Capítulo 1

Negação de Serviço: Ataques e Contramedidas

Rafael P. Laufer[1], Igor M. Moraes[1], Pedro B. Velloso[1,2],
Marco D. D. Bicudo[1], Miguel Elias M. Campista[1], Daniel de O. Cunha[1],
Luís Henrique M. K. Costa[1] e Otto Carlos M. B. Duarte[1]

[1]Grupo de Teleinformática e Automação – GTA*
COPPE-Poli – Universidade Federal do Rio de Janeiro

[2]Laboratoire d'Informatique de Paris 6 – LIP6
Université Pierre et Marie Curie - Paris VI

Abstract

The Internet architecture exposes the users to different types of digital attacks and threats. Denial of service (DoS) is a common attack which may cause huge damage to a victim. The prevention of such attacks is a challenge even for overprovisioned and up-to-date computers. As a consequence, the best way to inhibit DoS attacks is to be ready to react to those attacks on the fly. One such reactive mechanism is IP traceback. Such mechanism is important, since packets with spoofed source addresses are employed to disguise the actual source of the attack. This chapter presents the most common techniques used to combat DoS attacks in the Internet. Different IP traceback systems proposed in the literature are also analyzed in detail. At last, a stateless single-packet IP traceback system is introduced.

Resumo

A arquitetura da Internet expõe os usuários a diversos tipos de ataques e pragas digitais. Ataques comuns que causam enormes prejuízos à vítima são os ataques de negação de serviço. Prevenir-se contra tais ataques é um desafio, mesmo para computadores regularmente atualizados e superdimensionados. Por isso, a melhor maneira de inibir ataques de negação de serviço é estar pronto para reagir a estes ataques. Uma forma de inibição é adotar sistemas de rastreamento de pacotes IP. Tais sistemas são fundamentais, uma vez que pacotes com endereços IP de origem forjados podem ser usados no ataque para ocultar a verdadeira origem do atacante. Neste capítulo, são apresentadas as técnicas usadas para inibir ataques de negação de serviço na Internet. São caracterizados ainda diversos sistemas de rastreamento de pacotes IP propostos na literatura. Por fim, é introduzido um novo sistema de rastreamento de pacotes IP.

*Apoiado pelos recursos da CAPES, CNPq, FAPERJ, FINEP, RNP, FUNTTEL e UOL

1.1. Introdução

A Internet é um espaço virtual de encontros e ao mesmo tempo um local de conflitos, devido à diversidade cultural dos seus usuários. Esse ambiente cria um campo fértil para ações maliciosas, que exploram falhas de segurança existentes nas aplicações e nos protocolos de comunicação da Internet. Vírus são enviados por e-mail, infectando computadores e se disseminando sem serem notados. *Spams* enchem as caixas de mensagens dos usuários da rede. Usuários mal-intencionados, que vão desde quadrilhas muito bem estruturadas a meros adolescentes em busca de auto-afirmação, tentam invadir computadores e redes para roubar informações confidenciais ou apenas mostrar seu potencial. Ataques de negação de serviço (*Denial of Service* – DoS) consomem os recursos de servidores e roteadores e impedem que usuários legítimos tenham acesso a um determinado serviço. Enfim, a lista de ataques cresce a cada dia.

Estatísticas apontam um crescimento mundial de 226% no volume de mensagens eletrônicas não solicitadas (*spams*) de abril para maio de 2005 [IBM Report, 2005]. Neste mesmo período, o aumento no volume de e-mails contendo pragas digitais aumentou de 33%. O montante de e-mails não solicitados já representa 68,7% de todos os e-mails que circulam na Internet e os que contêm pragas digitais representam 3,12%. No Brasil esse problema é bem grave, pois atualmente o país é apontado como o 5^o maior receptor de *spams* no mundo [Globo Online, 2005]. Apesar de todos os esforços para bloqueá-los, muitos ainda não são filtrados devido ao avanço de técnicas para burlar os programas anti-*spam*. Provedores acabam aumentando o custo de sua operação por disponibilizar involuntariamente recursos para armazenar e enviar *spams*. No que se refere a negação de serviço na Internet, um estudo recente estima em mais de 4000 o número de ataques em um período de uma semana [Moore et al., 2001]. Um resultado importante mostra ainda que grande parte dos ataques são concentrados em vítimas brasileiras. De acordo com as informações analisadas, o domínio `.br` é o quarto domínio mais atacado por inundações visando a negação de serviço, concentrando cerca de 5 a 7% dos ataques. Em toda a Internet, somente os domínios `.net`, `.com` e `.ro` foram mais atacados que o domínio brasileiro. Uma informação do relatório anual do CSI/FBI (*Computer Security Institute/Federal Bureau of Investigation*) [Gordon et al., 2005] sobre crimes na área de computação afirma ainda que os ataques de negação de serviço estão entre os incidentes de segurança que mais causam prejuízo às instituições americanas.

Até o momento, não há uma solução completa para a maioria dos problemas de segurança da Internet. Porém, a ocorrência e os efeitos de alguns destes ataques foram reduzidos, através da adoção de determinadas ferramentas. O uso de *firewalls* reduz o número de invasões a redes de computadores, uma vez que, com exceção dos pacotes legítimos, todo o tráfego de entrada é bloqueado. Os programas antivírus previnem a execução de vírus e vermes conhecidos no computador no qual estão rodando. Dessa forma, os antivírus impedem a infecção do computador e interrompem a disseminação de tais pragas pela rede. Da mesma forma, os programas anti-*spam* buscam reduzir o número de mensagens não solicitadas recebidas. Entretanto, tanto os antivírus quanto os anti-*spams* têm que ser constantemente atualizados, pois novas pragas e técnicas para enviar mensagens não solicitadas surgem a cada dia. Sistemas operacionais e aplicações verificam periodicamente a existência de atualizações de software e automaticamente se remendam, o que reduz o número de vulnerabilidades que podem ser exploradas em um

computador. No entanto, nenhuma das ferramentas existentes é eficaz contra ataques de negação de serviço. Apesar do empenho das áreas acadêmica e industrial na busca de soluções para evitar e remediar tais ataques, as ferramentas existentes são capazes de lidar apenas com ataques pouco refinados. Por isso, os ataques de negação de serviço são, em sua maioria, bem sucedidos. Atualmente, é fácil burlar as ferramentas de defesa e negar o serviço da vítima durante o tempo desejado pelo atacante, sem que nenhuma medida efetiva possa ser tomada.

Contra ataques de negação de serviço, podem ser tomadas medidas preventivas e reativas. Para prevenir-se de um ataque, uma possível vítima deve atualizar regularmente seus programas e possuir recursos suficientes para suportar um ataque sem prejudicar seus usuários legítimos. Mesmo assim, tais práticas não garantem que um computador estará protegido contra ataques de negação de serviço. Novas vulnerabilidades são descobertas a cada dia e, mesmo para computadores superdimensionados, não é possível garantir a imunidade da vítima quando o tráfego de ataque é gerado por diversos atacantes. Dessa forma, por ser difícil evitar ataques de negação de serviço, é necessário adotar medidas para reagir a tais ataques. Uma medida reativa é rastrear os pacotes de ataque. Como a maioria dos ataques de negação de serviço utiliza pacotes com endereços IP de origem forjados, o objetivo do rastreamento é determinar a verdadeira origem e a rota percorrida por um pacote ou um conjunto de pacotes IP. Ou seja, a partir de pacotes recebidos pela vítima, deseja-se encontrar quem realmente os enviou, de forma a aplicar punições e inibir futuros ataques.

O objetivo principal deste capítulo é apresentar os conceitos e as técnicas usadas para inibir ataques de negação de serviço na Internet. Inicialmente são abordados alguns dos principais ataques na Internet, como a disseminação de pragas digitais e *spams*. Os tipos de ataques de negação de serviço são analisados assim como as contramedidas. Em seguida, são analisados ainda diversos sistemas de rastreamento de pacotes propostos na literatura, mostrando suas características e ressaltando suas virtudes e problemas. Por fim, é apresentado um novo sistema de rastreamento de pacotes que não armazena estado na infra-estrutura da rede e que é capaz de identificar um atacante utilizando apenas um pacote empregado no ataque.

1.2. Ataques na Internet

Os ataques na Internet podem ser realizados de diversas formas e com propósitos variados. Muitos ataques visam obter informações confidenciais, se apoderar de recursos computacionais para execução de uma determinada tarefa, anunciar produtos comerciais ou inviabilizar serviços. Dependendo do tipo de ataque, o alvo principal podem ser os computadores dos usuários, os servidores ou a própria rede. Assim, de alguma forma, a operação normal da rede é prejudicada pela incapacidade de comunicação entre os computadores. Na maioria das vezes, as vítimas não têm conhecimento dos ataques e podem inclusive colaborar involuntariamente, por exemplo, ao entrarem em um sítio divulgado por um *spam* ou ao executarem um programa malicioso por engano. Nesta seção, são abordados alguns dos principais ataques da Internet, como a disseminação de pragas digitais, como os vírus e os cavalos de Tróia, e o envio de *spams*.

1.2.1. Pragas Digitais

Os ataques associados a códigos maliciosos custam bilhões de dólares a cada ano. A maioria das organizações considera que as ações corretivas para estes ataques demandam muito tempo e um alto custo operacional. Em alguns casos, a recuperação dos prejuízos causados por um ataque e a normalização das operações cotidianas podem levar vários dias. Os profissionais da área de segurança despendem enormes esforços para aprimorar os métodos de defesa dos computadores dentro da rede de uma organização. Porém, à medida que estas defesas se tornam mais eficientes, os atacantes se adaptam às novas condições buscando novos caminhos para explorar vulnerabilidades.

Os primeiros programadores de códigos maliciosos eram principalmente pessoas conhecidas como *nerds* de computadores. Os *nerds* são considerados pessoas socialmente alienadas que podem invadir sistemas computacionais por diversão, para protestar contra regras sociais ou pela simples vaidade de mostrar sua capacidade ao atacar sítios de empresas de alta tecnologia ou de governos. Atualmente, este cenário se transformou. Hoje, muitos códigos maliciosos visam benefícios financeiros, ao tentar controlar computadores alheios para divulgação de produtos ou serviços através de mensagens de correio eletrônico, outros visam o enriquecimento ilícito, ao capturar dados pessoais e senhas de usuários do sistema financeiro. As perspectivas futuras são ainda piores devido à eficiência dos ataques, à dificuldade de se chegar ao atacante e à falta de uma legislação específica para os crimes digitais. Uma verdadeira "guerra informacional" entre atacantes com objetivos ilícitos e profissionais de segurança de sistemas computacionais e de redes de computadores se anuncia. Os alvos dos ataques são quaisquer pontos críticos da infra-estrutura de comunicação e computação, seja o usuário doméstico ou uma grande organização.

Os ataques de códigos maliciosos podem ser mascarados por uma vasta gama de aplicações legítimas e transportados por um número cada vez maior de mecanismos. Em geral, estas pragas digitais impedem o funcionamento normal de um computador ou de uma rede e a sua execução é comumente realizada sem o consentimento do usuário. Entender como estas pragas digitais operam é de grande interesse para o desenvolvimento de estratégias defensivas, para a seleção de produtos de segurança e para o preparo dos usuários contra ameaças em potencial. Esta seção explica os diferentes tipos de ataques por programas maliciosos. Os tipos de ataques maliciosos que caracterizam as principais pragas digitais são: os vírus de e-mail ou de outro tipo, os cavalos de tróia ou outro tipo de porta dos fundos (*backdoors*), os vermes (*worms*), o *spyware*, o *adware* e o *stealware*.

De maneira geral, as pragas se replicam e se espalham de um computador para outro. Os vírus, os vermes (*worms*) e os cavalos de Tróia são pragas que são confundidas.

Um vírus é um trecho de código que se anexa a um programa ou arquivo, de forma a poder se espalhar e infectar outros sistemas quando é transportado. Os efeitos nocivos dos vírus são bastante variados, indo de ações irritantes como a exibição de uma imagem na tela, em momentos aleatórios, até estragos no hardware, software ou sistemas de arquivos. A maioria dos vírus são anexados a arquivos executáveis, o que significa que o vírus pode estar presente no computador mas não provocará efeitos maliciosos a menos que o programa seja executado. É importante notar que o vírus não se espalha sem a ajuda de uma intervenção humana, como a execução de um programa. As pessoas ajudam a espalhar o vírus, muitas vezes sem saber, através do compartilhamento de arquivos ou

envio de arquivos contaminados anexados a e-mails.

Muitos vírus não têm uma tarefa específica a realizar e apenas se replicam indefinidamente, congestionando o sistema de e-mail. Por outro lado, alguns vírus possuem conteúdo malicioso, realizando tarefas como apagar ou corromper dados de arquivos ou desabilitar programas de segurança. Existem vírus que modificam um trecho de código de outro programa para facilitar a sua propagação. Geralmente, os vírus apresentam a seguinte estrutura:

- um mecanismo de replicação, que permite que os vírus se espalhem;
- um disparador, que é um evento externo ou interno para indicar o início da replicação do vírus, da tarefa que este realiza ou de ambas;
- e uma tarefa, ou seja, a ação deve ser executada.

Estes três componentes podem ter diversas formas e comportamentos. Os mecanismos de replicação variam consideravelmente. Um vírus pode executar uma incontável quantidade de combinações de tarefas. No entanto, alguns tipos de vírus são mais comuns e merecem destaque.

O vírus de setor de inicialização, ou setor de *boot*, infecta o primeiro setor de um disco rígido ou de um disquete. O primeiro setor destes discos contém o registro de inicialização mestre (*Master Boot Record* - MBR), que realiza configurações iniciais antes do carregamento do sistema operacional. Assim, quando o computador é ligado, o vírus é iniciado, antes mesmo do sistema operacional, tomando controle sobre qualquer recurso necessário. Geralmente, esse tipo de vírus infecta todo disquete que é inserido no computador. Um vírus de setor de *boot* geralmente não é transportado pela rede, sendo, portanto, cada vez mais incomum atualmente devido ao desuso do disquete.

Os vírus removedores têm a tarefa de apagar arquivos com nomes específicos que são necessários à execução básica do sistema operacional ou são essenciais a um editor de texto ou gráfico, por exemplo. Os vírus infecciosos geralmente se anexam a arquivos executáveis com extensões `.com` e `.exe`. Assim, a cada execução do programa infectado, o vírus se espalha ainda mais pelos arquivos do computador. Os vírus embutidos em conteúdo (*content-embedded viruses*) são vírus-infecciosos que residem, ou estão anexados, a arquivos de imagem, páginas em HTML (*HyperText Markup Language*), arquivos de vídeo ou arquivos de som. Os vírus de macro são semelhantes aos vírus embutidos em conteúdo, pois estão anexados a arquivos de editores como o Microsoft Word ou planilhas do Microsoft Excel. Como estas macros estão geralmente anexadas aos documentos e são executadas pelos programas, esses vírus podem utilizar disquetes, e-mail ou até um servidor de arquivos como meio de difusão.

Os vírus de e-mail em massa (*mass mailers viruses*) executam procedimentos do programa cliente de e-mail para se auto-replicar e enviar, através de um e-mail, para todos os endereços armazenados na agenda do usuário. Existe uma diferença no nível de ameaça entre vírus de e-mail em massa e vírus de e-mail comum. Ambos usam o mesmo método de replicação e de transporte, porém o que envia e-mails em massa é considerado mais

perigoso devido a sua taxa de replicação, à sobrecarga do sistema de correio eletrônico e à inundação da caixa de entrada dos usuários.

Os vírus furtivos (*stealth viruses*) se escondem dos softwares de detecção de vírus. Estes vírus são projetados para burlar as heurísticas utilizadas para detecção de vírus desconhecidos pelos softwares antivírus.

Os vírus de engenharia social são outro tipo cada vez mais comum. Junto com os boatos (*hoaxes*), que são definidos como brincadeiras ou "pegadinhas", este tipo de praga virtual convida o usuário a baixar e executar códigos maliciosos ou entrar em páginas que contenham códigos maliciosos.

Além destes tipos de vírus, existem vírus de características múltiplas, que combinam as funcionalidades e os atributos descritos acima.

O cavalo de Tróia tem este nome porque, tal como na lenda mitológica, o "presente" recebido aparentemente não oferece nenhum perigo. Ou seja, à primeira vista o programa que contém um cavalo de Tróia parece útil, no entanto uma vez instalado ele causará estragos no computador. Os cavalos de Tróia são códigos maliciosos que executam comandos não-autorizados no computador-alvo. Normalmente, as pessoas que instalam um programa com cavalo de Tróia o fazem por pensarem estar instalando software de uma fonte legítima. Os efeitos dos cavalos de Tróia podem ir desde simples ações irritantes, como a modificação da área de trabalho ou a criação de ícones indesejados, até prejuízos mais sérios como o apagamento de arquivos e destruição de informações. Alguns cavalos de Tróia podem criar "portas dos fundos" (*backdoors*) para liberar o acesso ao computador para usuários maliciosos executarem comandos remotamente, possivelmente permitindo que informação confidencial seja obtida. Diferentemente de vírus e vermes, os cavalos de Tróia não se replicam através da infecção de outros arquivos nem se auto-replicam.

Um verme é similar a um vírus em sua estrutura e pode ser considerado uma subclasse do vírus. O verme se alastra de um computador para outro, mas ao contrário de um vírus, o verme tem a capacidade de se alastrar sem a ajuda da intervenção humana. O verme se aproveita de algum arquivo ou de procedimentos de transferência que existem no sistema do computador que permitem que ele se propague sem nenhuma ajuda, ou seja, de forma autônoma. Estes computadores geralmente apresentam falhas de segurança exploradas pelo código malicioso do verme, que utiliza uma rede local ou uma conexão com a Internet. Quando o verme encontra um computador vulnerável (por exemplo, através de uma porta dos fundos), se replica, transfere sua réplica para o computador remoto e continua sua busca por novos alvos. A grande ameaça do verme é sua replicação no computador e, como conseqüência, o esgotamento da memória ou da capacidade de processamento da máquina. Devido à capacidade de replicação do verme, em vez do computador transferir um único verme ele pode transferir centenas ou milhares de suas cópias. Um exemplo de verme é o que faz transferência de uma cópia para toda a lista de correios eletrônicos que existe no computador.

O termo software-espião (*spyware*) designa qualquer programa que coleta informações sobre um indivíduo ou organização sem seu consentimento prévio. O software-espião pode ser instalado em um computador por diversos meios. O espião pode ser

introduzido como parte de um vírus, verme, cavalo de Tróia ou até mesmo como parte de programas legítimos. Vale ressaltar que os termos *spyware*, *adware* e *stealware* são comumente utilizados para descrever o mesmo tipo de código malicioso ou tipos semelhantes desta praga digital. Vários países estão em vias de banir legalmente, ou pelo menos controlar, este tipo de abuso digital. Os softwares-espiões são usados geralmente para colher informações com o fim de gerar estatísticas sobre senhas, hábitos do usuário, sítios da Internet acessados por um grupo de usuários de características específicas, sistema operacional utilizado, aplicações instaladas nos computadores, etc. Também podem ser coletadas informações de nomes e sobrenomes dos usuários, números de cartões de crédito e outras informações pessoais. Estas informações são geralmente combinadas com outras bases de dados de usuários e consumidores potenciais, criando assim perfis de indivíduos. Estes perfis podem então ser usados para fins de *marketing* e pesquisa de mercado.

O termo *adware*, que vem do inglês *advertisement software*, designa qualquer software ou parte de software, não necessariamente malicioso, que exibe propagandas enquanto uma aplicação principal é executada. Estas aplicações exibem na tela do usuário janelas de navegadores (*pop-ups*) ou do ambiente gráfico do usuário com anúncios de organizações patrocinadoras do software. Os *adwares* são geralmente encontrados em programas *shareware*, que são aplicativos que possuem licença de utilização temporária, ou programas *freeware*, que são gratuitos. Ambos estão comumente disponíveis na Internet. Através da comercialização destes anúncios, pequenas empresas de software ou programadores autônomos podem financiar o desenvolvimento de um software. Porém, um *adware* não designa apenas a utilização de aplicações como meio de propaganda. Um *adware* também pode se servir dos chamados defeitos de *Web* (*Web bugs*) para coletar informações sobre usuários de computadores e compilar perfis de usuários, se assemelhando ao comportamento de um *spyware*. Por exemplo, existem sítios da Internet que requisitam, às vezes compulsoriamente, aos usuários que acessam suas páginas que aceitem *cookies*, que são pequenos arquivos armazenados para monitorar o acesso dos usuários. Alguns desses *cookies* são considerados defeitos de *Web*, pois podem ser explorados como fonte de informação sobre o usuário e suas ações.

