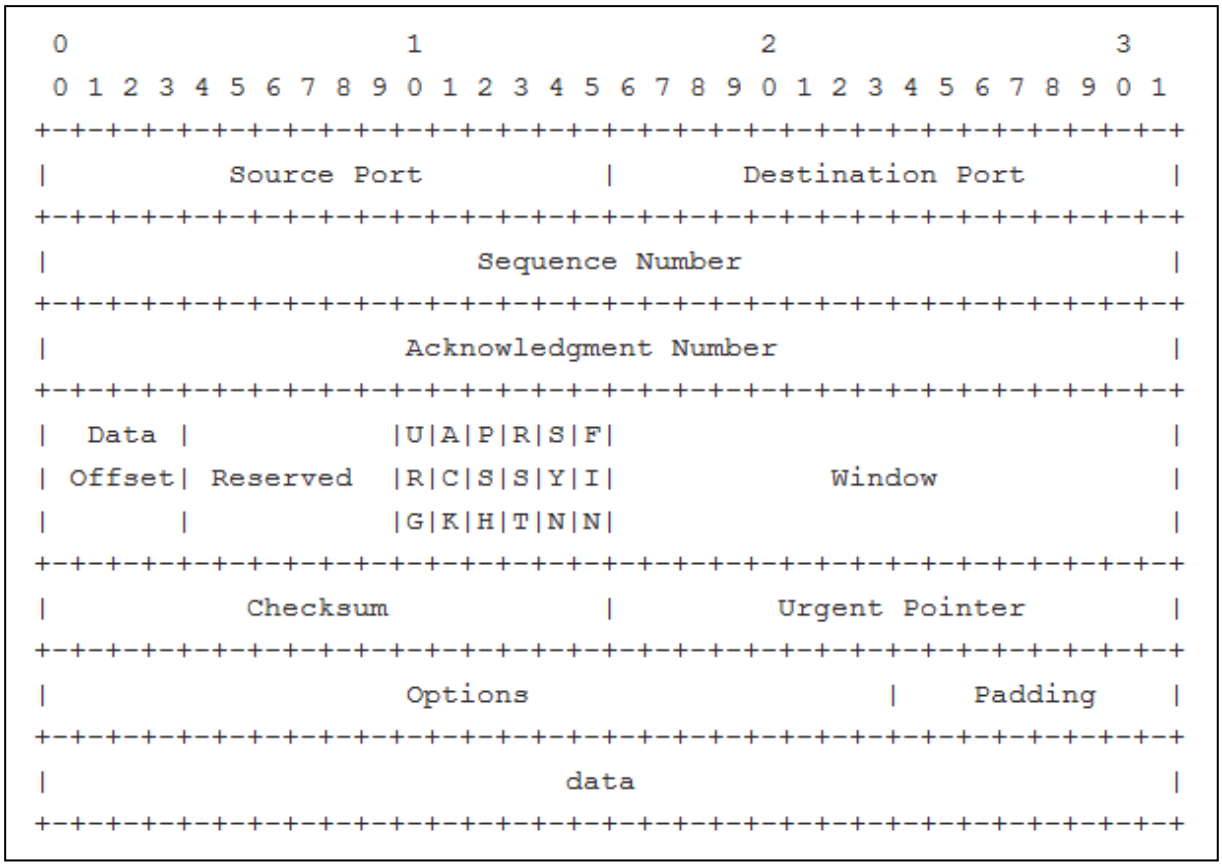

FIG. 3.3 - Cabeçalho TCP (INFORMATION SCIENCES INSTITUTE, 1981)

Para que as máquinas iniciem a troca de dados, antes é necessário que as duas máquinas estabeleçam uma conexão e isso é feito através do *handshake* de três vias, representado na FIG. 3.4.

Inicialmente, o cliente envia um pacote *TCP* com o *bit SYN* com o valor igual a um (quinto *bit* do campo *bits* de controle do cabeçalho *TCP*), o que identifica o pedido de conexão por parte do cliente. Esse pacote contém também um número de sequência de 32 *bits*, que será utilizado pelas máquinas para identificar a requisição *SYN*.

O servidor, ao receber o pedido de conexão, envia um pacote *TCP* com os *bits SYN* e *ACK* iguais a um (quinto e segundo *bits* do campo *bits* de controle do cabeçalho *TCP*, respectivamente). Ainda nesse pacote, o servidor gerará outro número de sequência e ajustará o valor do número de confirmação contendo o valor do número de sequência do pacote enviado pelo cliente acrescido de um, para que o cliente possa identificar a resposta. Por fim, o servidor alocará um espaço em seu *backlog* com o *IP* do cliente e o número de sequência enviado, espaço esse chamado de *Transmission Control Block* (*TCB*).

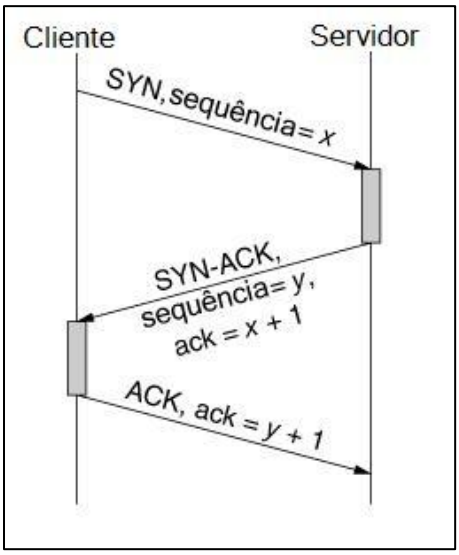

FIG. 3.4 – Representação de um *handshake* de três vias

O último passo para que o *handshake* seja concluído é feito pelo cliente. Este recebe o pacote *SYN-ACK* do servidor e envia um pacote *TCP* para o servidor contendo o *bit ACK* igual a um e com o número de confirmação igual ao sucessor do número de sequência recém-recebido. Com o recebimento deste último pacote pelo servidor, a conexão é estabelecida e o servidor liberará o espaço reservado em seu *backlog*.

O ataque *SYN Flood* explora justamente essa espera feita pelo servidor pela mensagem *ACK* do cliente. O atacante envia um grande fluxo de requisições *SYN* contendo um *IP* de origem falso – técnica conhecida como *IP spoofing* – para a vítima, ao ponto de sobrecarregar o seu *backlog*, o que impede clientes legítimos de estabelecerem uma conexão, causando, assim, uma negação de serviço.