O *stealware* é um programa "ladrão de audiência". Na Internet, muitas vezes o sítio de uma organização possui atalhos (*links*) para o sítio de patrocinadores da organização. A organização é remunerada pela empresa patrocinadora através da contagem dos cliques que os usuários realizaram no *link* do patrocinador. O *stealware* burla este mecanismo, de forma a contabilizar os cliques realizados para uma outra organização, diferente da organização legítima. A identificação da organização é geralmente realizada por informações armazenadas na estação do usuário, como *cookies*, ou através do próprio endereço da página *Web* acessada. Normalmente, o *stealware* é embutido em softwares que alguns sítios obrigam o usuário a instalar, como requisito para liberar o acesso do usuário ao sítio. O usuário, ao executar o software que permite acesso ao conteúdo restrito, executa também o código malicioso. Tudo o que o *stealware* tem a fazer então é modificar os *cookies* na máquina do usuário ou forjar o endereço da página acessada.

1.2.2. Mensagem Eletrônica Não Solicitada ou *Spam*

O *spam*[†], ou mensagem eletrônica não solicitada, é toda a mensagem enviada sem a autorização do destinatário. Dentre as mensagens eletrônicas que são utilizadas como *spams* estão o correio eletrônico (e-mails), os torpedos (*Short Message Service* – SMS ou *Multimedia Message Service* – MMS) e as mensagens instantâneas (*Instant Messenger*). O indivíduo que gera e envia *spams* é chamado de *spammer*.

A definição de *spam* pelo Grupo Brasil AntiSPAM é toda mensagem eletrônica que não segue o código de ética antiSPAM [Grupo Brasil AntiSPAM, 2005], ou seja, que atenda pelo menos dois dos itens a seguir:

- o remetente é inexistente ou possui identidade falsa;
- o destinatário não autorizou previamente o envio da mensagem;
- o destinatário não pode optar em não receber mais a mensagem;
- o assunto não condiz com o conteúdo da mensagem;
- a sigla NS (Não Solicitado) está ausente no assunto de uma mensagem que não foi previamente requisitada;
- o remetente não pode ser identificado;
- uma mensagem semelhante foi recebida anteriormente em menos de dez dias apenas com o remetente ou assunto diferentes.

Nos Estados Unidos, o órgão responsável pela legislação anti-*spam* é o FTC que definiu o estatuto CAN-SPAM (*Controlling the Assault of Non-Solicited Pornography and Marketing Act*) [FTC, 2005] para o controle dos *spams*. O estatuto CAN-SPAM estabelece regras para o envio de e-mails não solicitados. O remetente que não respeitar as regras definidas está sujeito a penas criminais. De acordo com as regras estabelecidas, uma mensagem eletrônica deve conter informações verdadeiras a respeito da sua origem. O assunto da mensagem deve estar relacionado com o próprio conteúdo. Caso seja uma propaganda, uma mensagem eletrônica deve indicar claramente o seu propósito. O endereço físico do remetente deve estar presente na mensagem para possíveis reclamações ou denúncias. Por fim, uma mensagem deve fornecer ao destinatário a opção de não receber mais mensagens semelhantes do mesmo remetente.

[†]O uso da palavra *spam* para denotar mensagens eletrônicas não solicitadas é de origem controversa. A palavra *spam* é uma marca registrada [Hormel Foods, 2000] pela Hormel Foods LLC. A Hormel Foods LLC é uma empresa de alimentos e o nome *spam* pode ter surgido de uma contração das palavras "*SPiced hAM*" que batizou um dos seus produtos. O produto *spam* se tornou conhecido em 1937 durante uma campanha publicitária e em seguida pela utilização durante a segunda guerra mundial pelo exército americano. Como a carne era racionada às tropas, o alimento largamente utilizado era o *spam*. Todos os soldados americanos ao retornar aos EUA recebiam uma medalha que ficou conhecida como medalha *spam*. Como a condecoração foi recebida por diversas pessoas, a palavra *spam* ficou associada a algo comum [Holmes, 2005]. Anos mais tarde, o grupo de comédia Monty Python filmou uma cena onde a palavra *spam* é repetida muitas vezes em pouco tempo. Assim, a palavra *spam* se tornou sinônimo de algo repetitivo e sem sentido como a maioria das mensagens eletrônicas não solicitadas.

A prática de enviar *spam* tem crescido na Internet a ponto de seus usuários manifestarem insatisfação e uma menor credibilidade no correio eletrônico. Os *spams* são de conteúdo variados como informações políticas, religiosas, culturais etc. No entanto, a maioria dos *spams* atuais possui conteúdos publicitários de produtos ou serviços, fraudes e atividades ilícitas. Esta seção descreve como os *spammers* obtêm as listas de endereços, porque é tão fácil enviar *spams*, como os *spammers* mantêm o anonimato e as possíveis medidas para combater o envio de *spams*.

Obtenção de Endereços e a Escolha das Vítimas

A prática do envio de e-mails não solicitados tornou-se uma atividade lucrativa não só com a venda dos produtos e serviços anunciados como também com a própria atividade de criação e envio de *spams*. Muitas empresas pagam *spammers* para criar mensagens eletrônicas para promover as suas ofertas. Outra forma de lucrar com os *spams* é através da obtenção de endereços eletrônicos de usuários da Internet. Para tal, os *spammers* podem utilizar diversas técnicas. Dentre elas estão a invasão de servidores para aquisição de listas de endereços de possíveis membros associados, a interceptação de e-mails de grupos e ainda a utilização de programas populares como o ICQ [ICQ, 2005], o MSN Messenger [Microsoft, 2005] e o Orkut [Büyükkokten, 2005]. Utilizando e-mails de grupos, o *spammer* pode obter diversos endereços a partir de listas em cópia. Conforme o e-mail vai sendo encaminhado, maior é o número de endereços presentes. Já os programas populares citados acima também podem ser usados por *spammers*, pois eles oferecem mecanismos automáticos de busca por nomes e geram listas de endereços associados.

Após obter os potenciais endereços dos destinatários, é necessário saber se eles são ativos. Para saber se um usuário é ativo existem diversas técnicas utilizadas pelos *spammers*. Uma delas consiste em enviar um atalho para um endereço de um sítio (*link*) em uma figura junto com o endereço de e-mail do destinatário conforme a Figura 1.1. Caso o destinatário tente visualizar a figura, ele irá baixá-la do sítio e abrir uma conexão HTTP (*HyperText Transfer Protocol*) com o servidor de imagens do *spammer*. Assim, através dos seus registros de HTTP o *spammer* saberá que o usuário existe, quem ele é e que figura ele visualizou. Esse procedimento permite validar o endereço como ativo e, ao mesmo tempo, criar um perfil associado a este endereço pelo tipo de figura que foi selecionada. Algumas configurações de programas de correio eletrônicos permitem exibir a mensagem automaticamente facilitando a obtenção dos endereços pelos *spammers*. Os *spammers* podem ainda verificar a existência do usuário ao receber a solicitação de não recebimento de um determinado e-mail. Assim, regras estabelecidas por legislações anti-*spam*, como o CAN-SPAM, podem ser utilizadas em benefício do próprio *spammer*. Para evitar esse tipo de rastreamento, um usuário nunca deve visualizar um *spam*.

```
<a href="http://www.meusite.com.br?email=destinatario@email.com.br">
<img src="http://www.meusite.com.br/imagem/figura.gif?email=destinatario@email.com.br"
height=1 width=1 alt=""><\a>
```

Figura 1.1. Técnica para descobrir se o usuário está ativo.

Os valores das listas de endereços aumentam conforme o número de endereços e a

qualidade dos perfis dos destinatários. Assim, *spams* específicos podem ser direcionados para determinadas listas de endereços. Algumas empresas oferecem uma porcentagem sobre cada produto vendido para a pessoa que obteve a lista [Spammer-X et al., 2004]. O perfil do usuário pode ser também obtido através da invasão de servidores específicos, como o servidor de um sítio de astrologia, sítio de jogos, etc.

Evolução do *Spam* e suas Finalidades

A simplicidade do protocolo SMTP (*Simple Mail Transport Protocol*) exige o conhecimento de poucos comandos para se enviar um e-mail, bastando apenas a abertura de uma conexão *Telnet* com um servidor SMTP. Como não é verificado se o usuário que originou o e-mail existe, torna-se fácil gerar um *spam* utilizando outros nomes de remetentes. Essas características facilitaram o surgimento de técnicas para se gerar e enviar e-mails com finalidades lucrativas e ainda manter o anonimato.

Inicialmente, os *spams* eram em maioria e-mails em HTML com uma figura chamativa ou apenas uma frase e um atalho que indicava onde encontrar mais do conteúdo anunciado. Conhecendo as preferências do destinatário, torna-se mais fácil influenciá-lo. A Figura 1.2 mostra um exemplo de um *spam* em HTML.

```
<HTML>
<head><title> Interessado em dinheiro fácil?</title> </head>
<body>
    <a href="http://www.meusite.com.br/dinheiro">
     Clique aqui e entre para o mundo do dinheiro fácil!!! </a>
</body>
</HTML>
```

Figura 1.2. Exemplo de um *spam* em HTML.

Para hospedarem os sítios, os *spammers* utilizavam seus próprios computadores e conexões discadas. Esperava-se que os usuários pagariam a assinatura ao acessar o sítio anunciado pelo *spam*. Desde o princípio, o número de e-mails já era grande o suficiente para garantir que se apenas uma pequena parcela dos destinatários fizesse a assinatura, o lucro já seria grande para o dono do sítio. O número de e-mails não solicitados aumentou rapidamente em virtude das vulnerabilidades da rede. O aumento no número de *spams* gerou a insatisfação dos usuários da rede motivando os desenvolvedores de software a criar programas anti-*spam* e programas adicionais (*plug-ins*) para servidores. As primeiras medidas para evitar os *spams* foram a ameaça do cancelamento das contas dos usuários que enviavam *spams* através de provedores da Internet e a imposição de termos e condições para as empresas promoverem seus produtos. A criação de listas negras (*Realtime Black List* - RBL) também dificulta a hospedagem dos sítios. Se os *spammers* enviarem e-mails com identidade ou IP próprios, eles podem entrar em listas negras. Conseqüentemente, todos os seus e-mails passam a ser considerados *spams* e, portanto, são bloqueados.

Para não entrar em listas negras e não serem encontrados, os *spammers* têm que achar locais onde hospedar os seus sítios e os respectivos arquivos. Se o sítio for hospe-

dado em provedores convencionais, a conta do *spammer* será cancelada dado que denúncias são feitas ao provedor por vítimas de *spams*. Provedores convencionais não permitem que seus assinantes sejam *spammers*, pois podem ser penalizados por infringir políticas anti-*spam*. Assim, os *spammers* utilizam provedores que se propõem a hospedar o conteúdo dos seus sítios. Esses provedores cobram caro por esse serviço e hospedam sítios com qualquer conteúdo ignorando a possibilidade de serem penalizados. Algumas outras técnicas podem ser utilizadas por *spammers* para hospedar seu conteúdo. Os *spammers* podem hospedar seu conteúdo pela Internet em computadores pessoais ou em servidores sem o conhecimento do proprietário. Para tal, é necessário que o computador seja invadido ou que o dono do computador instale inconscientemente programas que dão direitos aos *spammers* de utilizar o computador infectado. Por conseguinte, os *spammers* podem hospedar seu conteúdo gratuitamente.

O surgimento de penas criminais e mecanismos anti-*spams* como as listas negras trouxeram a necessidade de se manter o anonimato no envio dos *spams*. O anonimato é importante visto que os *spams* podem conter informações falsas, oferecer produtos inexistentes e ainda utilizar identidades de terceiros configurando em atividades sujeitas a penas criminais. Para evitar também que o seu endereço IP ou domínio seja bloqueado por outros servidores de e-mail, é necessário que o *spammer* mantenha o anonimato e utilize outros computadores para enviar os seus *spams*.

A primeira forma de enviar *spams* mantendo o anonimato foi através de servidores *proxy* transparentes que utilizam protocolos como o Socks v5 [Leech, 1996]. Utilizando o Socks v5 em sua configuração padrão, um *proxy* transparente permite que qualquer computador na Internet o utilize para encaminhar seus pacotes para qualquer servidor de e-mails, sem se identificar. Para evitar que esse *proxy* seja rapidamente adicionado a listas negras, o *spammer* utiliza alternadamente diversos servidores *proxy* vulneráveis para enviar os seus e-mails. Outra estratégia empregada para enviar *spams* é utilizar servidores em países com línguas diferentes para dificultar a comunicação entre o servidor de e-mail atacado e o responsável pelo *proxy* utilizado.

Utilizar servidores SMTP com *relay* aberto é uma outra forma de enviar *spams*. Ao enviar um e-mail, o *spammer* utiliza o servidor SMTP com *relay* aberto para reencaminhar o seu e-mail mantendo o anonimato, de forma similar ao envio por servidores *proxy*. O emprego de *relays* SMTP era bastante comum nos provedores de Internet permitindo que qualquer usuário da Internet os acessassem.

Para não serem usados por *spammers*, tanto os servidores *proxy* quanto os SMTP não devem encaminhar fluxos de dados de computadores externos à sua rede local e devem utilizar *firewalls*. É importante notar que os endereços IP de origem não podem ser forjados porque o SMTP funciona sobre o TCP (*Transmission Control Protocol*), portanto é necessário o estabelecimento de conexão.

Atualmente, servidores *proxy* e servidores SMTP com *relay* aberto não são tão utilizados por *spammers*. Assim, como os *spammers* conseguem facilmente encontrá-los na rede, eles também são rapidamente adicionados em listas negras de provedores da Internet. Algumas técnicas que também podem manter o *spammer* anônimo são exploradas. Os *spams* podem ser enviados através de programas de bate-papo como o ICQ ou o MSN Messenger por usuários maliciosos. Computadores que utilizam faixas de endereços IP

dinâmicos também podem ser utilizados para envio de e-mails não solicitados, pois garantem o anonimato da fonte. Diferentes endereços IP dentro de uma faixa são fornecidos aos usuários de grandes provedores na Internet cada vez que ele se conecta. Assim, não há como garantir que um determinado usuário é realmente o *spammer*. Para combater essa prática, é necessário bloquear faixas de endereços IP ou domínios. Porém, nem sempre essa atitude é possível porque na maioria das vezes essa faixa pertence a algum grande provedor de acesso.

Uma técnica elaborada de envio de *spams* consiste em se apropriar de um roteador e de uma faixa de endereços IP. Os *spammers* adquirem faixas de endereços IP válidos que não estejam sendo utilizados e invadem um roteador com falhas de segurança. Assim, ele manipula o pedido de atualização de rotas e anuncia a todos os outros roteadores vizinhos que o roteador invadido é a única rota para a faixa de endereços IP adquirida maliciosamente. O *spammer* configura o roteador para transferir todo os fluxos de dados gerados a partir de *spams* para o seu computador, toma posse de uma faixa de endereços IP e de um roteador para uso próprio. Quando um dos endereços IP da faixa é adicionado a uma lista negra, ele passa a utilizar outro endereço IP da faixa adquirida. Como não há relação entre endereços IP e regiões geográficas, é dificil de se localizar o roteador invadido a partir do seu endereço IP.

Mecanismos Anti-*spam*

Para se evitar *spams*, muitas empresas investem no desenvolvimento de mecanismos anti-*spam*. Existem diversas formas de se evitar os *spams*, porém todas elas se baseiam na filtragem de mensagens não solicitadas. A filtragem pode ser realizada de acordo com os campos do cabeçalho e o conteúdo de um e-mail.

A filtragem pelo o cabeçalho explora principalmente os campos de endereço de origem, nome do remetente e assunto do e-mail. Um usuário pode configurar seu cliente de e-mail para filtrar todas as mensagens provenientes de endereços ou remetentes indesejados. Além dessa filtragem, os programas anti-*spam* podem ainda verificar a consistência das informações contidas no cabeçalho. Por exemplo, se o endereço IP de origem não condiz com o domínio, esse e-mail deve ser filtrado. Se o e-mail tiver sido encaminhado por um *proxy*, o e-mail também pode ser bloqueado devido às conhecidas vulnerabilidades.

A consistência de informações contidas no cabeçalho pode ser verificada através do DNS (*Domain Name System*) reverso. O DNS reverso deve estar sempre configurado para apontar o endereço IP correspondente. Como os *spammers* devem manter o anonimato, o DNS reverso de seus computadores não são configurados, ou seja, eles não apontam o seu endereço IP. Os programas anti-*spam* bloqueiam então todo o e-mail cujo remetente não possui DNS reverso. Muitos provedores de acesso utilizam faixas de endereços IP dinâmicos e, por conseguinte, seus clientes utilizam um DNS reverso genérico adotado pelo provedor. Como utilizar endereços IP dinâmicos é uma forma de manter anonimato usado por *spammers*, os anti-*spams* normalmente bloqueiam esse tipo de conexão. Um dos problemas dessa abordagem é que diversos administradores de redes não configuram o DNS reverso, gerando falsos positivos.

Os programas anti-*spam* podem também utilizar listas para fazer filtragem por ca-

beçalho. Existem três tipos de listas: as negras, as cinzas e as brancas. As listas negras são utilizadas para bloqueio incondicional dos e-mails. Sempre que um e-mail é recebido proveniente de um endereço ou domínio, contido numa lista negra, ele é bloqueado. Os domínios ou endereços IP são adicionados às listas negras após se verificar um alto número de *spams* provenientes dessa fonte. Já os programas que utilizam listas cinzas (*Gray Lists*) bloqueiam sempre toda a primeira conexão SMTP desconhecida e enviam ao servidor SMTP do remetente uma mensagem de erro temporária. Caso o servidor de e-mail seja legítimo, ele irá reenviar mais tarde o e-mail. Para evitar a recepção de milhares de mensagens de erro, alguns *spammers* não utilizam servidores SMTP. São usados clientes que se conectam diretamente ao servidor SMTP do destinatário. Como os *spammers* não recebem as mensagens de erro temporárias, eles não retransmitem e-mails bloqueados por listas cinzas. As listas brancas (*White Lists*) são listas que contêm os nomes dos servidores e domínios confiáveis. Sempre que um e-mail é recebido proveniente de algum dos endereços IP ou domínios pertencentes à lista branca, ele é aceito.

Caso o assunto do e-mail possua palavras exploradas em e-mails comerciais, esses serão também filtrados.

O bloqueio por conteúdo é realizado através da detecção de palavras comerciais comuns no corpo dos e-mails, reconhecimento de padrões ou pela presença de anexos suspeitos. E-mails com palavras como “Viagra” são facilmente bloqueadas, pois na maioria das vezes não são requisitados. Uma forma de burlar um sistema anti-*spam* é mudar letras de palavras conhecidas como “Viagra” por outras com caracteres semelhantes como “\/iagra”. Uma outra forma de burlar o bloqueio por conteúdo é colocar o anúncio numa figura, assim o corpo e o assunto do e-mail não contêm nenhuma evidência procurada. Alguns anti-*spams* são configurados para sempre bloquear arquivos anexados porque o anúncio pode estar disfarçado como figura ou algum outro tipo de arquivo.

Outra forma de filtragem por conteúdo é através da análise de diversos e-mails enviados para destinatários distintos. Caso um padrão semelhante seja observado, há uma grande probabilidade de ser um e-mail não solicitado enviado a diversas pessoas. Como a maioria dos filtros por conteúdo é determinística, adicionando-se aleatoriedade a um *spam* evita-se que um comportamento ou conteúdo comum seja detectado. Para tornar os anti-*spams* adaptativos, mecanismos baseados em técnicas de inteligência artificial são empregados para filtrar os *spams* de acordo com o seu conteúdo.

Aproximações baseadas em técnicas de inteligência artificial usando redes Bayesianas para classificar e-mails vêm sendo propostas [Sahami et al., 1998]. Através das redes Bayesianas é possível a adaptação da classificação do e-mail ao longo da utilização do anti-*spam*. Baseado numa lista de regras e em experiências passadas é atribuída uma probabilidade de um e-mail ser um *spam* a cada uma das regras. Cada e-mail é comparado com cada regra para decidir se o e-mail se trata de um *spam*. Ao final da análise, a probabilidade do e-mail ser um *spam* é computada baseada em todas as comparações. As probabilidades de cada regra são atualizadas a cada e-mail de acordo com a ação do usuário, isto é, se um usuário exclui ou não o e-mail.

Os anti-*spams* podem tornar-se mais eficientes se aplicarem diversas técnicas em conjunto. Esse é o princípio de programas como o Spam-Assassin [SpamAssassin, 2005] que utiliza técnicas de filtragem por cabeçalho e por conteúdo. Novos conceitos baseados

em confiança podem ser aplicados aos programas anti-*spam*. O conceito de confiança reflete o quanto se acredita no remetente ou no endereço IP de origem do e-mail. A confiança pode então se tornar uma métrica dinâmica atualizada ao longo da operação do mecanismo anti-*spam*. Conforme se recebe e-mails de um mesmo remetente ou domínio, avalia-se o número de *spams* recebidos. Caso esse número seja alto, a confiança é baixa; do contrário, a confiança é alta. Cabe a cada usuário definir qual é o grau de confiança necessário para receber os e-mails.

1.3. Negação de Serviço

Diferentemente da maioria dos ataques da Internet, um ataque de negação de serviço não visa invadir um computador para extrair informações confidenciais, como números de cartões de crédito e senhas bancárias, e nem para modificar o conteúdo armazenado neste computador, como sítios da Internet. Tais ataques têm como objetivo tornar inacessíveis os serviços providos pela vítima a usuários legítimos. Nenhum dado é roubado, nada é alterado e não ocorre nenhum acesso não autorizado ao computador da vítima. A vítima simplesmente pára de oferecer o seu serviço aos clientes legítimos, enquanto tenta lidar com o tráfego gerado pelo ataque [Mirkovic et al., 2004]. Um serviço pode ser o uso de um buscador de páginas, a compra de um determinado produto ou simplesmente a troca de mensagens entre duas entidades. O resultado de um ataque de negação de serviço pode ser o congelamento ou a reinicialização do programa da vítima que presta o serviço, ou ainda o esgotamento completo de recursos necessários para prover o seu serviço.

Os primeiros problemas com ataques de negação de serviço na Internet surgiram no início dos anos 90. Nesse período, foram desenvolvidos os primeiros programas para ataques remotos. Para usar estes programas e obter um grande impacto, o atacante precisava de um computador de grande capacidade conectado a uma rede de alta velocidade. Esses dois requisitos são encontrados em computadores de universidades, o que acarretou em um grande número de roubo de senhas de usuários nestas instituições. Hoje, os ataques de negação de serviço são comuns e os prejuízos financeiros e de imagem que eles causam atingem cifras enormes. Sem sombra de dúvida, o estudo e o desenvolvimento de técnicas contra ataques de negação de serviço tornaram-se importantes temas na área de segurança.

Nesta seção são apresentadas as formas existentes de negação de serviço. São abordadas ainda as características da arquitetura da Internet que possibilitam a execução destes ataques, bem como são apresentados os principais fatores que motivam os atacantes a interferir em um serviço. Por fim, alguns dos principais ataques de negação de serviço são caracterizados e classificados.

1.3.1. Formas de Negação de Serviço

Uma forma comum de prejudicar a provisão de um determinado serviço é através do consumo de recursos essenciais para o seu funcionamento, como a memória, o processamento, o espaço em disco e a banda passante. Como estes recursos são limitados, sempre é possível inundar a vítima com um grande número de mensagens de forma a consumir quase que a totalidade dos seus recursos. Este tipo de ataque é denominado “ataque por inundação”. Nestes ataques, a aplicação, a vítima ou a própria infra-estrutura de rede

fica sobrecarregada com o tratamento das mensagens de ataque e não consegue atender ao tráfego de usuários legítimos.

Um dos fatores que dificulta a adoção de medidas contrárias aos ataques de inundação é o fato de não ser possível distinguir o tráfego de ataque do tráfego de usuários legítimos. Ou seja, os atacantes utilizam mensagens normalmente empregadas em comunicações convencionais. Desta forma, ações que priorizem o tráfego legítimo em detrimento do tráfego de ataque são difíceis de serem tomadas.

Para que um ataque de inundação seja bem sucedido, um atacante deve gerar mensagens a uma taxa superior à taxa na qual a vítima, ou a sua infra-estrutura de rede, consegue tratar estas mensagens. Este é um obstáculo para os ataques de inundação. Em vítimas com poucos recursos, é mais fácil obter o sucesso da negação de serviço, uma vez que é necessário um esforço menor para tornar o serviço indisponível. No entanto, sendo a vítima superdimensionada, como é o caso de servidores mais famosos, o sucesso de um ataque não é alcançado facilmente. Muitas vezes, o tráfego gerado por apenas uma estação de ataque não é suficiente para exaurir os recursos da vítima. Nestes casos, são necessárias diversas estações atacando em conjunto para que os recursos da vítima se esgotem. Quando diversas estações agem com um objetivo comum de atacar uma determinada vítima, tem-se um ataque de negação de serviço distribuído (*Distributed* DoS - DDoS).

Os ataques distribuídos são uma ameaça mesmo para vítimas superdimensionadas. Sempre é possível inundar uma vítima e negar o seu serviço desde que um número suficiente de estações participe do ataque. Alguns sítios famosos como Yahoo, Ebay, Amazon.com e CNN.com já foram alvos de ataques de negação de serviço distribuídos bem sucedidos [Garber, 2000], [CNN.com, 2000]. Hoje em dia, diversas ferramentas que automatizam ataques distribuídos podem ser encontradas na Internet [Dittrich, 1999a], [Dittrich, 1999c], [Dittrich, 1999b]. Com isso, o nível de conhecimento necessário para se iniciar ataques distribuídos é reduzido drasticamente, fazendo com que atacantes inexperientes consigam inundar vítimas com poucos comandos.

Uma segunda maneira de se negar um serviço é através da exploração de alguma vulnerabilidade existente na vítima, os chamados “ataques por vulnerabilidade”. Ao invés de inundar a vítima de forma a consumir seus recursos, tais ataques se aproveitam de determinadas vulnerabilidades para tornar seus serviços inacessíveis. Na maioria dos casos, estas vulnerabilidades são o resultado de falhas no projeto de determinada aplicação, do próprio sistema operacional ou até mesmo de um protocolo de comunicação [Watson, 2004], [Gont, 2004]. Diversas vulnerabilidades deste gênero têm sido identificadas e exploradas. Um caso particular dos ataques por vulnerabilidades são aqueles que exigem somente um pacote para causar a negação do serviço [CERT, 1996], [Gont, 2004], [Cisco, 2003], [US-CERT, 2005]. Neste caso, ao receber um pacote contendo determinadas características que ativam a vulnerabilidade, a vítima se encontra em uma situação imprevista e reage diferentemente em cada caso. Como resultado, o serviço pode ser indefinidamente interrompido até que uma intervenção manual ocorra. Através do envio periódico do mesmo pacote para a vítima, tais ataques podem ser prolongados até que um método de correção da vulnerabilidade seja divulgado. Estes ataques apresentam um grande desafio para sistemas de defesa, especialmente para os sistemas de

rastreamento, uma vez que estes dispõem de apenas um pacote para identificar o atacante.

1.3.2. A Negação de Serviço e a Arquitetura da Internet

Três características são responsáveis pelo sucesso da Internet: a velocidade de transmissão de dados, a confiabilidade e o compartilhamento do meio com a conseqüente redução do custo de comunicação. Tais características provêm da técnica de comutação por pacotes e dos protocolos TCP/IP (*Transmission Control Protocol*/*Internetworking Protocol*). Entretanto, alguns princípios da comutação de pacotes e dos protocolos TCP/IP facilitam a ploriferação de ataques de negação de serviço.

Na Internet não há reserva de recursos em uma comunicação entre dois nós, pois se adota o "melhor esforço" como procedimento básico. Tal fato faz com que um ataque de negação de serviço busque consumir o máximo dos recursos da rede. Dessa forma, os usuários legítimos são prejudicados, uma vez que os recursos, ora usados por eles, estão sendo empregados em ações maliciosas.

O roteamento é outro fator facilitador para os ataques de negação de serviço. Na Internet, um pacote pode ser encaminhado através de qualquer rota entre o nó fonte e o nó destino. O processo de seleção de rotas é feito pelos nós intermediários e é transparente para a fonte e para o destino. Além da transparência na seleção de rotas, somente o endereço de destino é usado pela infra-estrutura de rede para que um pacote seja devidamente roteado até o seu destino. Nenhuma verificação é realizada para confirmar a autenticidade dos pacotes IP, ou seja, não existem métodos para assegurar que o endereço de origem de cada pacote IP é autêntico. Desta forma, atacantes experientes que não necessitam de comunicações bidirecionais podem se aproveitar desta característica para manter o anonimato durante ataques de negação de serviço. Tanto em ataques por inundação quanto em ataques por vulnerabilidades, atacantes podem forjar o endereço IP de origem e enviar tais pacotes para a vítima sem deixar nenhum rastro que possa identificá-los.

Como a seleção de rotas é feita pelos nós intermediários, torna-se difícil detectar e filtrar pacotes IP com endereço de origem forjado. A Figura 1.3 ilustra esse problema. Os pacotes gerados pelo nó *A* e destinados ao nó *B* são encaminhados através do roteador R_1. O roteador R_2 está localizado em outro ponto da Internet. Quando um pacote com endereço de origem do nó *A* e destinado ao nó *B* é recebido por R_2, este nó não consegue determinar se o endereço da fonte contido no pacote é forjado, ou se o roteador R_1 falhou e, por isso, os pacotes estão sendo encaminhados por outra rota. Dessa forma, o roteador R_2 encaminha normalmente o pacote para o nó *B*.

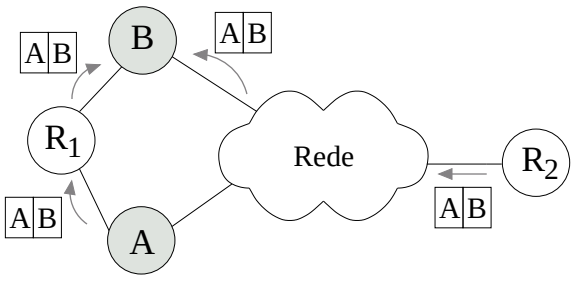

Figura 1.3. O problema do roteamento e da autenticidade dos endereços IP de origem.

A topologia da Internet também contribui para os ataques de negação de serviço. O núcleo da rede é composto por enlaces de alta capacidade. Por outro lado, os nós nas bordas são normalmente providos de enlaces de baixa capacidade e que estão conectados ao núcleo. Esta assimetria permite que os nós localizados no núcleo encaminhem pacotes de diversas origens distintas para diferentes destinos. Como resultado, ela também possibilita que os nós localizados nas bordas sejam inundados pelo tráfego agregado de outros nós. Isto é exatamente o que ocorre durante um ataque de negação de serviço distribuído. Para atacar uma determinada vítima, cada nó gera somente uma pequena parcela do tráfego agregado. Este tráfego gerado pode nem ser detectado como uma anomalia em determinadas redes e passar desapercebido pelos roteadores.

A arquitetura da Internet também dificulta a adoção de mecanismos de defesa contra ataques de negação de serviço. Na Internet, para garantir requisitos como a escalabilidade e o custo reduzido, a complexidade está localizada nos nós de borda da rede. Tal fato provém do modelo de melhor esforço, adotado para o encaminhamento simples dos pacotes na camada de rede, e do paradigma fim-a-fim, segundo o qual requisitos específicos da aplicação devem ser garantidos pelos protocolos da camada de transporte e de aplicação nos usuários finais. Por estes motivos, mecanismos de defesa localizados na infra-estrutura de rede são criticados, como será visto na Seção 1.4.

A forma como a Internet é gerenciada também cria desafios adicionais no combate aos ataques de negação de serviço. Não há hierarquia. A Internet é formada por uma comunidade de redes, interconectadas para prover acesso global aos seus usuários [Mirkovic et al., 2004]. Cada uma das redes que compõe a Internet é gerenciada localmente, de acordo com políticas definidas por seus proprietários. Neste sentido, a adoção de mecanismos de defesa contra ataques de negação de serviço, que necessitam operar em um grande número de nós da rede, é prejudicada. Como não existe uma entidade central, não há como garantir a adoção de tal sistema em larga escala.

1.3.3. A Motivação para os Atacantes

Diversos fatores, positivos ou negativos, podem motivar um ataque de negação de serviço. Um usuário bem-intencionado pode iniciar um ataque para provar que é possível inundar a vítima ou explorar alguma de suas vulnerabilidades. Em caso de sucesso, pode-se tentar encontrar uma solução para evitar que ataques originados de usuários mal-intencionados aconteçam. Uma vez encontrada a solução, ela pode ser amplamente divulgada para acelerar o processo de proteção.

Os atacantes também podem agir em busca de reconhecimento na comunidade virtual. Um atacante que consegue forçar um sítio bem conhecido a sair do ar ganha uma certa visibilidade e é bem considerado pelos colegas, podendo inclusive ser admitido em um determinado grupo de ataque de mais alto nível. Por outro lado, com a automação de fases importantes de ataques de negação de serviço distribuídos e o fácil acesso a estes procedimentos [Mirkovic e Reiher, 2004], tal tarefa ficou ainda mais fácil. Agora, até mesmo atacantes com pouca experiência são capazes de deixar suas vítimas inacessíveis, ganhando a reputação desejada.

Um ataque de negação de serviço também pode ser motivado por lucro financeiro. Um usuário com habilidade suficiente para iniciar um ataque pode deixar um determinado

servidor “fora do ar” por um certo período em troca de uma remuneração paga por terceiros. Uma empresa poderia contratar tais “serviços” e usar estes ataques para prejudicar a imagem de seus concorrentes e se beneficiar diretamente. Este procedimento pernicioso alcança requintes de organização inimagináveis, pois existem redes inteiramente formadas por computadores comprometidos, os chamados zumbis, que estão sendo alugadas de forma a serem usadas em ataques de negação de serviço [Reuters, 2004]. O trabalho de comprometer um determinado número de estações e controlá-las remotamente é realizado inicialmente e depois os recursos disponíveis em cada uma destas estações é colocado à disposição de quem quiser alugá-los para qualquer tipo de atividade, como o envio de *spams*, fraudes digitais e ataques de negação de serviço. Outro exemplo assustador é o uso destes ataques como forma de extorsão. Uma maneira que vem sendo usada freqüentemente por atacantes é exigir um pagamento de determinados sítios para evitar que seu servidor seja atacado [Shachtman, 2003]. Representantes de organizações que fazem o pagamento têm o sítio “assegurado” enquanto aqueles que não pagam sofrem consecutivos ataques de curta duração.

Motivos políticos também podem ser a causa de um ataque de negação de serviço. Um determinado grupo com idéias diferentes de uma determinada organização ou instituição pode atacar a sua rede e os seus servidores de modo a impedir a divulgação de certas informações. Como exemplo, o sítio da rede de televisão Al-Jazeera sofreu um intenso ataque de negação de serviço distribuído durante a investida americana no Iraque [Gray e Fried, 2003]. Aparentemente, o ataque foi direcionado aos servidores DNS, cuja função é converter o nome do sítio no seu respectivo endereço IP, usados pela emissora. Como conseqüência, não era possível determinar o seu endereço IP e o acesso era negado.

1.3.4. Classificação dos Ataques

Diversos fatores, como o número de atacantes envolvidos e o tipo de recurso explorado na vítima, podem ser usados para classificar os ataques de negação de serviço. Nesta seção, os ataques são classificados em: por inundação, por refletor, a infra-estrutura de redes, por vulnerabilidades e distribuídos.

Ataques por Inundação

Um dos ataques de negação de serviço mais conhecidos é o ataque por inundação de segmentos TCP SYN, que explora o procedimento de abertura de conexão do protocolo de transporte TCP. O protocolo TCP é utilizado em serviços que necessitam de entrega confiável de dados, como a transferência de arquivos. Uma conexão TCP se inicia com a negociação de determinados parâmetros entre o cliente e o servidor. A Figura 1.4 ilustra o processo de abertura de uma conexão TCP. Inicialmente, o cliente envia um segmento TCP SYN para o servidor, indicando um pedido de abertura de conexão. Este segmento leva consigo um parâmetro denominado número de seqüência inicial. O número de seqüência possibilita que o receptor reconheça dados perdidos, repetidos ou fora de ordem. Após o recebimento do segmento TCP SYN, o servidor necessita de tempo para processar o pedido de conexão e alocar memória para armazenar informações sobre o cliente. Em seguida, um segmento TCP SYN/ACK é enviado como resposta de forma a notificar o cliente que o seu pedido de conexão foi aceito. O segmento de resposta reco-

nhece o número de seqüência do cliente e envia o número de seqüência inicial do servidor. Por fim, o cliente envia um segmento TCP ACK para reconhecer o número de seqüência do servidor e completar a abertura da conexão. Esse procedimento é conhecido como o acordo de três vias (*three-way handshake*).

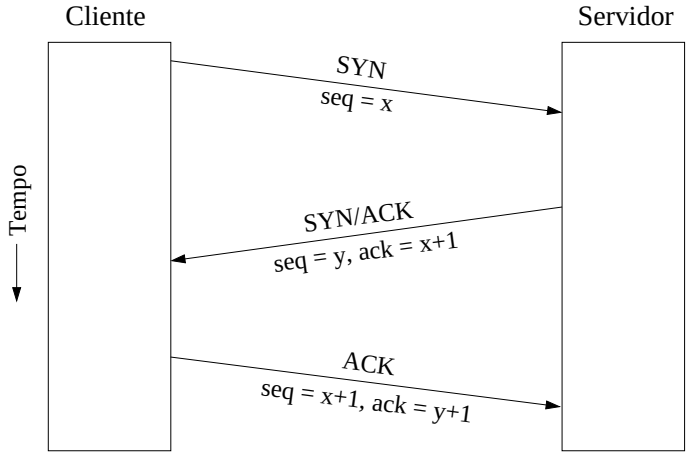

Figura 1.4. Abertura de uma conexão TCP e o acordo de três vias.

A partir deste procedimento, os recursos da vítima podem ser explorados de maneiras distintas. Para analisar com mais detalhes como cada recurso é explorado, são definidos alguns parâmetros relacionados à vítima e ao atacante. Seja t_a o intervalo de tempo entre cada segmento TCP SYN enviado pelo atacante, t_p o tempo necessário para que a vítima processe um pedido de conexão TCP e envie uma resposta, e t_m o tempo em que um determinado recurso de memória fica alocado para uma conexão TCP.

Um ataque ao processamento da vítima pode ser realizado quando o atacante consegue gerar segmentos a uma taxa mais rápida do que a vítima consegue processá-los, ou seja, quando $t_a < t_p$. A Figura 1.5 mostra um ataque de negação de serviço realizado para sobrecarregar o processamento da vítima. Neste caso, a vítima não consegue processar os pedidos de conexão em tempo hábil, o que faz com que a fila de pedidos encha e que muitos deles sejam descartados. Desta forma, caso um usuário legítimo tente acessar o serviço que está sendo atacado, é muito provável que seu pedido de conexão seja descartado junto com o tráfego de ataque. Isso ocorre porque o tráfego do usuário legítimo precisa disputar o mesmo recurso com os inúmeros segmentos enviados pelo atacante. É importante ressaltar ainda que, para permanecer anônimo, o atacante não precisa usar seu endereço IP verdadeiro para realizar o ataque. Na verdade, qualquer endereço IP pode ser usado como endereço de origem nos pacotes de ataque. O uso de endereços de origem forjados não altera em nada o efeito sofrido pela vítima. No entanto, a resposta ao pedido de conexão não retorna para o atacante, mas sim para o endereço usado em seus pacotes. Desta forma, o atacante consegue não só negar o serviço da vítima, mas também permanecer anônimo.