EDDY (2007) cita no *RFC* 4987 as defesas mais comuns contra ataques *SYN Flood*, as quais são descritas abaixo:

a) **Filtragem**: Técnicas de filtragem têm como objetivo impedir o sucesso na utilização de *IP spoofing* pelos atacantes. Essas técnicas podem ser implementadas em roteadores, *firewalls*[1] ou sistemas operacionais;

b) **Aumento do *backlog***: Com o aumento do backlog, os ataques terão maior dificuldade de sucesso. Entretanto, o mesmo *RFC* cita que essa tática traz uma

[1] ***Firewall*** – É um dispositivo capaz de aplicar políticas de controle de acesso entre pontos distintos em uma rede ou entre redes diferentes, aumentando a segurança. Firewalls são capazes de filtrar pacotes ou garantir ou negar acesso de programas à rede.

série de aspectos negativos. Algoritmos de busca e estruturas de dados, por exemplo, podem tornar-se ineficientes em *backlogs* muito grandes;

c) **Redução do tempo de permanência no *backlog***: Ao receber uma requisição *SYN*, o servidor aguardará por menos tempo a resposta *ACK* do cliente. Assim, o *backlog* será "esvaziado" a uma velocidade maior, dificultando a ação do atacante. Por outro lado, será mais difícil para clientes distantes efetuarem o *handshake* com o servidor;

d) **Reciclar os TCBs mais antigos**: Para evitar o "estouro" do *backlog*, uma implementação pode optar por substituir os TCBs mais antigos caso o *backlog* esteja cheio;

e) **SYN *Cache***: O *SYN Cache* tem por objetivo garantir que os *TCBs* ocupem uma quantidade menor de memória e que sejam descartados de forma mais eficiente. Valores *hash* serão calculados com bits secretos inseridos no número de sequência, com o *IP* do cliente e com a porta utilizada na conexão. Cada valor de *hash* possuirá uma capacidade máxima, que caso seja atingida, descartará as entradas mais antigas, garantindo uma alta efetividade contra ataques *SYN Flood*;

f) ***SYN Cookies***: Nesse tipo de abordagem, a idéia é fazer com que o servidor aloque recursos apenas ao final do *handshake*. O servidor utiliza uma senha juntamente com o *IP* de origem e a porta especificada para gerar um valor *hash* que será utilizado como número de sequência. Assim, ao receber o pacote *ACK*, o servidor comparará o valor do número de confirmação com o valor esperado para aquele *IP* e para aquela porta de destino. Sistemas Linux utilizam *SYN Cookies*;

g) **Abordagens híbridas**: Abordagem híbrida é a utilização de mais de uma das técnicas mencionadas acima;

h) ***Firewalls* e *proxies***: Para proteger os servidores de ataques *SYN Flood* uma alternativa é a colocação de um *firewall* anteposto ao servidor, de forma que este funcione como um *proxy*[1] e repasse as requisições válidas para o servidor. Apesar de essa estratégia não evitar o ataque, ela tem o mérito de isolar o servidor propriamente dito dos ataques.

[1] ***Proxy*** – É um dispositivo que se faz passar por outra máquina para atender às requisições da mesma. O *proxy* pode também armazenar informações temporárias (*cache*) e repassá-las ao cliente sem a necessidade de estabelecer uma conexão propriamente dita.

Vale ressaltar que regras e políticas de *firewall* são pouco efetivas contra ataques *SYN Flood*. Tentar bloquear *IPs* que estão realizando muitas requisições *SYN* sem completar o *handshake* funciona somente em casos em que o atacante está realizando *IP spoofing* com apenas um endereço de *IP*, bastando ao atacante gerar pacotes *SYN* com *IPs* aleatórios. Ainda assim, para o caso em que o atacante estiver gerando pacotes com um único *IP*, isso pode ser utilizado como uma variante do ataque, sendo a vítima o computador com o *IP* utilizado pelo atacante, que tem o acesso negado ao servidor protegido pelo *firewall*.

Outra tática comum em *firewalls* é limitar a taxa de chegada de pacotes *SYN* ao servidor. Esse tipo de política acaba por facilitar o ataque, sendo necessário somente que o atacante envie requisições *SYN* a uma taxa superior ao limite estabelecido na regra para causar a negação de serviço.

Para a detecção desse tipo de ataque, a observância de anomalias na proporção de requisições *SYN* com o *handshake* incompleto com relação ao número total de pacotes *SYN* recebidos costuma ser suficiente. Entretanto, há a possibilidade de identificação de falsos positivos, mas é algo pouco comum em ataques desse tipo.

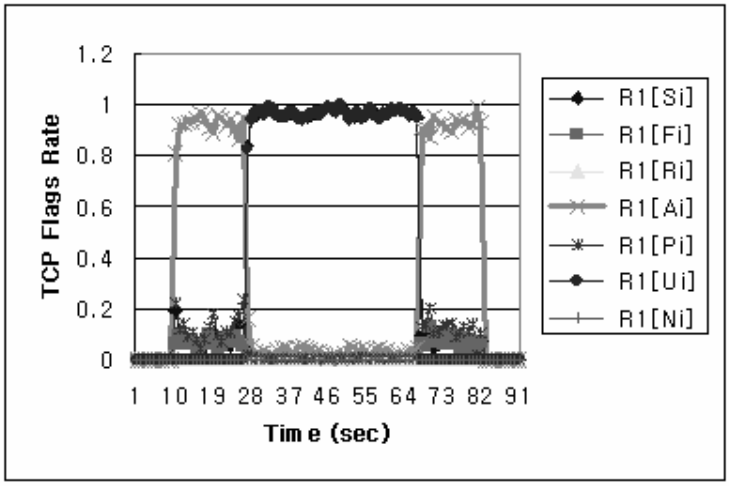

FIG. 3.5 – Taxa de flags *TCP* recebidas por uma máquina sob um ataque *SYN Flood* (LEE, NOH, CHOI, JUNG, 2004)

Lee, Noh, Choi e Jung (2004) caracterizaram o ataque *SYN Flood* através de uma análise das taxas do tráfego enviado e recebido em um computador com ou sem ataque. Os resultados mostraram que, antes dos ataques, a taxa de pacotes com a *flag SYN* com relação ao total de pacotes que chegavam à máquina, *R1[Si]*, não ultrapassava 15% do total, enquanto a taxa de

pacotes com a *flag ACK*, também recebidos pela máquina, *R1[Ai]*, era sempre maior do que 80% do total. Após o início do ataque, a taxa de pacotes com a *flag SYN* recebidos manteve-se próxima de 100%. A FIG. 3.5 ilustra a mudança dessas taxas, observadas após o ataque, feito entre os instantes 30 segundos e 70 segundos.

3.4 SMURF

O ataque *Smurf* tem como objetivo gerar um grande volume de pacotes *ICMP* do tipo *Echo Reply* para a vítima, causando um congestionamento na conexão.

O protocolo *Internet Control Message Protocol* (*ICMP*) é um protocolo da camada de transporte com o objetivo de reportar problemas ou prover controle no ambiente de comunicação, definido no *RFC* 792. O cabeçalho do protocolo *TCP* varia de acordo com o tipo da mensagem, mas, todas iniciam com os mesmos três campos a seguir: o campo tipo, que identifica a finalidade da mensagem e ocupa o primeiro octeto; o campo código, que fornece informações mais específicas sobre a mensagem e ocupa o octeto seguinte; e o campo *checksum*, que realiza a soma em complemento a 1 de todas as palavras de 16 *bits* do cabeçalho *ICMP* e armazena no terceiro e no quarto octetos, de forma a fornecer uma verificação dos dados transmitidos.

Pode-se utilizar mensagens *ICMP* para verificar se um destino é alcançável e está respondendo. Esse tipo de verificação, feita com o comando *ping* nos sistemas operacionais *Windows*, *Mac OS* e *Linux*, utiliza-se de dois tipos de mensagens *ICMP*. Uma máquina que recebe uma mensagem *ICMP* do tipo *Echo Request* (campo tipo no cabeçalho *ICMP* igual a oito) responderá com uma mensagem do tipo *Echo Reply* para a máquina de origem.