Um outro recurso que pode ser explorado durante ataques por inundação de segmentos TCP SYN é a memória da vítima. Quando o servidor recebe um pedido de abertura de conexão TCP, ele aloca uma pequena quantidade de memória para armazenar determinadas informações sobre o estado da conexão, como os números de seqüência ini-

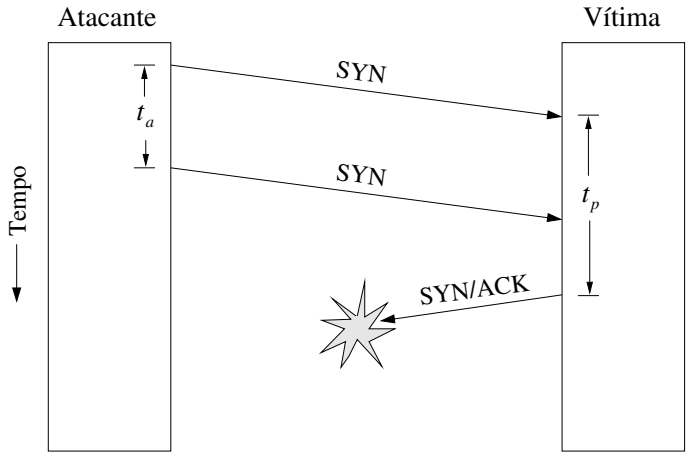

Figura 1.5. Ataque de negação de serviço consumindo o processamento da vítima.

ciais, os endereços IP e as portas TCP usadas nesta conexão. Em seguida, um segmento TCP SYN/ACK é enviado de volta para o cliente de forma a finalizar o acordo em três vias. Neste ponto, é dito que a conexão TCP se encontra semi-aberta. Os recursos reservados inicialmente ficam então alocados até que a conexão seja finalizada por meios convencionais, ou até que um temporizador estoure indicando que o esperado segmento TCP ACK não foi recebido. Neste caso, a memória alocada é finalmente liberada para ser aproveitada em conexões futuras. Em um ataque visando esgotar a memória, o atacante forja o endereço IP de origem dos pacotes, substituindo-o por algum endereço não utilizado. Estes pacotes são então enviados para a vítima de forma a iniciar simultaneamente diversas conexões semi-abertas, consumindo a sua memória. Para que este tipo de ataque tenha sucesso, é necessário que o atacante consiga gerar segmentos a uma taxa maior do que a vitima consegue liberar recursos para as novas conexões, ou seja, é preciso que $t_a < t_m$. A Figura 1.6 ilustra o caso em que o recurso atacado é a memória da vítima. Nesta situação, a mesma disputa entre o tráfego legítimo e o tráfego de ataque ocorre na vítima. À medida que uma conexão libera sua memória, um outro pedido de abertura de conexão é processado. Devido ao volume de tráfego de ataque enviado para a vítima, muito provavelmente este novo pedido processado não é legítimo e só serve para ocupar a memória da vítima por um determinado período. Quando a memória se esgota, novos pedidos de abertura de conexão não são atendidos. É importante observar que, mesmo com limites de memória controlados pelo sistema operacional, a negação de serviço ocorre quando a memória destinada para as conexões se esgota, sem ser necessário o esgotamento completo de toda a memória da vítima.

Nota-se, por definição, que a desigualdade $t_m > t_p$ é sempre válida. Desta forma, três possibilidades podem ocorrer. A primeira possibilidade ocorre quando $t_a < t_p < t_m$, ou seja, o intervalo entre o envio de pacotes de ataque é menor que o tempo de processamento da vítima. Este é o caso abordado anteriormente onde o processamento da vítima é sobrecarregado. Vale também ressaltar que, além do processamento, a memória da vítima também é sobrecarregada. Isto ocorre porque os pedidos de conexão que a vítima consegue atender a tempo, alocam um espaço de memória que só é liberado quando o temporizador estourar, uma vez que pacotes com endereços de origem forjados são usados. A segunda

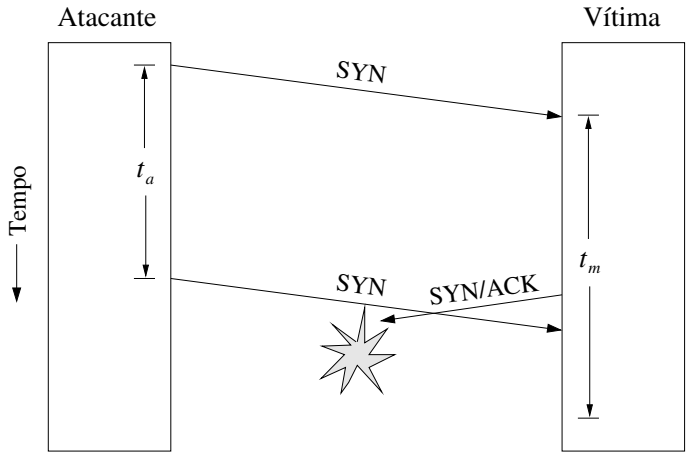

Figura 1.6. Ataque de negação de serviço consumindo a memória da vítima.

possibilidade ocorre quando $t_p \leq t_a < t_m$. Neste caso, a vítima consegue processar as requisições, mas espaços de memória vão sendo consumidos por cada nova conexão e não são liberados a tempo para atender a todas as conexões. Portanto, novas conexões não são aceitas por falta de memória. O último caso ocorre quando o atacante não consegue gerar pacotes de ataque a uma taxa suficiente nem para sobrecarregar a memória, ou seja, $t_a \geq t_m$. Neste caso, os recursos da vítima não são atacados diretamente e outras formas de ataques são necessárias para negar o serviço oferecido.

Uma solução proposta por Schuba *et al.* [Schuba et al., 1997] sugere que o servidor não armazene nenhum estado ao receber um segmento TCP SYN, de forma a evitar ataques por inundação que sobrecarreguem a memória. A idéia dos autores é que somente um segmento TCP SYN/ACK seja enviado como resposta e que o número de seqüência enviado pelo servidor seja o valor *hash* dos endereços IP de origem e destino, das portas TCP, do número de seqüência inicial do cliente e de um valor secreto armazenado no servidor. Ao receber o segmento TCP ACK, o servidor verifica se o seu número de seqüência está correto através do cálculo do valor *hash*. Caso esteja correto, a memória é enfim alocada para a conexão. Apesar de ser eficiente contra ataques que visam esgotar a memória, este mecanismo apresenta pequenas desvantagens. Um problema é a possibilidade de conexões serem estabelecidas somente com o segmento TCP ACK. Apesar de a probabilidade deste caso ocorrer seja pequena, tal evento não é impossível. Além disso, o protocolo TCP não consegue mais oferecer a tolerância a falhas para o caso de conexões semi-abertas, uma vez que não há mais registro destas conexões no servidor.

É importante ressaltar que o consumo de processamento e memória da vítima através de ataques de inundação não ocorre somente em ataques ao protocolo TCP. Na verdade, o conceito apresentado aqui é mais geral e pode ser aplicado a outros protocolos usados. O ataque ao processamento da vítima pode ser realizado praticamente a qualquer protocolo, uma vez que todas as mensagens recebidas precisam ser processadas. O ataque à memória, por outro lado, exige algum tipo de alocação de recursos de memória na vítima para que tenha sucesso.

Ataques por Refletor

Um outro tipo de ataque de negação de serviço conhecido é o ataque por refletor. Este ataque também é um ataque por inundação que visa consumir recursos da vítima. Porém, devido a presença de uma estação intermediária entre o atacante e a vítima, ele é aqui tratado como um ataque diferenciado. A idéia é usar a estação intermediária para refletir o tráfego de ataque em direção à vítima. Tal manobra dificulta ainda mais a descoberta da identidade dos atacantes, uma vez que o tráfego que chega à vítima é originado no refletor, e não no próprio atacante. Para a reflexão do tráfego de ataque, é necessário que o atacante envie algum tipo de requisição (REQ) para o refletor, usando como endereço de origem o endereço da vítima ao invés do seu próprio endereço. Ao receber uma requisição, o refletor não consegue verificar a autenticidade da origem dessa requisição e, conseqüentemente, envia uma resposta (RESP) diretamente para a vítima. A Figura 1.7 mostra este procedimento de maneira sucinta.

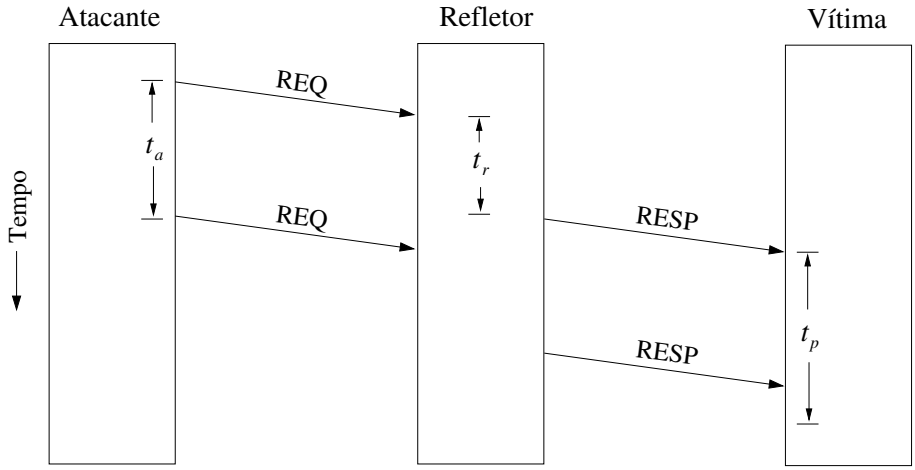

Figura 1.7. Ataque de negação de serviço por um refletor.

Para que o processamento da vítima seja sobrecarregado, é necessário que $t_a < t_p$. Entretanto, como o objetivo do ataque é usar os recursos do refletor e não inundá-lo, é preciso que o refletor consiga processar as requisições a tempo. Desta forma, o tempo de processamento do refletor t_r precisa ser menor ou igual ao intervalo entre pacotes de ataque, ou seja, $t_r \leq t_a$. Caso contrário, o processamento do refletor será sobrecarregado e o tráfego excedente será descartado, não apresentando efeito na vítima. Portanto, o ideal para o atacante é gerar os pacotes de ataque de acordo com a desigualdade $t_r \leq t_a < t_p$.

É importante ressaltar que este tipo de ataque não é restrito a um determinado protocolo. Para o seu funcionamento, é necessário somente um protocolo qualquer que envie um pacote de resposta ao atender a algum tipo de requisição. Desta forma, o próprio protocolo TCP abordado na seção anterior pode ser usado para tal finalidade. No caso, o atacante envia diversos segmentos TCP SYN para o refletor, que os responde com segmentos TCP SYN/ACK direcionados para a vítima. Outra possibilidade é o uso do protocolo UDP (*User Datagram Protocol*) associado ao serviço de DNS. Neste caso, o atacante pode enviar diversas requisições ao serviço de DNS do refletor, que envia uma resposta para a vítima.

Uma das vantagens deste tipo de ataque é que o próprio refletor pode contribuir para o consumo de recursos da vítima. Isso ocorre quando uma mensagem de requisição enviada pelo atacante é menor que a mensagem de resposta enviada pelo refletor. Neste caso, é dito que o refletor também atua como amplificador do tráfego de ataque. Um exemplo típico onde o tráfego enviado pelo atacante é amplificado é o ataque Smurf [CERT, 1998], ilustrado na Figura 1.8. Neste caso, o atacante envia pacotes ICMP para o endereço IP de difusão de uma determinada rede usando o endereço IP de origem da vítima. Como conseqüência, todos as estações daquela rede respondem à requisição, enviando uma resposta para a vítima. Neste caso, todas as estações da rede foram usadas como refletores apenas com o envio de um único pacote.

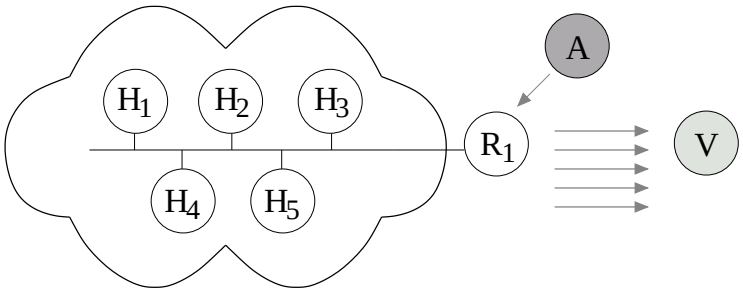

Figura 1.8. Ataque através da inundação ICMP: o atacante A envia pacotes `ping` para a rede refletora se passando pela vítima V para provocar uma inundação endereçada à vítima.

Outro exemplo de ataque por refletor inunda as vítimas com pacotes UDP. Neste caso, um atacante pode deixar fora do ar duas vítimas, fazendo com que as duas enviem continuamente pacotes uma para a outra. Para que tal fato ocorra, o atacante deve enviar um pacote aparentemente originado pelo serviço ICMP `echo` da vítima V_1 para o serviço de geração de caracteres (*chargen*) [Postel, 1983] da vítima V_2. Como mostra a Figura 1.9, ao receber o pacote ICMP `echo` com endereço de origem de V_1, a vítima V_2 envia uma resposta para a vítima V_1, que responde novamente para V_2 e dessa forma cria-se um *loop*. Conseqüentemente, o pacote de ataque nunca vai deixar a rede. Portanto, se um atacante envia um grande número de pacotes ICMP `echo` com o endereço IP de origem forjado, ele pode consumir a banda passante e sobrecarregar a vítima. O serviço *chargen* também pode ser utilizado para amplificar o tráfego de ataque. Apenas com o envio de um pequeno pacote para tal serviço, uma grande seqüência de caracteres é gerada e enviada como resposta. Desta forma, supondo um fator de amplificação de 50, o tráfego enviado pelo atacante a 128 kbps chega na vítima a 6,4 Mbps.

Ataques a Infra-estrutura de Rede

Ataques de negação de serviço muitas vezes são direcionados a sítios famosos na Internet. Desta forma, os seus autores conseguem uma fama maior como resultado do ataque. Nestes casos, é provável que a própria candidata à vítima seja superdimensionada, ou seja, possua recursos de processamento e de memória em abundância. Como conseqüência, ataques distribuídos de pequena escala não conseguem consumir tais recursos rápido o suficiente para que a vítima negue o seu serviço para usuários legítimos. Como uma alternativa, o atacante pode tentar concentrar seus esforços em algum ponto

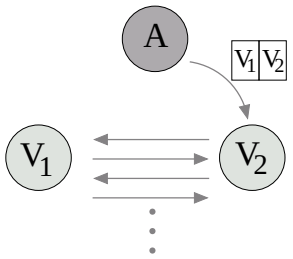

Figura 1.9. Ataque através da inundação UDP: o atacante A envia pacotes para a vítima V_2 se passando pela vítima V_1 para provocar uma troca infinita de mensagens entre as vítimas.

crucial para o funcionamento do serviço que não dependa da vítima. Um exemplo é tentar consumir toda a banda passante da vítima com o tráfego de ataque. Desta forma, por mais que a vítima consiga atender a todas as requisições que lhe chegam, muitas requisições são ainda perdidas na infra-estrutura de rede. Neste caso, o ponto onde as requisições são perdidas é algum roteador entre o atacante e a vítima onde existe um "gargalo," ou seja, onde o tráfego direcionado à vítima é maior do que o enlace de saída pode suportar. Desta forma, requisições legítimas ao serviço da vítima provavelmente são descartadas neste roteador, uma vez que precisam disputar o mesmo recurso com o tráfego enviado pelo atacante. Tais ataques são difíceis de serem combatidos, já que os pacotes não precisam ter nenhum padrão semelhante que possibilite filtrá-los.

Outro ataque que pode negar o serviço da vítima sem atacá-la diretamente é uma inundação aos seus servidores DNS. Estes servidores são responsáveis por traduzir nomes, usados pelas pessoas, em endereços IP, usados pelos computadores. Como geralmente as pessoas utilizam nomes para acessar um determinado servidor, um ataque ao serviço de resolução de nomes acaba por negar serviços de outros servidores. Um ataque deste tipo foi recentemente direcionado ao sítio da empresa Microsoft [Wagner, 2002]. Na época, seus servidores DNS estavam localizados em um mesmo segmento de rede e foi possível atacar a todos de uma vez através da inundação deste segmento. Como resultado, milhares de usuários ficaram sem acesso aos serviços providos pela empresa. Atualmente, seus servidores estão espalhados geograficamente e possuem enlaces redundantes de forma a minimizar os efeitos de um ataque semelhante.

Um outro alvo potencial para ser atacado são os próprios roteadores responsáveis por encaminhar os pacotes até a vítima. A função dos roteadores é simplesmente encaminhar cada pacote recebido, de acordo com o seu endereço de destino. Para isso, os roteadores difundem certas informações na rede de forma que cada roteador consiga construir uma tabela de roteamento, responsável por associar cada endereço de destino com uma interface de saída. Além da simples inundação, outros tipos de ataques podem ser direcionados a estes roteadores. Um atacante pode, por exemplo, encher a sua tabela de roteamento para dificultar a busca de endereços na tabela e atrasar muito o encaminhamento dos pacotes. Ou ainda, o roteador pode ser enganado e acabar encaminhando o tráfego legítimo para um local errado ao invés de entregá-lo para a vítima.

Ataques por Vulnerabilidade

Uma outra forma de negar os serviços providos pela vítima é deixá-la inoperante de alguma forma. Uma das maneiras de atingir este objetivo é explorar alguma vulnerabilidade na implementação da pilha de protocolos ou da própria aplicação da vítima. Um exemplo deste tipo de vulnerabilidade ocorreu, por exemplo, na implementação do protocolo TCP em sistemas operacionais Windows recentemente [Microsoft, 2002]. Para a sua ativação, o atacante precisava construir um segmento TCP com determinadas características e enviá-lo para a vítima. Ao receber este segmento, a vítima passava para um estado inesperado e o sistema operacional abortava, causando o congelamento total do seu processamento. O envio periódico destes segmentos poderia deixar a vítima inoperante por um bom tempo, dependendo da vontade do atacante.

Ataques de negação de serviço também já exploraram vulnerabilidades no próprio protocolo IP. Quando um determinado pacote é grande demais para ser transmitido sobre uma tecnologia de rede, o protocolo IP permite a quebra do pacote em fragmentos menores e o envio de cada fragmento separadamente. Para isso, cada fragmento possui um identificador, de forma que receptor consiga agregar os fragmentos de um mesmo pacote, e um número de seqüência, para determinar aonde no pacote original aquele fragmento se encontra. O receptor então recebe os fragmentos e os concatena de forma a remontar o pacote original. Assim, o ataque consistia no envio de diversos fragmentos IP pertencentes ao mesmo pacote com números de seqüência que se sobrepunham [CERT, 1997]. Desta forma, a vítima recebia uns poucos fragmentos intencionalmente mal formados e o protocolo entrava em um estado não previsto, resultando no seu congelamento ou reinicialização. Mesmo com a correção de tais vulnerabilidades, ataques usando fragmentos IP ainda são usados em ataques de negação de serviço que visam consumir os recursos da vítima. Tendo recebido um fragmento com um novo número identificador, a vítima armazena este fragmento por um determinado período até que todos os fragmentos restantes sejam recebidos ou até que um temporizador estoure. Além disso, o processamento de cada fragmento é necessário para determinar a sua posição dentro do pacote original. Desta forma, um atacante pode inundar a vítima com diversos fragmentos, cada um com um identificador diferente, de forma a esgotar a memória da vítima e aumentar o seu processamento. Dependendo da freqüência com que tais fragmentos são enviados para a vítima, a condição para a negação do seu serviço pode ser atingida.

Gont [Gont, 2004] descreve um novo tipo de ataque a conexões TCP já estabelecidas através do uso de somente um pacote ICMP. Uma das funções destes pacotes é relatar eventuais erros que ocorrem durante o roteamento de um pacote IP desde o emissor até o destinatário. O autor percebeu que é possível enviar um pacote ICMP forjado para uma das entidades que se comunicam, de forma a fingir que algum erro ocorreu durante a transmissão de um pacote. Desta forma, a conexão TCP é desfeita e uma nova conexão precisa ser estabelecida para o uso do serviço. O envio contínuo destes pacotes pode interromper o andamento de um serviço baseado no protocolo TCP. A identificação da origem de tais ataques que só utilizam um pacote para negar o serviço da vítima ainda é desafio para os sistemas de rastreamento atuais.

Ataques Distribuídos

Os ataques de negação de serviço distribuídos são geralmente usados em ataques por inundação, quando uma estação sozinha não é capaz de consumir algum recurso da vítima. Portanto, diversas estações precisam ser usadas para gerar o tráfego de ataque em direção à vítima e negar o seu serviço. Estas estações geralmente não pertencem ao atacante e são simplesmente computadores comprometidos por alguma falha de segurança. Uma vez tendo comprometido uma estação, o atacante apaga os rastros deixados pela invasão e instala um programa para comandá-la remotamente. Estes computadores ficam então sob o controle do atacante e, por isso, são chamados de agentes, escravos ou zumbis, denotando que são comandados por uma outra entidade.

Diferentes maneiras podem ser usadas pelo atacante para penetrar no sistema de outras estações. Geralmente, estas estações não possuem um sistema atualizado com as últimas versões dos programas usados. Como vulnerabilidades são encontradas freqüentemente, é possível encontrar uma série de estações com versões ultrapassadas que possuem vulnerabilidades de segurança. Algumas destas vulnerabilidades podem ser exploradas remotamente e liberar o acesso ao invasor. Com o acesso à estação invadida, é possível se aproveitar de outras vulnerabilidades locais para obter determinados privilégios de administrador e controlar totalmente à estação invadida. Outra possibilidade que vem sendo muito usada atualmente é obter a senha de uma determinada conta do sistema através de ataque de força bruta. O invasor tenta inúmeras possibilidades como senha para a conta em questão até conseguir penetrar na estação atacada.