Normalmente, a origem envia apenas algumas poucas mensagens de tamanho reduzido e a uma taxa bem abaixo do que é realizado em ataques de negação de serviço, para não sobrecarregar a rede ou a máquina de destino. Entretanto, atacantes enviam um grande fluxo de pacotes *ICMP Echo Request* via *broadcast* para a rede utilizando *IP spoofing* para falsificar o *IP* de origem com o *IP* da vítima. Assim, o roteador é utilizado como amplificador do ataque pelo fato de este repassar as requisições para todas as máquinas na rede, gerando uma grande quantidade de pacotes *ICMP Echo Reply* para a vítima, sobrecarregando, desta forma, a sua conexão com a rede. Uma rede classe C, por exemplo, é capaz de amplificar em

até duzentos e cinquenta vezes o tráfego gerado pelo atacante, dependendo do número de máquinas na rede.

Com a grande efetividade desse ataque, roteadores e sistemas operacionais passaram a bloquear requisições *ICMP Echo Request* via *broadcast*. Os atacantes então resolveram utilizar outro meio para realizar o ataque. Ao invés de explorar vulnerabilidades, eles passaram simplesmente a enviar um fluxo de pacotes maior ou na ordem da velocidade da conexão da vítima. Por exemplo, uma vítima com conexão discada (56 kbps) é facilmente afetada por esse tipo de ataque se o atacante estiver utilizando uma conexão de banda larga comum (256kps).

4 VIRTUALIZAÇÃO

Virtualização é a tecnologia que permite a existência de máquinas virtuais, que são cópias simuladas de um *hardware* real criadas por software capazes de executar sistemas operacionais, conforme define TANENBAUM (2010).

Assim, a virtualização permite que a máquina hospedeira execute várias máquinas virtuais, em que cada uma possui seu próprio sistema operacional. A grande utilidade dessa tecnologia é permitir a economia de recursos físicos, já que todas as máquinas virtuais compartilham um mesmo *hardware* real. Outras vantagens dessa tecnologia são a facilidade no gerenciamento de diversas máquinas e a possibilidade de se criar um ambiente diversificado de forma simplificada.

As máquinas virtuais são controladas por programas chamados de hipervisores (ou monitores de máquina virtual). Há dois tipos de hipervisores que definem como será feita a abordagem de virtualização:

a) **Hipervisor tipo 1**: Neste tipo de abordagem, o sistema operacional assume o papel de hipervisor e cria um processo distinto para cada máquina virtual a ser executada. Os *softwares VMware ESX* e *Microsoft Hyper-V* realizam esse tipo de abordagem, conforme Evans (2010).

b) **Hipervisor tipo 2**: Neste caso, o hipervisor é um programa de usuário sendo executado pelo sistema operacional que funciona como uma camada entre o sistema operacional e as máquinas virtuais, interpretando os comandos executados pelos sistemas operacionais das máquinas virtuais. Evans (2010) cita como exemplo dessa abordagem os *softwares VMWare GSX Server*, *Winodws Virtual PC* e *Oracle VirtualBox*.

Para se conseguir um melhor desempenho e evitar problemas como a chamada de instruções sensíveis[1], pode-se adotar a estratégia conhecida como paravirtualização, que consiste em modificar o sistema operacional da máquina virtual para que este execute uma interface definida pelo hipervisor ao invés do conjunto normal de instruções a que foi programado inicialmente.

[1] **Instrução sensível** – Instruções que somente podem ser executadas pelo sistema operacional hospedeiro, como instruções de configuração na unidade de gerenciamento de memória ou instruções de entrada e saída.

5 EXPERIMENTO

A partir desse ponto, serão descritos a estrutura dos ataques, a estrutura da defesa para os ataques e o ambiente utilizado para simular os ataques e verificar a capacidade e as medidas da defesa.

5.1 AMBIENTE DO EXPERIMENTO

No intuito de simular uma rede interna que possibilite avaliar a capacidade de defesa e o uso de máquinas virtuais para esse fim, criou-se um ambiente de testes para o experimento contendo 6 máquinas em uma mesma rede, sendo duas reais (Real 1 e Real 2) e quatro virtuais. Segue abaixo uma descrição sucinta das máquinas reais utilizadas no ambiente:

a) **Máquina Real 1**: Sistema operacional *Windows Vista Home Premium SP2*, *IP* 192.168.91.233, contendo *Oracle VM VirtualBox* 3.2.6 e possuindo contendo um processador *Intel® CoreTM 2 Duo T5550* 1,83GHz e disponibilizada com 512MB de memória RAM;

b) **Máquina Real 2**: Sistema operacional *Windows XP SP2*, *IP* 192.168.91.253, contendo *VMware GSX Server* 2.0.0, contendo um processador *Intel® CoreTM 2 Duo E4600* de 2,40GHz e disponibilizada com 512MB de memória RAM;

Quanto às máquinas virtuais, segue abaixo uma lista das máquinas utilizadas, cada uma com sua máquina hospedeira especificada:

a) **Máquina vítima**: Máquina virtual hospedada na máquina Real 1 com sistema operacional *Ubuntu* 9.04, *IP* 192.168.91.234, contendo *iptables* v1.4.1.1, *Apache HTTP Server* 2.2.14 e *MySQL* 5.1.37. Esta máquina teve a utilização de *SYN cookies* desativada para garantir que o ataque *SYN Flood* fosse realizado com sucesso;

b) **Máquina atacante**: Máquina virtual hospedada na máquina Real 2 com sistema operacional *Ubuntu* 9.10, *IP* 192.168.91.235, contendo *SendIP* 2.5;

c) **Máquina neutra 1**: Máquina virtual hospedada na máquina Real 2 com sistema operacional *Ubuntu* 9.10, *IP* 192.168.91.236;

d) **Máquina neutra 2**: Máquina virtual hospedada na máquina Real 2 com sistema operacional *Windows XP SP2*, *IP* 192.168.91.237;

Um esquema representativo do ambiente do experimento é representado conforme a FIG. 5.1.

No trabalho em questão, os softwares de máquina virtual, ambos hipervisores do tipo 2, foram utilizados por serem gratuitos e por já oferecerem uma estrutura suficiente para a realização do trabalho.

Como apenas máquinas atacante e vítima serão exigidas em termos de processamento e memória, sendo as demais apenas figurativas no ambiente, a especificação destes itens de hardware nas máquinas neutras é desnecessária.

Outro ponto a se ressaltar no ambiente é a largura de banda. Todas as máquinas estavam ligadas a uma rede *Gigabit Ethernet* (1000 Mbps) por meio de cabo de par trançado.

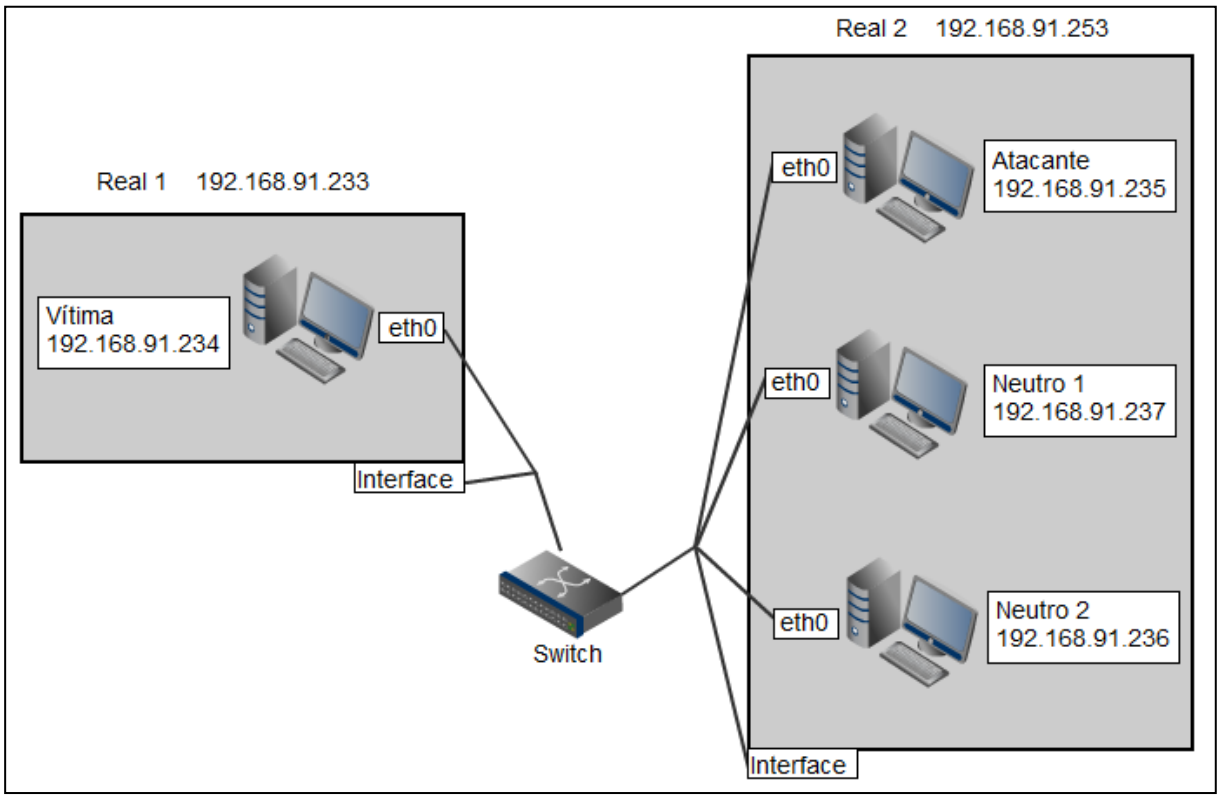

FIG. 5.1 - Esquema representativo do ambiente do experimento

As máquinas neutras presentes no ambiente têm como objetivo realizar requisições válidas durante o ataque para a vítima durante o ataque. A presença das mesmas permitiu

verificar também a capacidade da máquina atacante com as mesmas ligadas ou não durante o ataque. Já as máquinas reais possuíam como única função hospedar as máquinas virtuais.

É importante saber que há outras máquinas presentes na rede, mas estas pouco interferiram no desempenho dos testes.

5.2 ESTRUTURA DE ATAQUE

Para a realização dos ataques *SYN Flood* e *Smurf*, utilizou-se o programa S*endIP*, disponível para Linux. Huggins (2003) caracteriza o *SendIP* como: "*SendIP é uma ferramenta de linha de comando capaz de enviar pacotes IP arbitrários*" (tradução nossa). Assim, o usuário especifica por meio de linha de comando o cabeçalho e os dados do pacote que deseja que seja gerado pelo *SendIP*.

O uso do *SendIP* como ferramenta de ataque justifica-se pela sua capacidade de enviar pacotes especificados pelo usuário, inclusive o uso de *IP spoofing*. Hoch (2004) e Holland (2001) demonstram como o *SendIP* pode ser utilizado para os ataques em questão.

O *SendIP* conseguiu gerar pacotes em ambos os ataques a uma taxa de aproximadamente 530 pacotes/s.

A instalação do *SendIP* foi feita na máquina do atacante utilizando o comando a) do APÊNDICE 1.

5.2.1 Ataque *SYN Flood*

Serão realizados dois cenários de ataque *SYN Flood*. Em um primeiro cenário, chamado de **Cenário 1**, o ataque fará uso da técnica de *IP Spoofing* utilizando-se apenas de um mesmo *IP* durante todo o ataque. No outro cenário, chamado de **Cenário 2**, o ataque usará *IPs* aleatórios durante o ataque.

No **Cenário 1**, será utilizado o comando b) do APÊNDICE 1 para realizar o ataque. Esse comando fará com que o *SendIP* seja executado indefinidamente até que o usuário interrompa manualmente o processo. Assim, serão enviados pacotes *TCP SYN* destinados à máquina da vítima (*IP* 192.168.91.234) partindo de uma porta aleatória e de um *IP* de origem falso (*IP* 192.168.0.100). A porta de destino será a porta do servidor *web* (porta 80). A velocidade com que os pacotes são gerados e enviados dependerá do processador da máquina do atacante e da largura de banda disponível.

No **Cenário 2**, será utilizado o comando c) do APÊNDICE 1para realizar o ataque. Esse comando funcionará de forma semelhante ao comando utilizado no Cenário 1, com a diferença de que o *IP* de origem será escolhido aleatoriamente pelo *SendIP*.

5.2.2 Ataque *Smurf*

O ataque *Smurf* será realizado através do comando d) do APÊNDICE 1 no *SendIP*. Dessa forma, serão enviados pacotes *ICMP Echo Reply* destinados à máquina da vítima (*IP* 192.168.91.234) partindo de uma origem aleatória. Apesar de não se estar utilizando a configuração clássica do ataque *Smurf* em que o atacante envia pacotes via *broadcast* utilizando *IP spoofing* com o *IP* da vítima, não será notada diferença do ponto de vista da máquina da vítima, que continuará a receber pacotes *ICMP Echo Reply*.

5.3 ESTRUTURA DA DEFESA

Inicialmente, antes de se elaborar qualquer estratégia de defesa contra ataques de negação de serviço, é necessário que se detecte esses ataques. Estes apresentam assinaturas características que permitem identificá-los, que são determinadas pelos tipos de pacotes e pelo fluxo com que os mesmos chegam à vítima.

5.3.1 Assinaturas dos ataques

A característica principal a ser observada na detecção de um ataque *SYN Flood* reside na existência de um valor anormal na proporção de requisições *SYN* com o *handshake* incompleto com relação ao número total de pacotes *SYN* recebidos. Em situações normais, raros são os casos em que são necessárias retransmissões de pacotes *SYN-ACK* por parte do servidor. Assim, pode-se assumir que um valor absoluto pouco elevado de conexões em estado *SYN RECEIVED* (aguardando o *ACK* para o estabelecimento da conexão) já pode indicar um ataque *SYN Flood*. Burdach (2003) menciona esta métrica para a detecção de ataques *SYN Flood*.

Entretanto, é possível a detecção de falsos positivos por este tipo de medição, mas são casos raros e que dependem de condições especiais para ocorrerem, como picos de acesso descomunais por parte de usuários legítimos a um servidor. No ambiente do experimento, que é controlado, esse tipo de situação não ocorrerá.

Para um ataque *Smurf*, a chegada de uma quantidade elevada de pacotes *ICMP Echo Reply* não esperados em uma máquina é condição suficiente para se configurar um ataque, já que pacotes desse tipo devem estar precedidos de pelo menos uma requisição *ICMP Echo Request*, sendo que apenas requisições via *broadcast* podem receber mais respostas do que a quantidade enviada.