Como os ataques de negação de serviço distribuídos podem ser compostos por centenas ou milhares de zumbis [Mirkovic et al., 2004], a invasão manual de cada zumbi individualmente se torna uma atividade cansativa para o atacante. Além disso, à medida que o número de zumbis aumenta, é muito difícil controlar cada zumbi sem nenhuma forma de automação. Por isso, algumas ferramentas foram criadas de forma a encontrar estações vulneráveis e invadí-las automaticamente. Outras ferramentas possibilitam a criação de redes hierárquicas para permitir o controle de um grande número de zumbis. Através dessas ferramentas, o atacante consegue comandar a todos os zumbis que iniciem um ataque a uma determinada vítima com um simples comando.

De forma a esconder a sua identidade, o atacante pode usar diversas estações intermediárias entre si mesmo e os zumbis. Estas estações são denominadas mestres e cada uma controla um determinado conjunto de zumbis. O atacante controla diretamente cada estação-mestre que, por sua vez, repassa os comandos do atacante para os zumbis. A Figura 1.10 ilustra uma rede composta por um atacante A, pelas estações-mestre M_1, M_2 e M_3 e pelos zumbis $Z_1, Z_2, Z_3, \ldots, Z_{12}$ para atacar uma vítima V. Durante a execução de um ataque, A comanda a M_1, M_2 e M_3 que iniciem um determinado ataque direcionado a V. As estações-mestre então repassam o comando para os zumbis sob seu controle e eles inundam a vítima. O ataque escolhido pelo atacante A e realizado por cada zumbi Z_i pode ser qualquer um daqueles descritos anteriormente, inclusive ataques por refletores. Além da vantagem de manter a identidade do atacante escondida, o uso destas redes hierárquicas é essencial para o controle de uma grande quantidade de zumbis.

Para se rastrear o atacante, é necessário primeiro identificar os zumbis para depois descobrir quem são os mestres e, por fim, chegar até o atacante. Logo, quanto mais cama-

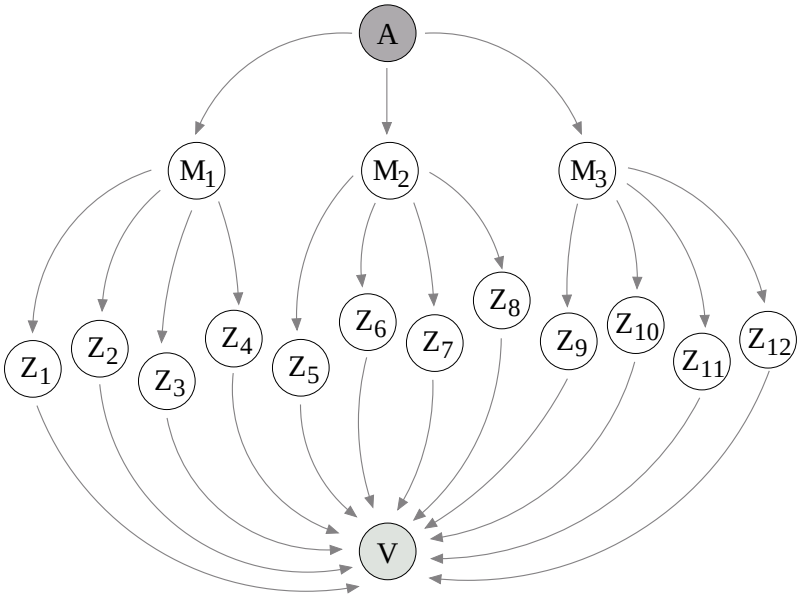

Figura 1.10. Ataque de negação de serviço distribuído.

das existem nesta hierarquia, mais protegido está o atacante. A Figura 1.11 ilustra duas técnicas utilizadas para o atacante comandar as estações-mestre. A primeira técnica usada por atacantes é entrar em diversas estações em seqüência antes de acessar os mestres. Inicialmente, o atacante *A* entra em uma estação intermediária I_1 e, por meio dela, o controle dos mestres é enfim realizado. Uma outra maneira de realizar este controle indireto é usar uma sala de bate-papo através do protocolo IRC (*Internet Relay Chat*), ou seja, um canal IRC. Neste caso, os mestres ou os próprios zumbis entram automaticamente em uma sala de bate-papo de um servidor escolhido pelo atacante e ficam esperando ordens. O atacante então entra na sala e envia os comandos para que o ataque seja iniciado [Gibson, 2001].

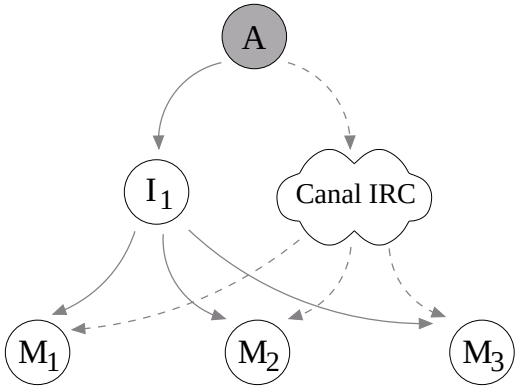

Figura 1.11. Estação intermediária ou um canal IRC para controlar as estações-mestre.

Diversas vantagens para o atacante surgem com a distribuição de ataques de negação de serviço. A primeira vantagem é conseguir deixar vítimas superdimensionadas inoperantes. Por possuírem recursos em abundância, estas vítimas são imunes a ataques de negação de serviço partindo de um único atacante. Entretanto, quando diversas estações são usadas para a geração de tráfego de ataque, a vítima pode ser seriamente afetada.

Um fato importante é que não importa o quão superdimensionada é uma determinada vítima, ela sempre pode ter o seu serviço negado desde que um número suficientemente grande de zumbis seja reunido para o ataque. Caso a vítima tente amenizar o efeito de um ataque distribuído através do aumento de seus recursos, o atacante pode simplesmente usar mais zumbis e obter o mesmo resultado.

Para interromper um ataque distribuído em andamento, a identificação de somente um zumbi não é suficiente. Como o tráfego agregado de diversos zumbis está inundando a vítima, é provável que, ao interromper um único zumbi, o efeito na vítima seja desprezível. Neste caso, é preciso identificar grande parte dos zumbis que estão atacando a vítima e tentar filtrar este tráfego de ataque o mais perto das fontes possível. Entretanto, só a identificação dos zumbis já é uma tarefa difícil que pode precisar da colaboração de diversos provedores de acesso. Além disso, é preciso contatar o responsável ou o administrador do zumbi para que a inundação originada por ele termine. À medida que o número de zumbis cresce, não é possível realizar esta tarefa para cada zumbi e é mais difícil interromper o ataque.

Como solução alternativa, é possível ainda tentar interromper a atividade de uma estação-mestre diretamente. Porém, é preciso antes identificar tais estações para depois tentar tomar alguma ação para interromper o ataque. Estas duas tarefas podem não ser simples. A identificação de uma estação-mestre exige primeiro a identificação de pelo menos um zumbi e a partir deste zumbi, tentar descobrir quem o está comandando remotamente. Além disso, as ferramentas de controle destas estações podem exigir algum tipo de autenticação e bloquear o acesso de quem não é autorizado. Desta forma, não é possível parar o ataque sem a intervenção do próprio administrador do zumbi.

1.3.5. Contramedidas aos Ataques de Negação de Serviço

Existem algumas medidas capazes de evitar ou, pelo menos, reduzir os danos causados por ataques de negação de serviço. Estas técnicas podem ser divididas em duas categorias: medidas preventivas e medidas reativas. As medidas preventivas buscam evitar ou reduzir a chance de que um ataque de negação de serviço ocorra. As medidas reativas, por outro lado, tentam lidar com ataques em andamento ou já encerrados. Para serem efetivas, os dois tipos de técnicas devem ser empregadas em conjunto. É importante utilizar medidas preventivas para reduzir o número de ataques; no entanto, é impossível garantir que não ocorrerão ataques. Para que as medidas preventivas sejam totalmente eficazes, é necessária a sua adoção em larga escala, possivelmente de forma cooperativa entre diversas administrações. Para estes ataques que escapam às medidas preventivas adotadas, resta a utilização de medidas reativas.

Medidas Preventivas

A medida preventiva mais imediata é a manutenção de todos os softwares dos sistemas finais devidamente atualizados. Este procedimento, além de minimizar os ataques por vulnerabilidades como os discutidos na Seção 1.3.4, também dificulta o domínio de possíveis zumbis. Fica claro, com isto, que a segurança dos usuários começa com os próprios usuários. Os responsáveis pelos sistemas finais não devem simplesmente espe-

rar que a infra-estrutura de rede forneça a segurança de seus sistemas, mas devem ser participantes ativos deste processo.

Uma técnica preventiva, denominada filtragem de ingresso (*ingress filtering*), foi sugerida para evitar que pacotes com endereços IP de origem forjados trafeguem na rede [Ferguson e Senie, 2000]. Implementada em alguns roteadores, a filtragem de ingresso descarta pacotes cujos endereços de origem não pertencem a determinados prefixos legitimamente anunciados. Em outras palavras, um roteador encarregado da agregação de rotas anunciadas por diversas redes em seu domínio deve impedir a saída de pacotes cujo endereço de origem não pertence a nenhuma dessas redes. A Figura 1.12 ilustra este conceito. Na figura, o atacante *A* está querendo atacar a vítima *V* usando pacotes com endereços de origem forjados, de forma a manter o anonimato. Seus pacotes seguem até a vítima *V* através dos roteadores R_1, R_2 e R_3. Caso a filtragem de ingresso esteja implementada em R_1, somente aqueles pacotes cujo endereço de origem pertence a faixa de endereços da Rede 1 são roteados. Pacotes com endereços de origem forjados são automaticamente impedidos de sair da rede e descartados por R_1. Desta forma, somente pacotes cujo endereço de origem estão dentro da faixa de endereços da Rede 1 podem ser usados para o ataque. Neste caso, a vítima *V* consegue descobrir facilmente a origem dos pacotes a partir de uma rápida análise no tráfego de ataque recebido. Roteadores podem ainda agregar faixas de endereços de origem para diminuir o processamento realizado por pacote. No exemplo, supondo que as faixas de endereços das Redes 1, 2, 3 e 4 podem ser agregadas em somente uma faixa de endereços, R_3 pode evitar a saída de pacotes forjados verificando somente se o endereço do pacote está dentro da faixa agregada.

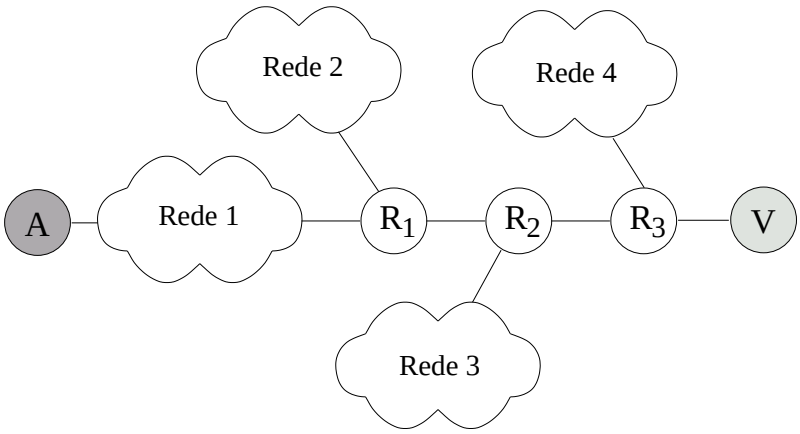

Figura 1.12. Filtragem de ingresso.

Apesar de simples, esta técnica apresenta desvantagens. Há uma indesejável dependência entre a segurança do destinatário final e as políticas adotadas nos roteadores ao longo do caminho. Na verdade, não existem incentivos para que domínios adotem a filtragem de ingresso como parte da política de segurança, uma vez que somente estações fora do próprio domínio são beneficiadas. Além disso, a filtragem de ingresso precisa ser implementada em uma escala global para ter efeito. Caso a implantação seja restrita a poucos roteadores, somente ataques passando por esses roteadores serão controlados. Uma outra desvantagem é processamento adicional inserido durante o roteamento. Como a filtragem influencia diretamente no processo de roteamento, o exame do endereço de

origem de cada pacote pode requisitar certos recursos que nem todo roteador disponibiliza. Existem ainda tecnologias que podem ser afetadas pela filtragem de ingresso, como o IP Móvel [Perkins, 2002], pois utilizam legitimamente pacotes com endereços de origem forjados.

Uma generalização da técnica de filtragem de ingresso foi proposta por Li *et al.* [Li et al., 2002]. Os autores propõem um protocolo que fornece aos roteadores algumas informações que possibilitam validar o endereço de origem dos pacotes. A idéia básica do protocolo é que mensagens contendo informações sobre endereços de origem válidos sejam propagadas a partir da própria rede de origem para todos os roteadores da rede. Desta forma, cada roteador consegue construir uma tabela de entrada que associa cada interface a um conjunto de endereços de origem válidos. Ao receber um pacote por uma de suas interfaces, o roteador verifica se o endereço de origem do pacote está dentro do conjunto de endereços associados àquela interface. Caso o endereço esteja dentro do conjunto, o pacote é roteado normalmente e, em caso contrário, ele é descartado. O protocolo proposto é semelhante a um protocolo de roteamento, onde as informações sobre os endereços de destinos são propagadas de forma que os roteadores possam construir uma tabela associando endereços de destino com interfaces de saída, a chamada tabela de roteamento. A diferença é que protocolos de roteamento visam determinar por qual interface um pacote será encaminhado enquanto que o protocolo de validação de endereço de origem verifica se o pacote foi recebido pela interface adequada.

A desvantagem principal do protocolo de validação proposto é o processamento adicional necessário durante o roteamento de cada pacote. Caso este protocolo venha a ser implementado, roteadores terão que fazer consultas a cada uma das tabelas. Uma consulta na tabela de entrada é necessária para determinar se o endereço de origem é válido e outra consulta na tabela de roteamento é necessária para determinar a interface de saída e o próximo nó a receber o pacote. Além disso, o protocolo proposto também apresenta os problemas de um protocolo de roteamento convencional, como a necessidade de atualizações periódicas, a reação às mudanças topológicas da rede, a garantia da integridade e da autenticidade das mensagens trocadas e o uso eficiente da banda. Como objetivo final, ele deve ainda garantir que pacotes com endereços de origem forjados sejam descartados e que nenhum pacote com endereço legítimo o seja. Tudo isso torna o protocolo complexo.

Medidas Reativas

Nos casos em que o ataque não pode ser evitado, não resta alternativa se não tentar fazê-lo parar e buscar localizar o atacante para que sejam tomadas medidas reparativas. Para isto, o primeiro passo é necessariamente obter informações sobre a origem do ataque.

O rastreamento de pacotes surge como uma forma de identificar a rota percorrida por um determinado pacote até o seu verdadeiro emissor. No caso de ataques de negação de serviço, a identificação dos atacantes é importante tanto para interromper o ataque através de filtragem como para a adoção de medidas judiciais contra o próprio atacante. Entretanto, o rastreamento é ainda mais geral e pode ser útil em qualquer ocasião onde é necessário determinar a origem ou a rota percorrida por um determinado pacote.

A importância do rastreamento é ainda maior quando se leva em consideração que,

devido à arquitetura da Internet, não se pode confiar no endereço de origem de nenhum pacote como forma de identificação de sua verdadeira origem. É impossível apenas analisando o pacote determinar se o endereço é verdadeiro ou foi forjado. Desta forma, para ser confiável, toda informação sobre a origem de um ataque deve ser obtida através do rastreamento, pois se deve sempre levar em conta a possibilidade de endereços forjados.

1.4. O Rastreamento de Pacotes IP

O rastreamento de pacotes tem como finalidade identificar a rota percorrida por um determinado pacote e o seu verdadeiro emissor. No caso de ataques de negação de serviço, é fundamental identificar a origem do ataque, mesmo que estes não empreguem endereços de origem forjados, de forma a filtrar o tráfego de ataque na sua origem. Em ataques por refletor e através de zumbis, o rastreamento deve ser realizado para se identificar o refletor ou zumbi e a partir destes, um novo rastreamento deve ser realizado para se chegar aos verdadeiros atacantes.

As técnicas de rastreamento de pacotes IP podem ser divididas em duas categorias: sistemas de rastreamento sem estado e sistemas de rastreamento baseados em auditorias [Laufer et al., 2005b]. Os sistemas de rastreamento sem estados não armazenam nenhum tipo de informação sobre a origem dos pacotes nem na infra-estrutura da rede, nem nas estações finais. Neste caso, toda a atividade de rastreamento se baseia no estado da rede no momento em que o rastreamento é feito. Por outro lado, os sistemas baseados em auditoria armazenam de alguma forma, durante o processo de envio dos dados, informações que podem ser úteis para reconstruir o caminho real percorrido pelos pacotes. A Seção 1.4.1 discute possíveis soluções de rastreamento sem o armazenamento de informações e a Seção 1.4.2 trata das soluções que se baseiam na auditoria de dados armazenados durante o processo de envio dos pacotes.

1.4.1. Sistemas de Rastreamento Sem Estado

As soluções de rastreamento sem estados possuem a vantagem de não exigir uma estrutura capaz de armazenar informações sobre os pacotes enviados em tempo real. A busca por uma técnica que não requeira armazenagem de dados é significativa, pois o número de pacotes comutados na rede é enorme e, como conseqüência, o volume de dados a ser armazenado é grande. Também existe o aspecto legal que não permite que se armazene conteúdo de pacotes que circulam na rede. No entanto, sem o auxílio de dados de auditoria, a única opção para determinar a rota seguida pelos pacotes do ataque é testar os enlaces (interfaces) da rede para determinar se o tráfego de ataque passa ou não por esse enlace. Desta forma, estas soluções só podem ser utilizadas na detecção de ataques por inundação e enquanto os ataques ainda estão em andamento, sendo incapazes de realizar o rastreamento *post mortem*.

A solução atualmente empregada para o rastreamento de ataques é mais precária possível, pois se serve de procedimentos que requerem a intervenção humana. Assim, quando ocorre um ataque, a vítima entra em contato com o administrador da rede do seu provedor de serviço (*Internet Service Provider* - ISP) e solicita a determinação da rota de ataque. A vítima ou o administrador do ISP procura uma característica (assinatura) dos pacotes usados no ataque ou faz-se uma análise do tráfego no roteador associado à vítima.

É feito um teste no roteador mais próximo da vítima para determinar por qual enlace (interface de rede) o tráfego do ataque chega a esse último roteador. Uma vez determinado o enlace correto, o procedimento é repetido no roteador localizado na outra extremidade do enlace. Depois, salto-a-salto, este processo é repetido até que seja determinado o roteador de onde parte o ataque. Pode ser necessário testar até d roteadores, onde d é o diâmetro da rede. Em alguns casos, é necessária a interação entre provedores de serviço, o que torna o processo lento.

Em situações nas quais é possível manipular o roteamento da rede através da inclusão de algumas rotas estáticas, uma opção é fazer com que todo o tráfego destinado à vítima seja direcionado a um roteador previamente determinado. Se esse roteador for configurado de forma a estar no centro da rede, pode-se iniciar um teste de enlaces salto-a-salto partindo do centro da rede. Desta forma, o teste de enlaces é feito em no máximo $d/2$ roteadores. Se a análise estiver sendo feita numa rede similar a um ISP, na qual parte-se do princípio de que os ataques só podem vir de alguém conectado a um dos roteadores de borda, existem duas outras opções para o teste de enlaces. A primeira, caso a rede disponha de recursos para a construção de topologias lógicas sobre a topologia física, é a construção de uma rede lógica de rastreamento em sobreposição (*overlay*) ligada a todos os roteadores de borda da rede em questão. Com isso, é possível realizar o teste de enlaces salto-a-salto partindo da vítima até o roteador de borda por onde o tráfego de ataque entra na rede através da rede lógica de rastreamento. Neste caso, o número máximo de roteadores testados é função do diâmetro da rede lógica. A outra opção, caso deseje-se apenas determinar por qual roteador o tráfego de ataque entra na rede, é analisar estatísticas de tráfego nos roteadores de borda da rede. Assim, baseado na assinatura do ataque, é possível determinar qual roteador de borda é o responsável pela entrada do tráfego de ataque na rede.

As duas técnicas de rastreamento sem estados apresentadas nesta seção são, na verdade, formas de automatizar e otimizar o teste dos enlaces. A primeira técnica, chamada *CenterTrack* [Stone, 2000], tem o escopo limitado à rede de um único ISP e busca determinar por onde o tráfego de ataque entra na rede deste ISP. Neste sistema, o rastreamento é otimizado através da utilização de uma rede lógica de rastreamento em sobreposição, na qual são realizados os testes de enlace salto-a-salto. A segunda técnica é conhecida como “inundação controlada” [Burch e Cheswick, 2000] e tem como objetivo automatizar os testes de enlace salto-a-salto numa rede composta por diferentes ISPs e, portanto, sem uma administração central.

Para que a técnica *CenterTrack* possa ser utilizada, é necessária a construção de uma rede lógica de rastreamento na rede do ISP em questão. Desta forma, esta técnica aumenta a complexidade da rede e acarreta um aumento nas tarefas administrativas do ISP. Cuidados especiais devem ser tomados, por exemplo, para evitar que os túneis utilizados na construção da topologia lógica da rede de rastreamento interfiram na operação dos protocolos de roteamento que estejam sendo utilizados no interior da rede. No entanto, esta técnica tem como vantagem requerer a presença de ferramentas de diagnóstico apenas nos nós de borda e nos nós que compõem o *backbone* da rede de rastreamento.

O *backbone* da rede de rastreamento pode ter diversas arquiteturas, cada uma adequada a certas características da rede física do ISP [Stone, 2000], mas deve estar ligada

a todos os roteadores de borda da rede. Desta forma, este *backbone* liga todas possíveis entradas do tráfego de ataque a todas as possíveis vítimas. Uma vez descoberto o ataque, são criadas rotas estáticas apontando para a vítima que farão com que após a convergência do protocolo de roteamento todo o tráfego direcionado à vítima passe pela rede de rastreamento.

Após a reconstrução das rotas através da rede lógica de rastreamento, pode-se realizar testes de enlace salto-a-salto, partindo da vítima, através desta rede lógica. Embora estes testes não identifiquem com exatidão *todos* os roteadores da rota, ele resultará na descoberta do roteador por onde o tráfego entra na rede. Caso deseje-se filtrar o tráfego de ataque, esta filtragem deve ser realizada neste roteador de entrada.

Esta técnica apresenta algumas limitações. Uma destas limitações que merece destaque é que o sistema não ajuda no rastreamento de ataques originados no *backbone* do ISP, pois este roteador pode não fazer parte da rede de rastreamento. Outro problema surge quando a vítima é um dos roteadores do *backbone* da rede do ISP. Neste caso, o procedimento de direcionar o tráfego destinado à vítima através da rede de rastreamento pode resultar em *loops* e colapso de túneis. A utilização de túneis gera, ainda, mais dois problemas. O primeiro deles é a sobrecarga introduzida pelo uso de túneis. Desta forma, se o ataque for realizado com vários pacotes pequenos, o uso de túneis podem aumentar o tamanho do pacote significativamente. Em última instância, os túneis servirão como um amplificador do ataque. Além disso, se os túneis não forem autenticados de alguma forma, um ataque mais inteligente pode atrapalhar a operação do sistema forjando pacotes de modo a que estes pacotes sejam injetados diretamente nos túneis. Uma questão que também merece atenção é que este sistema redireciona o tráfego endereçado para a vítima. Esta característica possibilita ao atacante descobrir através de ferramentas como o *traceroute* que alguma coisa está errada e que ele está possivelmente sendo rastreado. Sabendo disso, o atacante pode parar o ataque antes que o processo de identificação esteja completo. Deve-se destacar, por fim, que esta técnica pode ser facilmente adaptada para se tornar uma técnica baseada em auditoria. Para isto, é necessário que equipamentos capazes de "farejar" a rede sejam incluídos no *backbone* da rede de rastreamento.

A técnica da inundação controlada busca automatizar o processo de teste de enlaces em redes compostas por diferentes ISPs. Esta técnica necessita que algumas ferramentas estejam disponíveis em todos ou, pelo menos, na maior parte dos roteadores da rede considerada. A principal vantagem deste sistema é possibilitar o rastreamento sem a intervenção dos ISPs envolvidos. Na realidade, esta técnica pode até mesmo ser empregada sem que a administração do ISP tome conhecimento.

Este sistema se baseia no teste de enlace salto-a-salto na rede inteira. Para isso, é necessário que o responsável pelo rastreamento tenha conhecimento da topologia da rede como um todo. Isto pode ser obtido através de *traceroutes* da vítima para as outras redes. Este processo, além de custoso do ponto de vista de tempo e recursos, é complicado caso existam enlaces assimétricos. Embora se possa assumir que a maior parte das rotas são simétricas, pode ser necessária a utilização de outros meios para obter uma visão suficientemente detalhada da rede.

Um ponto interessante da inundação controlada é a forma com que o teste de enlace é realizado. Ao invés de analisar o tráfego que passa por um determinado en-

lace e comparar com uma assinatura conhecida do ataque, na inundação controlada o teste de enlace é feito através da inundação do enlace em questão. Após esta inundação, determina-se se o enlace pertence ou não à rota de ataque observando o fluxo do ataque que chega à vítima. Se o fluxo do ataque diminui após a inundação do enlace, assume-se que o enlace faz parte da rota de ataque e a diminuição do fluxo na vítima se deu por descarte de parte dos pacotes de ataque devido ao congestionamento criado no enlace testado. Pode-se dizer que neste sistema o teste de enlace é realizado de forma indireta. Duas questões importantes surgem com este tipo de teste de enlace. A primeira é a influência do comportamento do fluxo do ataque sobre a identificação dos enlaces pertencentes à rota de ataque. Uma vez que a identificação depende da observação de variações do fluxo na vítima, se o ataque for originalmente feito com um fluxo imprevisível de pacotes, o teste de enlaces se torna muito difícil de ser realizado com alguma confiança. Além disso, é necessário que o responsável pelo rastreamento seja capaz de efetivamente inundar o enlace testado, o que se torna mais difícil à medida que a capacidade do enlace aumenta. É importante destacar que essa inundação deve ser idealmente contida ao enlace em questão para não mascarar resultados nem congestionar outras partes da rede. Stone [Stone, 2000] sugere o uso do serviço de geração de caracteres (*chargen*). Esta ferramenta gera um fluxo contínuo de dados partindo do roteador em direção a quem se conecta ao serviço. A inundação pode ser restrita ao enlace em questão se o responsável pelo rastreamento utilizar o serviço forjando o endereço de origem do pedido para fazer parecer que quem está se conectando é o roteador na outra extremidade do enlace testado. Os testes realizados demonstram que na maioria dos casos esta ferramenta é capaz de inundar o enlace, especialmente considerando-se que esta inundação é facilitada pelo tráfego legítimo que circula pela rede. Os resultados mostram que na maioria das vezes é possível identificar o segmento da rede de onde parte o ataque. Entretanto, em alguns casos, o teste de enlace pode apresentar resultados que ajudam pouco na determinação da rota de ataque.

Um aspecto importante da inundação controlada é a sua implicação ética. Inundar um enlace para testar se ele faz parte da rota de um ataque de negação de serviço é, de certa forma, realizar um ataque de negação de serviço ao ataque original. Este segundo ataque, ao inundar a rede, pode prejudicar outros usuários. Surge então a pergunta se o rastreamento não pode causar mais dano do que o próprio ataque. Espera-se que se os testes de enlace são feitos na escala temporal correta (apenas poucos segundos), a maior parte dos usuários sequer percebem mudanças na rede. Por fim, há a discussão sobre a utilização de serviços disponíveis nos roteadores, como o *chargen*. O fato de um ISP deixar um serviço funcionando em um roteador não significa que este ISP concorda que qualquer um se utilize deste serviço.

1.4.2. Sistemas de Rastreamento Baseados em Auditoria

O rastreamento de pacotes baseado em auditoria consiste basicamente na coleta de informações sobre os pacotes que circulam na rede a fim de viabilizar a reconstrução do caminho percorrido por pacotes provindos de atacantes. Este tipo de sistema pode ser classificado segundo o local onde as informações coletadas são armazenadas. Assim, as informações podem ser armazenadas nas estações finais, na infra-estrutura de rede ou ainda em ambas.

Armazenamento nas Estações Finais

O mecanismo mais simples de rastreamento de pacotes baseado em auditoria é a inclusão do endereço IP de cada roteador pelo qual o pacote passa. Este é um esquema similar à opção de Record Route do IP [Postel, 1981], na qual todo pacote possui o caminho inteiro percorrido na rede, da fonte ao destino. Além da simplicidade, este esquema permite o rastreamento do caminho de ataque com apenas um pacote, o que seria ideal em ataques distribuídos. No entanto, ele apresenta algumas desvantagens que o torna praticamente inviável [Savage et al., 2001]. A adição de dados ao pacote durante o roteamento implica um acréscimo significativo de processamento. Outro problema é o aumento do tamanho do pacote a cada roteador, podendo acarretar em fragmentações desnecessárias que podem sobrecarregar os roteadores.

Uma característica bastante comum nos sistemas com armazenamento nas estações finais é a coleta de informações através da marcação de pacotes. A marcação de pacotes consiste na utilização de campos no interior do pacote que possibilitem registrar uma determinada característica. Para efeito de rastreamento, são visadas as informações sobre o caminho percorrido pelo pacote do nó origem até o nó destino. Para não se alterar o tamanho original do pacote é comum usar campos pouco usados do cabeçalho IP para transportar estas informações. A marcação pode ser determinística, quando todos os pacotes são marcados, ou probabilística, quando os pacotes são marcados aleatoriamente segundo uma determinada probabilidade.

A marcação de pacotes probabilística (*Probabilistic Packet Marking* - PPM) foi introduzida originalmente por Savage *et al.* [Savage et al., 2001]. Nesta abordagem, cada roteador probabilisticamente insere informações sobre si mesmo no cabeçalho IP do pacote. Após o recebimento de uma certa quantidade de pacotes, a vítima é capaz de reconstruir a rota de ataque. A Figura 1.13 mostra a idéia básica do mecanismo de marcação de pacotes.

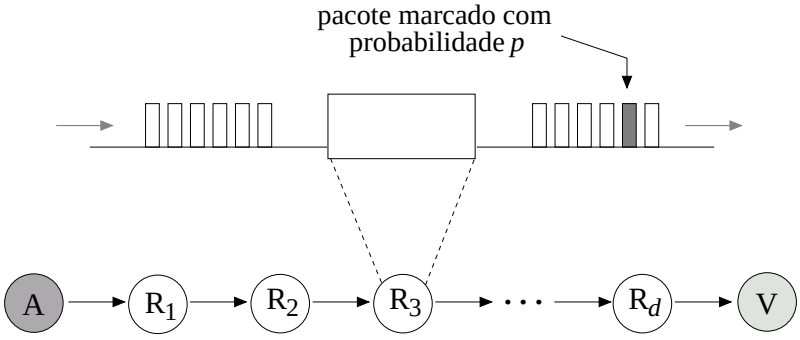

Figura 1.13. A marcação probabilística de pacotes.

O atacante A envia pacotes em direção da vítima V. Estes pacotes seguirão o caminho indicado, passando pelos roteadores $R_1, R_2, R_3, \ldots, R_d$. Cada roteador do caminho irá acrescentar ao cabeçalho do pacote o seu endereço IP, com uma certa probabilidade p. A vítima V deve armazenar um número suficiente de pacotes de modo a garantir a existência de pelo menos um pacote com o endereço de cada roteador do caminho. Deste modo,

após o recebimento dos pacotes, a vítima é capaz de reconstruir o caminho até o atacante. A ordem dos roteadores na rota reconstruída é deduzida a partir da quantidade de pacotes marcados por cada roteador. A probabilidade de um pacote chegar marcado por um determinado roteador é inversamente proporcional à sua distância para a vítima. Isto porque quanto mais próximo da vítima, menor é a probabilidade de um outro roteador sobrescrever o endereço já marcado. Assim, quanto maior é a quantidade de pacotes marcados por um determinado roteador, menor é a sua distância para a vítima. A probabilidade α_i de um pacote chegar marcado por um determinado roteador R_i é expressa por

$$\alpha_i = p(1-p)^{d-i}, \tag{1}$$

onde p é a probabilidade de um pacote ser marcado por R_i, d é o número de roteadores na rota de ataque e $(d-i)$ é a distância do roteador R_i à vítima.

Um dos problemas da marcação de pacote é a quantidade de pacotes que a vítima deve coletar. Por exemplo, se a distância do atacante for igual a 15 ($d = 15$) e $p = 0,51$ a vítima deve armazenar em média 42 mil pacotes para receber pelo menos um pacote do último roteador. Sendo que, para obter uma garantia de 95%, o número de pacotes armazenados passa para aproximadamente 300 mil pacotes. Por isso, todos os mecanismos baseados na marcação de pacotes partem da premissa que os ataques de negação de serviço são caracterizados pelo envio de milhares de pacotes [Savage et al., 2001]. Esta grande quantidade de pacotes pode tornar a reconstrução da rota um procedimento lento. Outro problema é a falta de robustez em relação a múltiplos atacantes. Neste caso, haverá mais de um roteador com a mesma distância dificultando a definição da rota de ataque. A única alternativa possível seria o conhecimento da topologia para distinguir entre os possíveis caminhos. Por fim, o atacante pode ainda tentar forjar um sufixo da rota ao acrescentar falsos endereços aos pacotes de ataque. A probabilidade α_0 de um pacote chegar a vítima com o endereço forjado pelo atacante é

$$\alpha_0 = (1-p)^d. \tag{2}$$

Existe um compromisso entre a capacidade do atacante de forjar endereços e a quantidade de pacotes necessários para a reconstrução da rota. Quanto maior é o valor de p, menor é a probabilidade da vítima receber um pacote marcado somente pelo atacante, conforme a Equação 2. Por outro lado, quanto menor é o valor de p, menos pacotes recebidos são necessários para o rastreamento. Existe um valor ótimo para p que minimiza a quantidade de pacotes a serem coletados. Um valor abaixo ou acima deste valor implica um aumento da quantidade de pacotes.

Apesar das desvantagens, a marcação de pacotes apresenta algumas vantagens interessantes. A marcação de pacotes não exige uma cooperação entre os ISPs. Diferentemente dos mecanismos baseados em inundação, o tráfego adicional acrescentado pelo esquema de marcação de pacotes não é significativo. Além disso, um mecanismo baseado na marcação de pacotes permite a descoberta da rota de ataque mesmo após o seu término. Por fim, a marcação de pacotes não representa uma grande sobrecarga para os roteadores.

Este é o princípio básico do mecanismo de rastreamento baseado na marcação de pacotes. Muitos trabalhos propuseram pequenas adaptações a fim de minimizar as desvantagens deste novo paradigma. Savage *et al.* propõem a amostragem de enlaces (arestas)

no lugar de amostragem de nós (vértices). Uma aresta pode representar um enlace entre dois roteadores conectados diretamente, ou através de outros roteadores que não implementam o mecanismo. Assim, cada pacote é marcado não apenas por um determinado nó, mas por dois nós consecutivos. Esta pequena modificação no esquema de marcação resolve o problema dos múltiplos atacantes, além de facilitar a implementação progressiva. Para tanto, são utilizados dois campos no cabeçalho do pacote onde serão armazenados os endereços de começo e fim da aresta e um campo para computar a distância, como mostra a Figura 1.14.

Figura 1.14. Espaço necessário nos mecanismos de amostragem de arestas.

Quando um nó decide marcar um determinado pacote, ele deve acrescentar seu endereço no campo *endereço 1* e em seguida zerar o campo *distância*. Caso contrário, o nó deve checar o campo *distância*. Sempre que o valor deste campo for igual à zero, o nó deve inserir seu endereço ao campo *endereço 2* e incrementar o valor do campo *distância*. Para qualquer outro valor, o roteador deve apenas incrementar o valor do campo *distância*. Ao final da coleta dos pacotes de ataque, a vítima terá informações sobre diferentes arestas, com diversas distâncias da vítima, a partir das quais é possível construir um grafo, onde a vítima é a raiz e os roteadores mais próximos dos atacantes são as folhas. O número médio X de pacotes necessários para a reconstrução da rota pode ser calculado segundo a expressão do problema do coletador de cupons. Nesse problema, é calculado o número médio de cupons que devem ser tirados de uma urna de forma a se ter pelo menos um cupom de cada tipo. Assim, o número médio X de pacotes necessários é expresso por

$$E(X) \leq \frac{\ln(d)}{p(1-p)^{d-1}}, \tag{3}$$

onde d representa a distância entre o atacante e a vítima e p a probabilidade de marcação. Este valor é referente a uma rota de ataque. No caso da amostragem de arestas, mesmo que o atacante consiga forjar um endereço, o efeito será o acréscimo de uma aresta ao final da rota. Isto significa que o atacante é incapaz de forjar arestas entre a vítima e o último roteador da rota de ataque.

O fato de utilizar um espaço adicional no campo do cabeçalho IP cria um problema de incompatibilidade com as versões anteriores. No mecanismo de amostragem de arestas, é necessário um espaço de 72 bits (Figura 1.14). Para solucionar este problema, Savage *et al.* propõem uma adaptação ao mecanismo de amostragem de arestas com o objetivo de diminuir o tamanho do espaço necessário no cabeçalho IP, de maneira a utilizar um espaço já existente e raramente utilizado. Assim, é proposta a utilização do campo de identificação de fragmentação (*IP identification*), que contém 16 bits. A novidade está na codificação da identificação das arestas, que antes era feito através dos endereços de começo e fim das arestas. Nesta nova versão, os endereços são intercalados pelo seu próprio *hash* e divididos em k fragmentos, como mostra a Figura 1.15. Assim, ao marcar um

pacote, ao invés de acrescentar seu endereço, o roteador deve acrescentar apenas um desses fragmentos, escolhido aleatoriamente, e escrever em um outro espaço do cabeçalho o *identificador de fragmento* (*ID frag*), além de zerar o campo *distância*. O nó seguinte, caso não marque o pacote, após verificar que o campo *distância* está em zero, deve escrever no campo *fragmento* o resultado da operação OU exclusivo (XOR) entre o *n*-ésimo fragmento de seu endereço com o fragmento já escrito, onde *n* é o número indicado no campo *identificador de fragmento* (Figura 1.16).

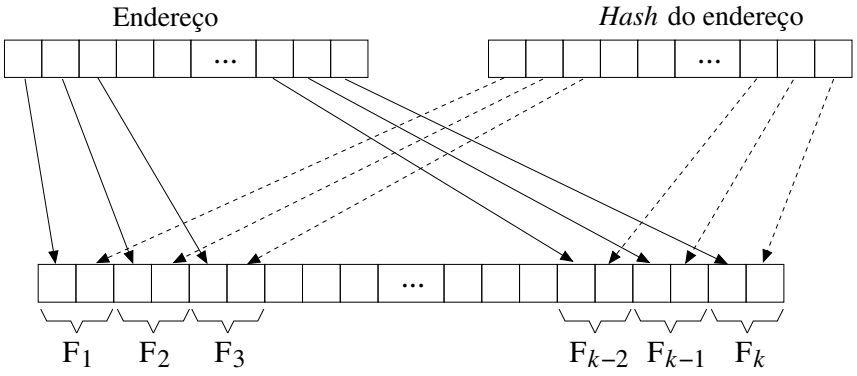

Figura 1.15. Geração dos fragmentos.

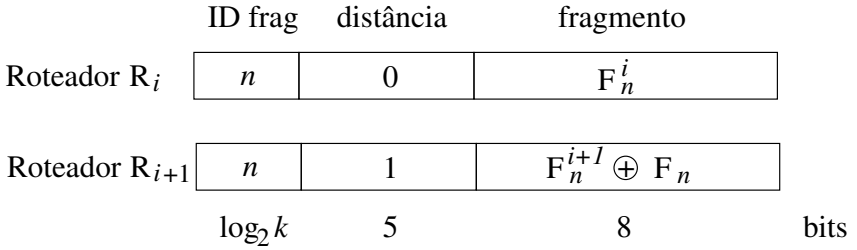

Figura 1.16. Marcação do pacote na amostragem de arestas, usando fragmentos.

Apesar de permitir a utilização de apenas 16 bits e manter a compatibilidade com as versões anteriores, este novo esquema aumenta o número de pacotes necessários para a reconstrução da rota de ataque, pois é necessário coletar pelo menos uma amostra de todos os fragmentos de cada aresta. Isto multiplica por *k* o número de pacotes necessários. Além disso, em um ataque distribuído com *m* atacantes, mesmo de posse de pelo menos uma amostra de cada um dos *k* fragmentos de cada aresta, no final, a vítima terá *m* exemplares diferentes com a mesma distância e o mesmo identificador de fragmento, provindas de *m* roteadores diferentes. Assim, para remontar todos os fragmentos de um mesmo roteador e obter seu endereço (intercalado pelo *hash*), a vítima deve testar todas as combinações possíveis e checar o *hash* de cada uma. Dependendo do número de atacantes, este procedimento pode implicar um processamento significativo. Mesmo tendo realizado todo este procedimento, ainda existe a possibilidade de obter endereços que, apesar de possuir um *hash* válido, não representam enlaces existentes. Estes enlaces são chamados de falsos positivos.