5.3.2 Estratégias de defesa

Com as assinaturas dos ataques caracterizadas, faz-se necessária a criação de estratégias que permitirão a defesa contra os ataques. Neste experimento, a estratégia genérica para a defesa será feita por meio de filtragem de pacotes, ou seja, os pacotes devem ser descartados antes de prejudicarem a máquina vítima. No *Linux*, o *iptables* pode ser configurado para descartar pacotes de acordo com regras especificadas e alteráveis. O *iptables* é o programa de linha de comando usado para configurar o conjunto de regras de filtragem de pacotes utilizados no *Linux* 2.4 e 2.6, conforme define Ayuso (2010), fornecendo uma interface para administração do *firewall* do *kernell* do *Linux*.

Entretanto, o *iptables* não fornece inteligência e não consegue detectar padrões de ataques. Sendo assim, não é possível estabelecer uma defesa específica para os ataques *SYN Flood* e *Smurf* apenas com a especificação de regras no mesmo. Conforme mencionado na seção 3.3, regras e políticas de *firewall* são pouco efetivas contra ataques *SYN Flood*. Já com relação a ataques *Smurf*, não há como o *iptables* diferenciar pacotes *ICMP Echo Reply* legítimos de pacotes utilizados no ataque.

Então se faz necessário o uso de alguma ferramenta que permita a detecção dos ataques e que possibilite um gerenciamento do *iptables*. Assim criou-se uma aplicação em *Java* que cumprirá essas duas finalidades. Os pacotes serão capturados com as classes e métodos da biblioteca *jpcap*, criada por Kujii (2007).

Ao serem capturados, os pacotes serão analisados de acordo com as métricas supracitadas. Essas métricas possuem parâmetros que terão seus valores atribuídos de acordo com os resultados dos experimentos e de acordo com outros parâmetros estáticos, como tamanho do *backlog* e largura de banda. Os parâmetros utilizados, assim como os valores obtidos nos experimentos serão citados e discutidos posteriormente.

Caso a análise dos pacotes identifique um ataque, a aplicação irá acrescentar, modificar ou remover regras no *iptables*. Uma representação do diagrama de classes simplificada da aplicação pode ser observada na FIG. 5.2.

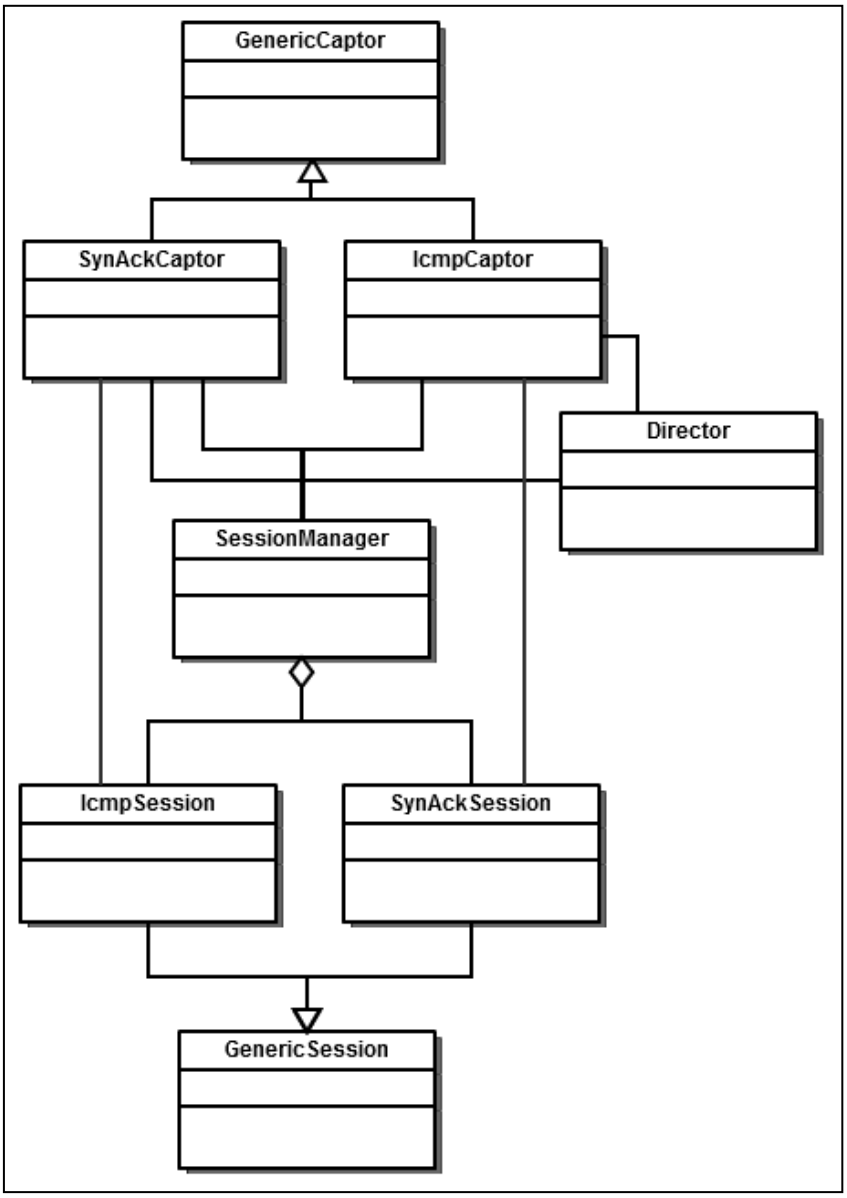

FIG. 5.2 – Diagrama de classes da aplicação

A classe **GenericCaptor** define um capturador de pacotes genérico. As classes **IcmpCaptor** e **SynAckCaptor** são subclasses específicas da classe **GenericCaptor** que possuem filtros específicos para a captura de pacotes e que analisarão os mesmos de acordo com as métricas específicas para cada ataque monitorado.

Na classe **Director**, haverá um conjunto de regras do *iptables* e um método que permite que essas regras sejam incluídas, modificadas ou removidas. A sua execução será feita pelas classes **IcmpCaptor** e **SynAckCaptor** quando houver necessidade.

A classe **SessionManager** agregra as classes **IcmpSession** e **SynAckSession**, sendo responsável pela manutenção dos conjuntos de instâncias dessas classes e por fornecer informações como total de instâncias desses conjuntos para as classes **IcmpCaptor** e **SynAckCaptor**. Vale ressaltar que a classe **SessionManager** funciona como um *Singleton* e só será instanciada uma única vez, para garantir que haja apenas uma coleção de **IcmpSession** e uma de **SynAckSession**.

As sessões criadas para cada pacote, que se auto-monitoram, são definidas pela classe genérica **GenericSession**, onde serão definidos os tempos de duração da sessão de cada pacote e os métodos pelos quais a mesma irá verificar se é igual a outro objeto instanciado do mesmo tipo.

Por fim, é necessário definir como os pacotes serão analisados individual e coletivamente e como os mesmos atuarão perante o *iptables*. Isso diferirá a implementação das classes **IcmpSession** e **SynAckSession** e das classes **IcmpCaptor** e **SynAckCaptor**.

Com relação ao ataque *Smurf*, o *iptables* terá uma regra que bloqueará a chegada de quaisquer pacotes *ICMP Echo Reply*. Essa regra será incluída na cadeia **icmpchain**. Uma cadeia é um local que armazena um conjunto de regras específicas definidas pelo usuário. Os comandos para criar a cadeia **icmpchain** no *iptables* e inserí-la na cadeia **INPUT** da tabela **filter** estão nos itens a) e b) do APÊNDICE 2.

Para adicionar a regra à cadeia **icmpchain** que descartará todos os pacotes do tipo *ICMP Echo Reply* que chegarem à máquina vítima (*IP* 192.168.91.234), será executado o comando c) do APÊNDICE 2.

A aplicação, ao rastrear o envio de um pacote *ICMP Echo Request*, deverá criar um objeto do tipo **IcmpSession** que armazenará o *IP* de origem, o *IP* de destino e o número de sequência do pacote *ICMP*. A aplicação inserirá uma regra antes da regra principal que descartará todos os pacotes *ICMP Echo Reply* no *iptables* aceitando pacotes desse tipo com origem no *IP* de destino do pacote enviado. Essa regra será definida pela sintaxe presente em d) do APÊNDICE 2, colocando-se o *IP* correto da máquina de destino.

O **IcmpCaptor**, ao receber o pacote *ICMP Echo Reply* da origem, removerá a regra anterior com o comando e) da APÊNDICE 2.

A cada segundo, a instância de **IcmpSession** criada verificará a chegada do pacote. Em caso positivo, a sessão se encerrará automaticamente. Caso contrário, ela fará essa verificação um número determinado de vezes, chamado de **numberOfIcmpLoops**, que é um dos parâmetros a ser definido.

Para o ataque *SYN Flood*, ao contrário do que foi feito com o ataque *Smurf*, não se pode proibir a chegada de requisições *SYN*, pois não há como diferenciar pacotes legítimos de pacotes maliciosos. Além disso, apesar de o ataque sempre ser detectável, não há defesa com os meios utilizados para o **Cenário 2**, descrito na seção 5.2.1, conforme foi explicado na seção 3.3.

Para a detecção do ataque no **Cenário 2**, utilizou-se um parâmetro, chamado de **tcpMaxGlobalOpen**. Esse parâmetro determinará o número máximo de conexões no estado *SYN RECEIVED* existentes simultaneamente, em um intervalo de tempo determinado pelo parâmetro **tcpTimestamp**.

No **Cenário 1**, haverá um parâmetro, chamado de **tcpMaxUniqueOpen**, que determinará o número máximo de conexões no estado *SYN RECEIVED* existentes simultaneamente para cada IP, em um intervalo de tempo novamente determinado pelo parâmetro **tcpTimestamp**.

Caso um mesmo *IP* consiga gerar um número máximo de conexões com *handshake* incompleto, a aplicação inserirá a regra f) do APÊNDICE 2 na cadeia **synchain** da tabela **filter** no *iptables* para rejeitar os pacotes vindos do *IP* que está causando o ataque.

Essa regra ficará ativa por um tempo especificado pelo parâmetro **synTimeout**.

Com a chegada de um pacote *SYN*, será instanciado um objeto do tipo **SynAckSession**, contendo o *IP* de origem e o valor do sucessor do número de sequência recebido. Ao ser enviado um pacote *SYN-ACK* pelo servidor, este também gravará o valor do sucessor do número de sequência deste pacote. Assim, será possível verificar os próximos pacotes *ACK* que chegarão e determinar quando a conexão for completamente estabelecida.

6 ANÁLISE DOS RESULTADOS

Antes da determinação dos resultados, deve-se atentar para os valores do tamanho do *backlog*, do número de vezes que o sistema operacional tenta reenviar pacotes *SYN-ACK* e do tempo de espera por chegada de pacotes.

O *Ubuntu* define um valor padrão de 1024 entradas para o tamanho do *backlog*. Esse valor pode ser verificado na variável */proc/sys/net/ipv4/tcp_syn_backlog* e não será alterado.

O número de vezes que o sistema operacional tenta reenviar pacotes *SYN-ACK* assume um valor padrão de 5 e também não será alterado. Este valor pode ser verificado em */proc/sys/net/ipv4/tcp_synack_retries*.

Tempo de espera de pacotes: Caso o sistema operacional não receba uma confirmação durante uma conexão *TCP*, este reenviará os pacotes para o destino ou removerá a conexão em aberto do *backlog*. O tempo de espera por pacotes varia com o tempo e é determinado dinamicamente utilizando-se dos tempos de transmissão medidos anteriormente, conhecidos como *Round Trip Time and Timeout* (*RTT*), definido no *RFC* 793. O *Smoothed Round Trip Time and Timeout* (*SRTT*) representa o valor utilizado pelo sistema operacional para estimar o tempo de transmissão de um pacote e é calculado através de (6.1)**Erro! Fonte de referência não encontrada.**:

$$SRTT_{atual} = \alpha * SRTT_{anterior} + (1 - \alpha) * RTT \quad (6.1)$$

O *RTT* é a medida do tempo de transmissão do último pacote feita com sucesso e α representa o fator de amortecimento do *SRTT*. Com o valor do *SRTT* determinado, o sistema operacional aguardará um tempo de espera *Retransmission Timeout* (*RTO*), determinado por (6.2):

$$RTO = min\,(valor_{maximo}\,, max(valor_{minimo}\,, \beta * SRTT)) \quad (6.2)$$

Em que $valor_{maximo}$ representa o tempo máximo de espera, $valor_{minimo}$ representa o tempo mínimo de espera e β representa um fator de variância, com o intuito de fornecer uma margem

de segurança. O valor de **tcpTimestamp** será calculado de forma semelhante ao *RTO*, utilizando-se os seguintes valores: α = 95%, β = 2, RTT_{incial} = 40s, $valor_{maximo}$ = 60s e $valor_{mínimo}$ = 1s, que são valores próximos dos valores citados por COMER (2007) e por Information Sciences Institute (1981).

Foi determinado também que o valor de **numberOfIcmpLoops** seria calculado de forma semelhante ao *RTO*, com o valor de RTT_{incial} = 5s. Os pacotes *ICMP Echo Request* geralmente são pacotes pequenos (o padrão verificado na máquina vítima é de 56 bytes) e por isso o tempo de resposta costuma ser menor do que 2 segundos. Entretanto, esses pacotes são processados com baixa prioridade pelo sistema operacional, o que pode levar a altos tempos de resposta.

O valor de **tcpMaxGlobalOpen**, que representa o número mínimo de sessões do tipo **SynAckSession** existentes simultaneamente que configuram um ataque, foi estimado em 180, o que equivale a 15% do total do *backlog*. Este número variará de acordo com a freqüência de acessos ao servidor. Servidores com alta taxa de acessos devem elevar esse valor.

Outro parâmetro a ser determinado é o valor de **tcpMaxUniqueOpen**. Esse valor representa o número mínimo de conexões com *handshake* incompleto no estado *SYN RECEIVED* partindo de um único *IP*, de forma a se configurar um ataque. Observou-se que normalmente poucas perdas de pacotes ocorrem devido à alta confiabilidade do protocolo *TCP*. Além disso, máquinas normalmente não enviam requisições SYN a uma taxa de tempo alta para outras máquinas. Sendo assim, atribuiu-se o valor de **tcpMaxUniqueOpen** como 30, equivalente a 3% do *backlog*. Ao atingir esse valor, a aplicação irá incluir uma regra no *iptables* descartando os pacotes *SYN* originados do *IP* de origem dos pacotes.