Pode-se mostrar que a proposta de amostragem de fragmentos sugerida por Savage *et al.* requer uma grande carga computacional e produz um significativo número de falsos positivos no processo de reconstrução da rota de ataque, na presença de múltiplos atacantes [Song e Perrig, 2001]. Por exemplo, com somente 25 atacantes, esta abordagem pode levar dias para computar a rota de ataque, apresentando milhares de falsos positivos. Assim, Song e Perrig [Song e Perrig, 2001] propõem um novo mecanismo de rastreamento de pacotes baseado na amostragem de arestas, que apresenta uma complexidade e número de falsos positivo inferior à proposição de Savage *et al*. A principal diferença entre as duas propostas está na codificação das arestas. Ao invés de acrescentar um fragmento do endereço IP intercalado com o *hash* do endereço, é acrescentado o valor do *hash* do endereço IP. Para isto, é definida a utilização de duas funções *hash* h e h', ambas de 11 bits. Cinco bits são utilizados para o campo *distância*, totalizando 16 bits. Desta forma, antes de encaminhar um pacote ao próximo nó, o roteador R_i deve decidir se irá marcar o pacote, com uma probabilidade p. Sempre que um pacote é marcado, o roteador insere o valor $h(E_i)$ no campo *aresta*, onde E_i é o seu endereço, e zera o campo *distância*. Caso o pacote não seja marcado, o roteador R_i deve checar o campo *distância*. Se o valor for igual à zero, o roteador deve atualizar o conteúdo do campo *aresta* com o resultado da operação OU exclusivo (XOR) entre o conteúdo do campo *aresta* e o valor $h'(E_i)$, e incrementar o campo *distância*. Para qualquer outro valor, apenas o campo *distância* é incrementado, como mostra a Figura 1.17.

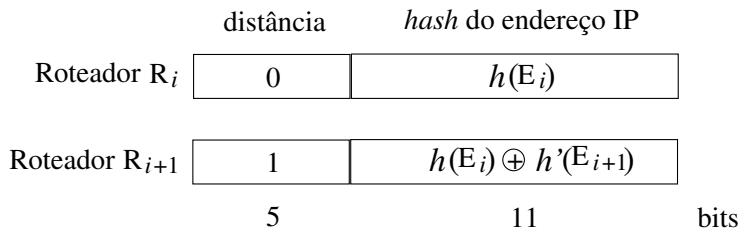

Figura 1.17. Marcação do cabeçalho segundo Song e Perrig.

A reconstrução é bastante simples. Primeiramente, considera-se que a vítima contém um mapa dos roteadores da Internet, representado por um grafo cuja raiz é a própria vítima. Assim, após a coleta de pacotes, a vítima deve comparar o valor $h(F_v)$ com o campo *aresta* de todos os pacotes cujo campo *distância* tem o valor zero, onde F_v contém todos os filhos diretos da vítima. Todos os endereços cujo resultado da comparação é positivo são acrescentados a um conjunto denominado S_0. O conjunto S_i contém todos os endereços IP cujo resultado da comparação é positiva, onde i representa a distância para a vítima. Após a identificação dos roteadores de distância zero, a vítima realiza a operação de OU exclusivo (XOR) entre o valor $h'(S_0)$ e o campo *aresta* de todos os pacotes cujo campo *distância* é igual a um, e comparar o resultado com $h(F_0)$, onde F_0 são todos os filhos do conjunto S_0. Neste novo esquema, um falso positivo pode ocorrer quando o resultado da função *hash* de dois nós irmãos, filhos do mesmo roteador, tem o mesmo valor. A probabilidade deste evento ocorrer é 2^{-11}. No caso da existência de muitos vizinhos a probabilidade da ocorrência de falso positivo aumenta significativamente. Para resolver este problema, é proposta uma pequena alteração no procedimento de marcação. Ao invés

de utilizar apenas duas funções *hash* independentes, são utilizado dois conjuntos g e g' de funções *hash*, todas independentes. Ao marcar um pacote, o roteador em questão deve escolher uma função *hash* do grupo g a ser utilizada. Após fazer a marcação e zerar o campo *distância*, ele deve inserir o valor sorteado no campo *identificador de função hash (ID hash)*. O roteador seguinte, que fará a operação OU exclusivo (XOR) para delimitar a aresta, deve utilizar a função *hash* com o mesmo ID *hash*, mas do grupo g'. Durante a reconstrução, uma determinada aresta só é considerada como verdadeira se possuir um número mínimo de marcações, cada uma com um valor *hash* diferente. Desta forma, os falsos positivos diminuem. Song e Perrig ainda propõem um mecanismo de marcação de pacotes com autenticação, baseado no esquema anterior, a fim de evitar que roteadores comprometidos possam forjar arestas e dificultar ou até mesmo impedir que a rota de ataque seja construída.

Neste sistema de marcação de pacotes, pode-se calcular a capacidade do atacante de forjar uma aresta [Park e Lee, 2001]. Para isto, basta que o atacante faça uma marcação e o pacote não seja marcado por nenhum roteador do caminho. A probabilidade de um pacote chegar na vítima sem ter sido marcado por nenhum roteador é dada pela Equação 2. Para que a vítima receba mais pacotes marcados pelo atacante que pacotes marcados pelo primeiro roteador do caminho, basta que:

$$\begin{aligned} \alpha_0 &\geq \alpha_1 \\ (1-p)^d &\geq p(1-p)^{d-1} \\ 1-p &\geq p \\ p &\leq 0,5. \end{aligned} \tag{4}$$

Para que a vítima receba mais pacotes marcados pelo atacante que por todos os outros roteadores, basta que:

$$\begin{aligned} \alpha_0 &\geq \sum_{i=1}^{d} \alpha_i \\ (1-p)^d &\geq \sum_{i=1}^{d} p(1-p)^{d-i} \\ (1-p)^d &\geq 1-(1-p)^d \\ p &\leq 1-2^{-1/d}. \end{aligned} \tag{5}$$

Por exemplo, para $d = 10$ tem-se $p \leq 0,067$. Isto mostra que o atacante pode forjar m caminhos diferentes com a mesma probabilidade que o caminho original, como mostra a Figura 1.18.

Se o valor de m for grande, ou seja, se o atacante for capaz de forjar muitos caminhos, a vítima terá problemas para identificar o verdadeiro caminho. No entanto, o valor de m é limitado por $(d-1)$ [Park e Lee, 2001]. Também é mostrado que um aumento no valor de p diminui o valor de m. Desta maneira, Park e Lee demonstram que mecanismos baseados em PPM podem ser eficientes para ataques de apenas um atacante, mas são vulneráveis a ataques distribuídos com múltiplos atacantes. Isto ocorre, pois a maneira de maximizar o número de caminhos forjados m é forçar a diminuição do valor

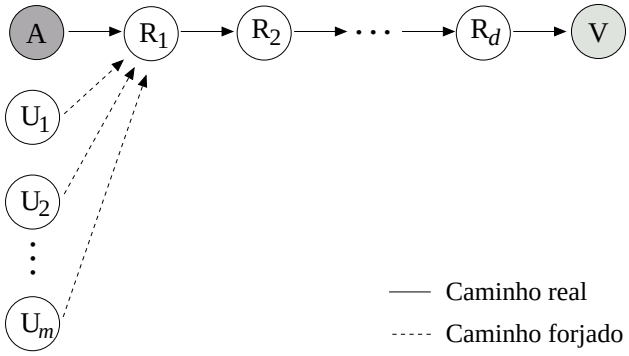

Figura 1.18. Caminhos forjados pelo atacante.

de *p*. A única possibilidade seria diminuir o número de pacotes *N* enviados pelo atacante. Isto porque *N* deve ser suficiente para que a vítima possa receber pelo menos um pacote do roteador R_1. Entretanto, o valor de *N* não é pequeno em ataques de negação de serviço por inundação. Assim, no caso de apenas um atacante, é mostrado que com *N* entre 10^3 e 10^5 pacotes e $d = 25$, o valor de *m* fica limitado entre dois e cinco. Porém, em um ataque com múltiplos atacantes, onde cada atacante possa reduzir significativamente o número de pacotes enviados, o valor de *m* tenderá à $(d-1)$, que no caso de $d = 25$ é uma quantidade significativa de caminhos possíveis para cada atacante.

Com objetivo de diminuir a complexidade do algoritmo de reconstrução de rota e de aumentar a confiabilidade a fim de permitir o rastreamento de um ataque distribuído com um grande número de atacantes, um novo esquema de rastreamento foi proposto [Goodrich, 2002]. Nesta nova proposta, foi introduzido o conceito de mensagem (M_R) que será enviada pelo roteador *R* à vítima. O conteúdo da mensagem depende do seu tamanho. Mas a princípio, a mensagem mais simples contém o endereço IP do roteador em questão, sendo possível acrescentar outras informações, como por exemplo, o endereço do próximo roteador e até mesmo informações para autenticação. O processo de marcação de pacote é basicamente o mesmo, com a diferença que não existe mais o campo *distância*. Isto significa que quando o pacote não é marcado, não há sobrecarga para o roteador. A codificação da mensagem no cabeçalho IP é feita utilizando 25 bits. Primeiramente, a mensagem é preenchida, se necessário, para que seu tamanho seja múltiplo de *l*. Em seguida é gerada uma soma de verificação (*checksum*) de *c* bits da mensagem ($C(M_R)$). Depois a mensagem é divida em *l* partes iguais, denominadas de palavras, w_0, $w_1, \ldots, w_{l-1}$. Os 25 bits do cabeçalho são divididos, como mostra a Figura 1.19.

índice	$C(M_R)$	palavra (w_i)	
$\log_2 l$	c	$25 - c - \log_2 l$	bits

Figura 1.19. Marcação do cabeçalho segundo Goodrich.

Ao marcar um pacote, o roteador deve escrever uma das palavras da mensagem no campo *palavra*, o valor de $C(M_R)$, no campo *checksum* e o índice da palavra no campo

índice. A reconstrução é bastante similar a de Savage *et al*. A diferença está na remontagem da mensagem. O *checksum* além de ser o responsável por garantir se a mensagem foi corretamente montada, serve também para identificar todas as palavras da mesma mensagem. Deste modo, basta que a vítima separe todos os pacotes que possuam o mesmo *checksum*, eliminar as repetições e testar todas as combinações possíveis com as palavras de mesmo índice e mesmo *checksum*. Goodrich demonstra que seu mecanismo apresenta uma complexidade bastante inferior às propostas de Savage *et al.* e Song e Perrig. Outro fator importante é o fato de conseguir rastrear ataques com até 1000 atacantes. Por último, o esquema de mensagem é flexível, permitindo acrescentar informações para autenticação dos roteadores.

Após diversos trabalhos na área de marcação de pacotes, percebeu-se que o grande problema dos mecanismos baseados na PPM é o fato dos atacantes poderem forjar uma marcação. Assim, acabar com a possibilidade do atacante forjar uma marcação, resolveria o principal inconveniente deste tipo de abordagem. Por isso, foi proposto um novo mecanismo de marcação de pacotes determinístico que utiliza 17 bits (16 do campo *Packet ID* e um do campo reservado *flag*) do cabeçalho IP [Belenky e Ansari, 2003b]. O mecanismo pressupõe que a marcação seja feita apenas pelos roteadores de borda. Desta maneira, todo pacote que entrar por uma interface de "borda" será marcado com o endereço IP da respectiva interface. Como são usados apenas 17 bits, o endereço da interface é divido em duas partes. O roteador deve escrever no pacote uma das duas partes com igual probabilidade. Em seguida deve-se acrescentar zero ou um ao campo *flag*, de acordo com a parte do endereço escrita no campo *Packet ID*. Assim, após a recepção de alguns pacotes, a vítima obterá as duas partes do endereço da interface de entrada do roteador mais próximo do atacante. Com apenas 7 pacotes a vítima tem 99% de chance de obter o endereço completo. Um dos problemas desta abordagem é a fragmentação de pacotes, pois todos os fragmentos de um determinado pacote devem conter o mesmo valor para o campo *Packet ID* para serem corretamente remontados. Isto pode não acontecer, dado que dois valores distintos podem ser escritos no campo *Packet ID*. Este problema foi resolvido identificando os fragmentos e escrevendo a mesma parte do endereço da interface de entrada no campo *Packet ID* [Belenky e Ansari, 2003a]. Entretanto, esta abordagem possui dois problemas graves que podem impedir que a vítima descubra o endereço da interface de entrada do roteador de borda. O primeiro problema aparece quando o atacante envia apenas um pacote com cada endereço IP. O segundo caso seria um ataque distribuído onde m atacantes enviam pacotes com os mesmos endereços IP. Assim, para cada endereço IP diferente a vítima receberia m pacotes para cada parte do endereço.

Ainda na tentativa de impedir o atacante de forjar as marcações, foi desenvolvido um outro mecanismo de rastreamento baseado na marcação de pacotes, denominado *Dynamic Probabilistic Packet Marking* (DPPM) [Liu et al., 2003]. Seu objetivo é minimizar a capacidade do atacante de forjar caminhos, que, como foi mostrado por Park e Lee [Park e Lee, 2001], está diretamente relacionada com o valor de p e com o número N de pacotes enviados pelo atacante. Como este último fator é determinado pelo atacante, a única opção é mudar o valor de p. Desta forma, a proposta consiste em atribuir diferentes probabilidades de marcação de acordo com a distância que o pacote já percorreu. Assim, para um caminho com d roteadores, o pacote será marcado pelo roteador R_i com a probabilidade $p_i = 1/i$. A probabilidade de um pacote chegar marcado pelo roteador R_i na

vítima é dado pela seguinte expressão

$$\alpha_i = p_i \prod_{j=i+1}^{d} 1 - p_j. \tag{6}$$

A Equação 1 é um caso particular da Equação 6, onde $p_i = p$, para $1 \leq i \leq d$. Expandindo a Equação 6 e substituindo $p_i = 1/i$, tem-se a seguinte expressão:

$$\begin{aligned}
\alpha_i &= p_i \prod_{j=i+1}^{d} 1 - p_j \\
&= p_i(1-p_{i+1})(1-p_{i+2})\ldots(1-p_d) \\
&= \left(\frac{1}{i}\right)\left(1-\frac{1}{i+1}\right)\left(1-\frac{1}{i+2}\right)\ldots\left(1-\frac{1}{d}\right) \\
&= \frac{1}{d}.
\end{aligned} \tag{7}$$

Isto significa que a probabilidade da vítima receber um pacote marcado pelo roteador R_i é a mesma para todos os roteadores, não importando a distância para a vítima. Este resultado é bastante interessante, pois a probabilidade do pacote chegar marcado pelo atacante α_0, ou seja, a probabilidade do atacante forjar uma marcação é igual à zero. Intuitivamente, pode-se ver este resultado a partir da seguinte expressão:

$$\alpha_0 = 1 - \sum_{i=1}^{d} \alpha_i = 0 \tag{8}$$

Além disso, o valor mínimo de pacotes necessários para a reconstrução da rota é igual à d, pois $N \cdot \min(\alpha_i) \geq 1$. Na PPM, $\min(\alpha_i) = \alpha_1$, mas no DPPM todos os α_i são iguais à $1/d$. Logo,

$$\begin{aligned}
N \cdot \frac{1}{d} &\geq 1 \\
N &\geq d.
\end{aligned} \tag{9}$$

O grande problema desta abordagem é saber quantos nós o pacote já percorreu. Uma das possíveis soluções é a utilização do campo TTL (*Time To Live*) do cabeçalho IP para a contagem dos saltos de distância do atacante. No entanto, o atacante pode facilmente forjar o TTL para tentar burlar o mecanismo de rastreamento de pacote. Por isso, os autores sugerem a utilização de um valor máximo de TTL inicial (TTL_{max}) para um pacote, de maneira que se um roteador recebe um pacote com $TTL > TTL_{max}$, ele deve sobrescrever o valor do TTL com TTL_{max}. O valor sugerido para o TTL máximo é de 32 ou 64, considerando-se que a maioria das rotas da Internet possuem menos que 25 roteadores [Liu et al., 2003]. Neste caso, se o atacante escolhe um valor menor para o campo TTL, como $TTL = TTL_{max} - z$, a probabilidade p_i de um roteador marcar um determinado pacote será

$$p_i = \frac{1}{z+i}. \tag{10}$$

Substituindo este novo valor na Equação 6, obtém-se a probabilidade do pacote chegar a vítima marcado pelo roteador R_i,

$$\alpha_i = \frac{1}{z+d}. \tag{11}$$

Assim, a probabilidade de chegar à vítima um pacote marcado pelo atacante pode ser escrito da seguinte maneira:

$$\begin{aligned} \alpha_0 &= 1 - \sum_{i=1}^{d} \alpha_i \\ &= 1 - \frac{d}{z+d} \\ &= \frac{z}{z+d}. \end{aligned} \tag{12}$$

Esta não aparenta ser uma boa solução. Por exemplo, no caso de $TTL_{max} = 64$ e o atacante ter marcado um TTL = 28 ($z = 36$), α_0 será igual a $0,36$. De acordo com os próprios autores, este valor de TTL ainda é suficiente para alcançar a grande maioria dos sítios da Internet. O atacante poderia ainda diminuir o valor do TTL, de acordo com a distância da vítima, piorando ainda mais a eficiência do mecanismo de rastreamento.

Além da preocupação com a capacidade do atacante de forjar arestas, existe ainda um grande interesse na otimização da codificação das arestas, para tornar o rastreamento baseando na marcação de pacotes um sistema mais eficiente. Um mecanismo denominado SNITCH [Aljifri et al., 2003] é uma extensão a PPM na qual é modificada a codificação da identificação das arestas. Neste trabalho é proposta a utilização de uma técnica de compressão do cabeçalho IP, definida na RFC 2507 [M.Degermark et al., 1999] e na RFC 1144 [Jacobson, 1990]. O objetivo desta técnica de compressão do cabeçalho IP é diminuir o tamanho do pacote para aumentar a vazão de enlaces de baixa velocidade. Esta técnica pressupõe a existência de campos no cabeçalho IP que não mudam em uma seqüência de pacotes do mesmo fluxo. Estes campos são denominados de contexto do pacote. Assim, apenas o primeiro pacote enviado conterá todos os campos do cabeçalho que será associado a um contexto. Os pacotes seguintes, que possuam o mesmo contexto, terão estes campos removidos do cabeçalho e armazenarão o identificador do contexto ao qual estão associados (*Context ID* – CID). O espaço economizado com esta técnica é igual a 144 bits. Na proposta de Aljifri *et al.* [Aljifri et al., 2003], estes 144 bits são utilizados para a marcação dos pacotes. Desta maneira, ao decidir marcar um pacote, o roteador associa o contexto deste pacote a um CID. Em seguida, o roteador compara o contexto do primeiro pacote com o pacote seguinte. Caso os contextos sejam idênticos, o roteador comprime este cabeçalho e acrescenta as informações relativas ao mecanismo de rastreamento. A vítima, ao receber este pacote, retira as informações de rastreamento antes de descomprimir o cabeçalho. Após coletar o número suficiente de pacotes, as duplicatas são eliminadas e a rota pode ser construída a partir das arestas e das distâncias. Apesar de conseguir aumentar a quantidade de dados escritos no cabeçalho, evitando a fragmentação das informações de rastreamento, a proposta apresenta o mesmo problema da alta probabilidade do atacante forjar caminhos.

Uma alternativa para a codificação das informações de rastreamento é apresentada por Dean *et al.* [Dean et al., 2002]. A codificação é baseada em equações algébricas. As-

sim, cada roteador indica sua presença na rota de ataque, acrescentando o resultado de um polinômio no qual a variável é o seu endereço IP. A vítima deve ter conhecimento deste polinômio. Após a recepção de um número suficiente de pacotes, a reconstrução da rota de ataque é feita a partir da resolução de um sistema de equações. No entanto, a proposta também é baseada na marcação de pacotes e, por conseqüência, possui as mesmas desvantagens em relação à capacidade do atacante forjar marcações. Além disso, diferente do sistema proposto por Savage *et al.*, este sistema não apresenta nenhum código de detecção de erro para reduzir a probabilidade de se construir um sistema de equações utilizando equações provenientes de rotas diferentes. Conseqüentemente, ainda mais falsos positivos são esperados para ataques distribuídos de pequeno porte. Um outro tipo de equações algébricas foi sugerido na tentativa de diminuir a quantidade de bits usados para a codificação [Bai et al., 2004].

Existe ainda outro método baseado em auditoria, que não se baseia na marcação de pacotes [Bellovin et al., 2003]. Ao rotear um pacote, cada roteador probabilisticamente envia para a vítima um pacote ICMP com informações sobre si mesmo e sobre seus roteadores adjacentes. Para um fluxo suficientemente longo, a vítima usa estes dados recebidos para reconstruir a rota de ataque. Entretanto, como as informações de auditoria são enviadas em pacotes separados, a autenticação das mensagens é necessária de forma a evitar mensagens forjadas pelo atacante. Logo, a adoção de uma infra-estrutura de chave pública se torna inevitável. Uma extensão desta idéia inclui novos conceitos como "utilidade" e "valor" das mensagens de rastreamento [Mankin et al., 2001], a fim melhorar o sistema.