Os valores de 15% e 3% supracitados foram obtidos experimentalmente e são os que permitiram uma resposta rápida ao ataque com uma capacidade baixa de produzir falsos positivos.

Para determinar quanto tempo a regra acima ficará ativa, utilizou-se o parâmetro **synTimeout**. Baseado na quantidade de tentativas de conexões que o sistema operacional irá realizar (5 tentativas) e no valor máximo assumido pelo *RTO* (60 segundos), a primeira requisição *SYN* recebida que se originou do *IP* de origem dos pacotes maliciosos irá permanecer até 300 segundos no *backlog*. O dobro desse valor é suficiente para que todas as entradas sejam descartadas do *backlog*. Assim, o valor de **synTimeout** foi determinado como 600 segundos.

Assumindo-se os valores supracitados, a combinação da aplicação com o *iptables* conseguiu defender o ataque *Smurf* sem ter recebido um pacote malicioso sequer. Mesmo quando a máquina da vítima enviava requisições *ICMP Echo Request* para a máquina do atacante, nenhum pacote foi recebido pelo sistema. O intervalo de tempo em que se espera uma resposta legítima é insuficiente para que o atacante consiga enviar uma quantidade significativa de pacotes maliciosos, utilizando-se do ataque definido na seção 5.2.2, já que o *IP* de origem dos pacotes é gerado aleatoriamente.

Para o ataque *SYN Flood*, o ataque foi detectado nos dois cenários e a defesa foi realizada no cenário 1. O tempo levado para a detecção em todos os testes foi menor do que 5 segundos.

7 CONCLUSÕES

A aplicação proposta respondeu bem aos cenários do ataque *SYN Flood* e ao ataque *Smurf*, mesmo com a limitação de não se poder realizar a defesa no cenário 2. Não foi possível determinar se a aplicação suportaria ataques mais intensos e complexos, entretanto a utilização de sessões para os pacotes de acordo com a assinatura de um ataque de forma a detectar ataques e executar comandos de *firewall* mostrou-se uma estratégia eficiente de acordo com os resultados obtidos.

O ambiente utilizado permitiu uma avaliação de forma satisfatória dos parâmetros. Entretanto, o uso da rede por terceiros, apesar de tornar o ambiente mais real, e a utilização de máquinas virtuais, que tinham de compartilhar processamento entre si e com as máquinas hospedeiras, vieram a prejudicar o desempenho dos testes em alguns casos em que prejudicavam o rendimento do atacante. Melhorias nesse sentido poderiam ser feitas em um ambiente apenas com máquinas físicas e isento de máquinas de terceiros.

Cabe ressaltar também que a presença das máquinas neutras ligadas, mesmo estando ociosas, prejudicou enormemente o desempenho do atacante, inviabilizando a execução de ataques com uma taxa que pudesse gerar uma resposta rápida e precisa da vítima.

Uma forma de determinar os parâmetros de forma mais precisa poderia ser realizada caso fossem utilizadas máquinas com configurações distintas no ataque e na vítima.

Sobre os ataques de negação de serviço, observa-se que mesmo os mais simples ataques necessitam de uma defesa complexa. Muitos desses ataques poderiam ser evitados caso roteadores ou mesmo os sistemas operacionais mais utilizados bloqueassem o *IP spoofing*.

Por fim, é importante ressaltar que a segurança cibernética não depende apenas da existência de *hardwares*, *softwares* e protocolos seguros. Ela envolve também uma orientação aos usuários sobre práticas a serem adotadas e de uma configuração eficiente de redes e máquinas. Como menciona Schneier (2000), criptógrafo e especialista em segurança americano, "*Segurança é um processo, não um produto*" (tradução nossa).

8 TRABALHOS FUTUROS

A utilização de uma aplicação interligada a um *firewall* pode ser utilizada para a defesa e a detecção de outros tipos de ataques de negação de serviço.

Novos ataques mais danosos e de difícil detecção e defesa surgem de tempos em tempos, exigindo que pesquisas na área de segurança cibernética sejam sempre realizadas. A utilização de ataques DoS diminuiu principalmente devido a modificações feitas em sistemas operacionais e ao aumento da largura de banda utilizada, sendo hoje mais utilizados ataques DDoS ou DRDoS.

Há também espaço para trabalhos na área de detecção da origem do ataque. A utilização de *IP spoofing* dificulta a detecção da origem real do ataque, mas a utilização de protocolos como o *BGP* em roteadores possibilita que isso seja feito.

Além disso, outras soluções em nível de sistema operacional ou de roteador podem ser desenvolvidas ou melhoradas, assim como o estudo de uma combinação de soluções.

O estudo de métodos para a determinação de parâmetros de acordo com o ambiente permitiria que ambientes fossem protegidos de forma mais eficiente e precisa.

Por fim, o *Internet Protocol, Version 6* (*IPv6*), que, apesar de corrigir algumas falhas da versão 4, atualmente a mais utilizada, abre novas possibilidades de ataques devido à presença de novos campos em seu cabeçalho.

9 REFERÊNCIAS BIBLIOGRÁFICAS

AYUSO, Pablo Neira. **Netfilter/Iptables Project Homepage**. Netfilter.org [online]. 2010 [S.l.]. Disponível: http://www.netfilter.org/projects/iptables/index.html [capturado 03 ago. 2010].

CASTRO, Luiz Fernando Damaceno Moura e. **Estônia sofre ataque virtual**. Belo Horizonte, 2 jul. 2007. PUC Minas - Conjuntura Internacional [online]. Disponível: http://www.pucminas.br/imagedb/conjuntura/CNO_ARQ_NOTIC20070704113456.pdf?PHPSESSID=40d7b2656db775ea31268bf3df1c04cc [capturado 11 nov. 2009].

CENTRO DE ESTUDOS, RESPOSTA E TRATAMENTO DE INCIDENTES DE SEGURANÇA NO BRASIL. **Cartliha de Segurança para a Internet Versão 3.1**. CERT.br [online]. 11 jul. 2007. Disponível: http://cartilha.cert.br/ [capturado em 2 nov. 2009].

COMER, Douglas E. **Redes de Computadores e Internet**. 4. ed. São Paulo: Bookman. 2007. ISBN 8560031367.

DUARTE, Otto Carlos Muniz Bandeira. Denial of Service – Negação de Serviço [online]. [S.l.]. Disponível: http://www.gta.ufrj.br/grad/06_1/dos/ [capturado 3 dez. 2009].

DUTRA, André Melo Carvalhais. **Introdução à Guerra Cibernética: a necessidade de um despertar brasileiro para o assunto**. In: IX Simpósio de Guerra Eletrônica, 2007, São José dos Campos. Artigo. São José dos Campos: Instituto Tecnológico de Aeronáutica, 2007.

EDDY, Wesley M. **TCP SYN Flooding Attacks and Common Mitigations**. Request for Comments: 4987 [online]. Cleveland: Verizon Federal Network Systems, NASA Glenn Research Center, ago. 2007. Disponível: http://tools.ietf.org/html/rfc4987 [capturado 15 jul. 2010].

ERICKSON, Jon. **Hacking**. Tradução: Marcelo Madeira; Henriqueta Grubba. 2. ed. São Paulo: Digerati Books, 2009. ISBN 978-85-6048-30-2.

EVANS, Paul. **Type 1 and Type 2 Client Hypervisors**. ShareVM [online], 9 mar. 2010. Disponível: http://blog.sharevm.com/2010/03/09/type-1-and-type-2-client-hypervisors/ [capturado 19 ago. 2010]

GIBSON, Steve. **Distributed Reflection Denial of Service**. Gibson Research Corporation [online], 22 fev. 2002. Disponível: http://www.understandingcomputers.ca/articles/grc/drdos_copy.html [capturado 02 ago. 2010].

HOCH, Darren. **Analysis of Denial of Service**. UNIX Users Association of Southern California [online]. California, 11 abr. 2004. Disponível: http://www.ufsdump.org/papers/uuasc-november-ddos.html [capturado 18 jul. 2010].

HOLLAND, Jeff. **Netmask-based ICMP Echo Request Smurf Broadcast Scanning with Crafted ICMP Payloads using SendIP**. Whitehats.ca [online]. Disponível em: http://www.whitehats.ca/main/members/Jeff/gcia_assign_2/gcia_assign_2.html [capturado 18 jul. 2010].

HUGGINS, Simon. **SendIP**. Project Purple [online]. [S.l.], 29 jul. 2003. Disponível em: http://www.earth.li/projectpurple/progs/sendip.html [capturado 20 jun. 2010]

INFORMATION SCIENCES INSTITUTE. **Transmission Control Protocol**. Request for Comments: 793 [online]. Marina del Rey: University of Southern California, set. 1981. Disponível: http://tools.ietf.org/html/rfc793 [capturado 15 jul. 2010].

KUJII, Keita. **Jpcap – a Java library for capturing and sending network packets**. University of California, Irvine [online], 15 mai. 2007. Disponível: http://netresearch.ics.uci.edu/kfujii/jpcap/doc/ [capturado 02 ago. 2010].

LEE, Cheolho; NOH, Sanguk; CHOI, Kyunghee; JUNG, Gihyun. **Characterizing DDoS Attacks With Traffic Rate Analysis**. IADIS International Conference e-Society 2003 [online]. ISN 1611-3349. 29 set. 2004. Disponível: http://www.iadis.net/dl/final_uploads/200301L011.pdf [capturado 03 ago. 2010].

MIRKOVIC, Jelena; REIHER, Peter. **A taxonomy of DDoS attack and DDoS defense mechanisms**. ACM SIGCOMM Computer Communication Review [online], New York, v. 34, n. 2, p. 39-53, abr. 2007. ISSN 0146-4833. Disponível: http://www.cis.udel.edu/~sunshine/publications/ccr.pdf [capturado 19 fev. 2010].

MIRKOVIC, Jelena; DIETRICH, Sven; DITTRICH, David; REIHER, Peter. **Internet Denial of Service: Attack and Defense Mechanisms**. Upper Saddle River: Prentice Hall PTR, 2004. ISBN 0-13-147573-8.

NORTH ATLANTIC TREATY ORGANIZATION. Science for Peace and Security. **Cyber Terrorism – a Serious Threat to Peace and Security in the 21st Century**. Advanced Research Workshop - ARW 982439 [online]. 31 mai. 2007. Disponível:

http://www.nato.int/science/studies_and_projects/nato_funded/arw_asi_anw/arw_982439.htm [capturado 2 nov. 2009].

OLLMAN, Gunter. **North Korea Kicks-off DDoS Cyber War?** Dambala [online], 10 jul. 2009. Disponível em: http://blog.damballa.com/?p=288 [capturado 28 jul. 2010]

PARKS, Raymond C.; DUGGAN, David P.. Principles of Cyber-warfare. In: Workshop on Information Assurance and Security, 2001, West Point. **Proceedings of the 2001 IEEE**. IEEE, 6 jun. 2001. ISBN 0-7803-9814-9.

PILKINGTON, Ed. **China winning cyber war, Congress warned**. guardian.co.uk [online]. Nova York, 20 nov. 2008. Disponível: http://www.guardian.co.uk/technology/2008/nov/20/china-us-military-hacking [capturado 11 nov. 2009].

POSTEL, J. **Internet Control Message Protocol**. Request for Comments: 792 [online]. Marina del Rey: University of Southern California, Information Sciences Institute, set. 1981. Disponível: http://tools.ietf.org/html/rfc792 [capturado 15/07/2010].

PRESSE, France. **Ataques cibernéticos estão entre três maiores ameaças mundiais, diz FBI**. Folhaonline [online]. Informática. Nova York, 7 jan. 2009. Disponível: http://www1.folha.uol.com.br/folha/informatica/ult124u487415.shtml [capturado 11 nov. 2009].

REKHTER, Yakov, LI, Tony, HARES, Susan. **A Border Gateway Protocol 4 (BGP-4)**. Request for Comments: 4271 [online]. [S.l.], jan. 2006. Disponível: http://tools.ietf.org/html/rfc4987 [capturado 15/07/2010].

SANDOVAL, Greg; WOLVERTON, Troy. **Leading Web sites under attack**. CNET [online]. 9 fev. 2000. Disponível: http://news.cnet.com/Leading-Web-sites-under-attack/2100-1017_3-236683.html [capturado 11 nov. 2009].

SCHNEIER, Bruce. **Crypto-Gram Newlister**. Schneier.com [online]. [S.l.], 15 mai. 2000. Disponível em http://www.schneier.com/crypto-gram-0005.html [capturado 26 jul. 2010].

SOUTH Korean government websites, banks hit by suspected cyber attack. **The Globe And Mail** [online], 8 jun. 2009. Disponível: http://www.theglobeandmail.com/news/technology/south-korean-government-websites-banks-hit-by-suspected-cyber-attack/article1210171/ [capturado 28 jul. 2010].

TANENBAUM, Andrew S. **Sistemas Operacionais Modernos**. 3. Ed. São Paulo: Pearson, 2010. ISBN-13 9788576052371

ÜBERGEEK, Das. **Onda de ataques de hackers assolam sites governamentais do Irã**. Geek. Digerati [online]. São Paulo, 17 jun. 2009. Disponível: http://www.geek.com.br/blogs/832697632/posts/10261 [capturado 11 nov. 2009].

10 APÊNDICES

10.1 APÊNDICE 1: COMANDOS UTILIZADOS NA MÁQUINA ATACANTE

a) **#apt-get install sendip**
b) **#while true; do sendip –p ipv4 –p tcp –tfs 1 –is 192.168.0.100 –ts r –td 80 192.168.91.234; done**
c) **#while true; do sendip –p ipv4 –p tcp –tfs 1 –is r –ts r –td 80 192.168.91.234; done**
d) **#while true; do sendip –p ipv4 –p icmp –ct 0 –is r 192.168.91.234; done**

10.2 APÊNDICE 2: COMANDOS UTILIZADOS NA MÁQUINA VÍTIMA

a) **#iptables -t filter -N icmpchain**
b) **#iptables -t filter -A INPUT -j icmpchain**
c) **#iptables -t filter -A icmpchain -p icmp --icmp-type echo-reply -d 192.168.91.234 -j DROP**
d) **#iptables -t filter -I icmpchain 1 -p icmp --icmp-type echo-reply -s [IP da máquina de destino] -d 192.168.91.234 -j ACCEPT**
e) **#iptables -t filter -D icmpchain -p icmp --icmp-type echo-reply -s [IP da máquina de destino] -d 192.168.91.234 -j ACCEPT**
f) **#iptables -t filter -I synchain 1 -p tcp --syn --port [porta sendo atacada] -s [IP a ser bloqueado] -j DROP**