Armazenamento na Infra-estrutura de Rede

Sob a perspectiva de armazenamento na própria infra-estrutura de rede, a maneira mais simples de recolher rastros de auditoria é cada roteador registrar todos os pacotes que o atravessam [Stone, 2000]. Porém, recursos excessivos são requisitados tanto para armazenagem quanto para a mineração dos dados. Além disso, a invasão de um roteador acarretaria ainda em problemas de privacidade, uma vez que ele contém informações sobre todos os pacotes roteados.

Uma alternativa para reduzir a armazenagem de um grande volume de informações é utilizar um Filtro de Bloom [Bloom, 1970]. Ultimamente, estes filtros têm sido amplamente usados em redes de computadores [Broder e Mitzenmacher, 2003].

O Filtro de Bloom [Bloom, 1970] é uma estrutura de dados usada para representar de forma compacta um conjunto $S = \{s_1, s_2, \ldots, s_n\}$ de n elementos. Ele é constituído por um vetor de m bits e por k funções *hash* independentes $h_1, h_2, \ldots, h_k$ cujas saídas variam uniformemente no espaço discreto $\{0, 1, \ldots, m-1\}$. O vetor de bits é obtido da seguinte forma. Inicialmente, todos os seus bits encontram-se zerados. Para cada elemento $s_i \in S$, os bits do vetor correspondentes às posições $h_1(s_i), h_2(s_i), \ldots, h_k(s_i)$ são preenchidos com 1. O mesmo bit pode ser preenchido diversas vezes sem restrições. A Figura 1.20 ilustra a inserção de um elemento no filtro de maneira sucinta. Uma vez que o Filtro de Bloom é uma forma compacta de representar um conjunto de elementos, testes de pertinência podem ser realizados visando determinar se um elemento x pertence ou não ao conjunto S. Para isso, verifica-se se os bits do vetor correspondentes às posições

$h_1(x), h_2(x), \ldots, h_k(x)$ estão preenchidos com 1. Se pelo menos um bit estiver zerado, então com certeza $x \notin S$. Por outro lado, se todos os bits estão preenchidos, então assume-se que $x \in S$. Na verdade, um elemento externo $x \notin S$ pode ser reconhecido como um autêntico elemento do conjunto, criando um falso positivo. Tal anomalia ocorre quando todos os bits $h_1(x), h_2(x), \ldots, h_k(x)$ são preenchidos por um subconjunto dos elementos de S.

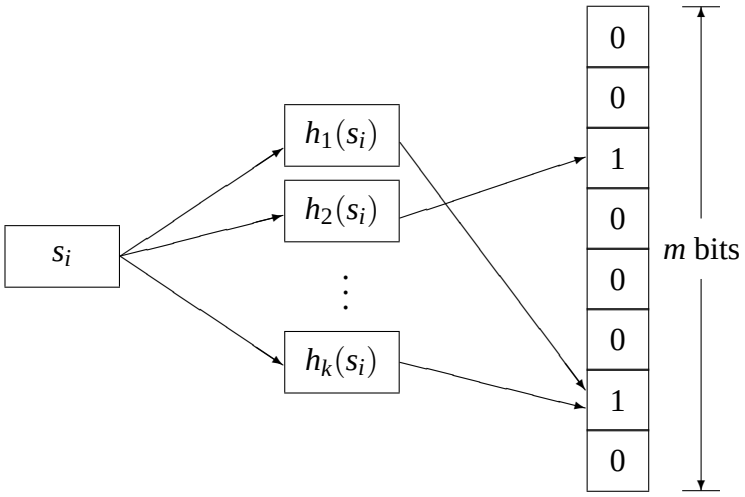

Figura 1.20. Inserção de um elemento em um Filtro de Bloom.

A probabilidade de se encontrar um falso positivo para um elemento $x \notin S$ é calculada de maneira trivial. Dado que as funções *hash* usadas são uniformes e independentes, a probabilidade p de um determinado bit permanecer em zero mesmo depois de inseridos os n elementos é

$$p = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{-kn/m}. \tag{13}$$

Como o mesmo cálculo se aplica para todos os bits do vetor, na média, uma fração $e^{-kn/m}$ dos bits permanece zerada após as inserções [Mitzenmacher, 2002]. Desta forma, a fração média de bits preenchidos com 1 depois de n inserções é $\left(1 - e^{-kn/m}\right)$. A probabilidade de falso positivo f é então a probabilidade de se encontrar um bit em 1 para cada uma das k posições indicadas, ou seja

$$f = \left(1 - e^{-kn/m}\right)^k. \tag{14}$$

A utilização do Filtro de Bloom no sistema de rastreamento permitiu a criação de um mecanismo que possui a vantagem de rastrear um único pacote IP que tenha passado na rede sem a necessidade de se armazenar todo o tráfego roteado [Snoeren et al., 2002]. Para isso, são usados Filtros de Bloom em dispositivos acoplados aos roteadores que armazenam os pacotes roteados de forma compacta. Periodicamente, os filtros saturados são armazenados para futuras requisições e trocados por novos filtros vazios. Para mais tarde determinar se um pacote passou pelo roteador, o seu filtro simplesmente é verificado. Um processo repetitivo pode ser feito por cada roteador para reconstruir o caminho do

pacote até a sua verdadeira origem. Porém, mesmo com o uso de Filtros de Bloom, tal sistema exige uma alta capacidade de armazenamento.

Um outro mecanismo baseado no armazenamento na infra-estrutura de rede consiste em cada roteador amostrar uma pequena porcentagem do tráfego e armazenar os sumários dos pacotes amostrados [Li et al., 2004]. O compromisso da amostragem é a dificuldade inserida no processo de rastreamento. Para minimizar o número de pacotes necessários para rastrear a origem de um ataque com acurácia, é necessário aumentar o fator de correlação entre roteadores vizinhos. Este fator representa a porcentagem dos pacotes amostrados por um roteador que também foram amostrados pelo roteador seguinte. Além disso, os autores introduzem um arcabouço que utiliza a teoria da informação para responder algumas perguntas sobre a escolha de parâmetros do sistema e sobre o compromisso entre os recursos necessários e a acurácia do rastreamento.

O método de amostragem independente não apresenta um bom funcionamento já que o fator de correlação dos pacotes amostrados por roteadores vizinhos é somente p, o mesmo valor da taxa de amostragem. Para aumentar este fator, um outro método é abordado pelos autores. A idéia principal deste método é que, além de amostrar o pacote, o roteador também marca o pacote amostrado de modo que o próximo roteador possa coordenar a sua marcação com a do roteador anterior. O campo necessário para a marcação é de apenas 1 bit e pode se localizar, por exemplo, no campo *reserved flag*, não utilizado do cabeçalho IP.

O processo de rastreamento se inicia com a vítima verificando seus roteadores vizinhos. Para cada roteador S, a vítima pede a verificação da passagem dos pacotes do conjunto L_v (conjunto de pacotes de ataque selecionados pela vítima) nos seus Filtros de Bloom correspondentes. Se pelo menos um pacote é reconhecido, o roteador S monta um conjunto de pacotes L_S a partir dos pacotes em L_v que foram encontrados em S. Agora, cada roteador R vizinho de S será verificado com L_S. Sendo R "condenado", todo o conjunto L_v é novamente repassado para o roteador para a formação de um conjunto L_R. Este processo é repetido recursivamente para cada roteador. Através da elaboração de um arcabouço de teoria da informação, os autores chegam a conclusão de que a utilização de $k = 12$ (k é o número de funções *hash* usadas no Filtro de Bloom) é um valor ótimo para o desempenho do método de rastreamento. As melhorias propostas por Li *et al.* [Li et al., 2004] diminuem drasticamente o espaço necessário para o armazenamento embora a capacidade de se rastrear um único pacote seja comprometida.

Baseado ainda na técnica de utilização de funções *hash* para o armazenamento das informações de rastreamento nos roteadores da rede [Snoeren et al., 2001], foi proposto um novo mecanismo que armazena as informações referentes aos fluxos a fim de reduzir a quantidade de informações armazenadas [Lee et al., 2004]. Uma outra melhoria foi proposta para melhorar a acurácia do grafo reconstruído [Hilgenstieler e Duarte Jr., 2004].

Armazenamento Híbrido

Algumas propostas visam conciliar as vantagens do armazenamento nas estações finais com as do armazenamento na infra-estrutura de redes, a fim de minimizar os problemas intrínsecos a cada uma das abordagens. Com este objetivo, é proposto o armaze-

namento híbrido, que consiste em armazenar parte das informações na vítima e parte nos roteadores da rede.

Dentro deste contexto, foi proposto um novo mecanismo que combina a marcação de pacotes com o armazenamento de informações nos roteadores [Choi e Dai, 2004]. O esquema consiste em acrescentar ao cabeçalho o enlace pelo qual o pacote entrou. Para isto, é reservada uma parte do cabeçalho IP de 32 bits para armazenar as informações. O primeiro bit é definido como o campo *save flag* que indica se as informações de rastreamento referentes àquele pacote foram armazenadas em algum roteador. Os 31 bits restantes são utilizados para guardar os enlaces. Este campo é sempre iniciado com o primeiro bit em 1 e os outros em zero, pois o último bit em 1 indica o fim do último enlace acrescentado, como apresentado na Figura 1.21.

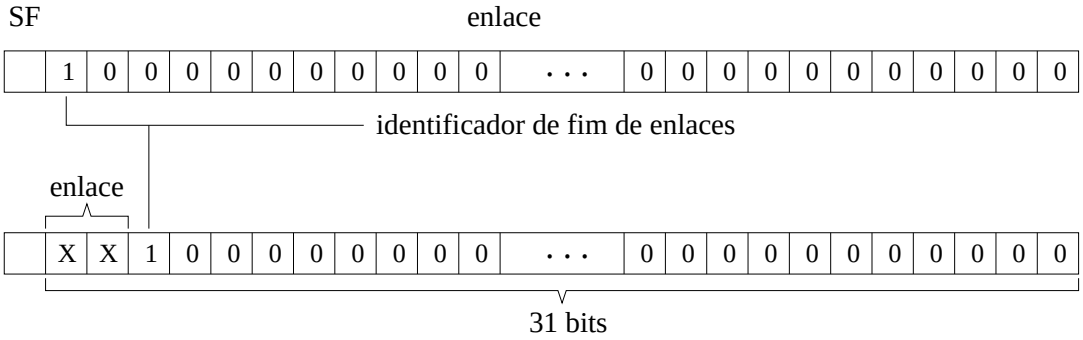

Figura 1.21. Codificação do esquema de marcação utilizando armazenamento híbrido.

Assim, ao perceber que não há mais espaço no cabeçalho para escrever o enlace, o roteador dever criar um identificador para o pacote a partir da aplicação de uma função *hash* em uma parte estática do pacote, ou seja, uma parte que não irá mudar até chegar a vítima. Em seguida, o nó deve armazenar todo o campo *enlace* em sua memória e reiniciá-lo. Por último, é preciso fazer a marcação e atualização do campo *save flag* com o valor 1. Para economizar espaço e, por conseqüência, minimizar o armazenamento de informações nos roteadores, os enlaces são representados por palavras geradas a partir do código de Huffman. O princípio de funcionamento deste código é representar as informações mais freqüentes com uma quantidade menor de bits. Desta maneira, cada roteador armazena uma tabela de enlaces, que contém todos os enlaces com seus vizinhos. Cada enlace será associado a uma freqüência que será utilizada para obter a sua codificação. A freqüência de um enlace é definida pela quantidade de pacotes que o atravessam. O objetivo é evitar ao máximo o armazenamento de informações nos roteadores. A eficiência deste mecanismo depende, basicamente, do número de vizinhos de cada roteador e do número de roteadores no caminho. Choi e Dai consideram que o número médio de vizinhos na Internet é 3,15. Pare este valor, a média de bits utilizados para representar os enlaces é 2. Isto significa que até 15 nós (15×2) não é preciso o armazenamento nos roteadores. Uma rota com mais de 15 nós será armazena pelo menos uma vez em um roteador intermediário. Também é mostrado que a quantidade de informação armazenada é inferior à quantidade das propostas baseadas em *hash*.

A reconstrução da rota de ataque começa a partir do roteador que está diretamente

conectado a vítima. O primeiro passo é consultar a tabela de enlaces deste roteador para identificar a que enlace a primeira palavra se refere. Em seguida, deve-se fazer um deslocamento para esquerda de *n* bits, onde *n* representa o tamanho da palavra recém decodificada. Este procedimento será realizado até que o campo enlace retorne a sua condição inicial (primeiro bit em 1 e o restante em zero). Neste momento, é necessário checar o campo *save flag*. O valor deste campo em 0 significa que o procedimento de reconstrução chegou ao fim. O valor deste campo em 1 significa que existe informação sobre este pacote armazenada no roteador em questão. Por isso, o nó deve recuperar o campo enlace armazenado e continuar o procedimento de reconstrução.

1.5. Um Novo Sistema de Rastreamento de Pacotes IP

Esta seção apresenta brevemente o sistema de rastreamento de pacotes IP proposto [Laufer et al., 2005c]. Esse sistema objetiva rastrear a origem de cada pacote individualmente. Ele é baseado na inserção de informações nos pacotes para evitar o armazenamento na infra-estrutura da rede. Resumidamente, cada roteador insere no pacote uma "assinatura" que indica a sua presença na rota. Ao receber um pacote de ataque, a vítima usa as marcações feitas pelos roteadores para reconstruir a rota reversa. Para diminuir o espaço necessário no cabeçalho e o custo de processamento, um Filtro de Bloom é embutido em cada pacote para armazenar a rota de ataque. Para isso, um campo fixo é reservado para o filtro no cabeçalho do pacote. A técnica de Filtro de Bloom é usada para que a quantidade de informações inserida seja reduzida e permaneça constante. Um tamanho constante para as marcações é importante para evitar tanto a fragmentação dos pacotes quanto o processamento resultante da adição de dados. Além disso, uma generalização do Filtro de Bloom foi proposta para evitar que o atacante possa forjar as "assinaturas" e prejudicar o rastreamento.

O procedimento de marcação para o sistema de rastreamento é bem simples e ocorre pouco antes de reencaminhar um pacote. Neste momento, o roteador insere no filtro daquele pacote o endereço IP da sua interface de saída. Dessa forma, ao receber um pacote de ataque, a vítima dispõe de um filtro cujos elementos são os endereços de todos os roteadores da rota de ataque. Uma vantagem importante desse procedimento de marcação é o seu baixo processamento adicional. Na verdade, o cálculo dos valores *hash* não precisa ser realizado para cada pacote roteado. Os valores *hash* do endereço IP de cada interface do roteador podem ser calculados inicialmente e armazenados em um registrador existente para cada interface. Esse registrador pode ser interpretado como um Filtro de Bloom com somente um elemento inserido: o endereço IP da interface. Quando um pacote vai ser encaminhado, o seu filtro é atualizado com o resultado de uma operação OU bit-a-bit do próprio filtro do pacote com o registrador da interface de saída do roteador. Dessa forma, o endereço IP da interface de saída do roteador é adicionado ao filtro do pacote de maneira eficiente.

Para reconstruir a rota de ataque, o seguinte algoritmo é utilizado. Inicialmente, a vítima verifica quais dos seus roteadores vizinhos estão presentes no filtro do pacote recebido. Aquele que for reconhecido como elemento do filtro é identificado como o roteador pelo qual o pacote chegou e é, portanto, integrado à rota de ataque. Em seguida, o roteador vizinho recebe o filtro da vítima de forma a continuar o procedimento de reconstrução. Ele então verifica qual dos seus roteadores vizinhos também é reconhecido

como elemento do filtro, identificando assim o próximo componente da rota de ataque. O processo é então realizado sucessivamente por cada roteador visando reconstruir o caminho do pacote até a sua verdadeira origem. Quando nenhum roteador é reconhecido, o procedimento termina e o último roteador identificado é considerado a fonte do ataque. A Figura 1.22 ilustra a reconstrução de rota iniciada pela vítima V em direção ao atacante A. Inicialmente, o atacante envia um pacote para a vítima que passa por (R_5, R_4, R_2, R_1). Ao receber o pacote de ataque, a vítima inicia o procedimento de reconstrução, testando a presença de R_1 no filtro do pacote recebido (1). Como R_1 é reconhecido, ele recebe o filtro de V e continua o procedimento. Assim, R_1 verifica a presença dos seus vizinhos R_2 e R_3 no filtro (2). Como somente R_2 é reconhecido, o filtro é então repassado somente para R_2, que faz o mesmo procedimento com seu vizinho R_4 (3). O roteador R_4 verifica qual dos seus dois vizinhos R_3 e R_5 é reconhecido pelo filtro (4); somente R_5 é reconhecido. Finalmente, R_5 testa a presença de R_7 no filtro (5). Uma resposta negativa é retornada e o procedimento de reconstrução termina.

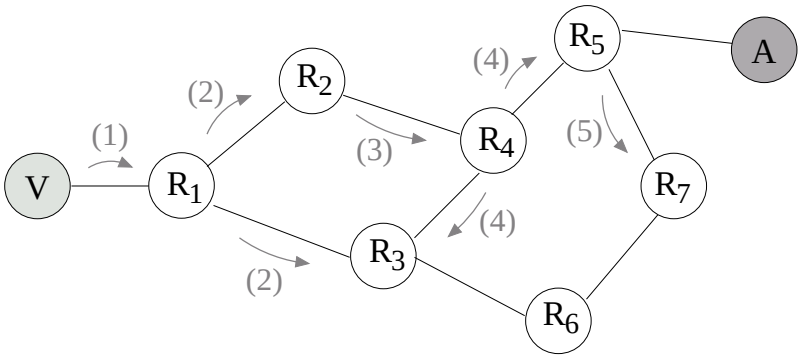

Figura 1.22. Exemplo do procedimento de reconstrução de rota.

Algumas vantagens surgem como resultado da adoção dessa abordagem. Em primeiro lugar, a rota completa de cada pacote pode ser determinada individualmente. Tal comportamento é idealizado por todo sistema de rastreamento de pacotes, uma vez que possibilita a identificação de qualquer fonte em um ataque distribuído. Além disso, nenhuma informação é armazenada na infra-estrutura de rede. Todos os dados relativos ao rastreamento estão localizados na própria vítima, que opta por guardá-los ou não de acordo com a política de segurança local. Outra vantagem é que o sistema proposto não só evita o processamento resultante da fragmentação e da adição de dados ao pacote como também introduz muito pouco processamento adicional ao roteamento. Na verdade, somente uma operação OU bit-a-bit é adicionada ao processo de roteamento. Além disso, é possível realizar o rastreamento após o término do ataque e sem ajuda de operadores de rede. No caso, todo o procedimento de reconstrução pode ser automatizado e tornar-se independente de intervenções manuais.

Por outro lado, esta abordagem também possui desvantagens presentes em outros sistemas de rastreamento [Bellovin et al., 2003], [Dean et al., 2002], [Savage et al., 2001], [Snoeren et al., 2002], [Song e Perrig, 2001]. Primeiramente, um processamento adicional é introduzido ao roteamento de pacotes. Embora o sistema proposto introduza menos processamento que outros sistemas, roteadores com poucos recursos podem ser afetados. Além disso, como em qualquer outro sistema de rastreamento, a cooperação dos roteado-

res é fundamental para a correta marcação dos pacotes. Se alguns roteadores não marcam os pacotes, é provável que existam lacunas na rota reconstruída e que a origem do ataque não seja encontrada. Uma outra desvantagem é que o próprio atacante não é encontrado pelo rastreamento; na verdade, somente o roteador mais próximo do atacante é revelado. Após a identificação deste roteador, maiores esforços são necessários para identificar o atacante. Por fim, a adoção de um Filtro de Bloom para representar a rota de ataque introduz uma certa probabilidade de falso positivo. Durante o algoritmo de reconstrução, um falso positivo implica a incorreta integração de um roteador à rota de ataque. Porém, se esta probabilidade é pequena, a ocorrência de falsos positivos não tem impacto significativo na reconstrução. Algumas rotas para um mesmo pacote existiriam em paralelo, mas ainda assim o escopo de possíveis atacantes seria restringido. No entanto, como o atacante tem controle sobre o conteúdo inicial do pacote, ele pode preencher com 1 todos os bits do cabeçalho do pacote que são reservados ao filtro. Ao saturar o filtro, o atacante faz com que a vítima receba um filtro cujos bits estão todos preenchidos com 1. Conseqüentemente, todo roteador é integrado à rota de ataque durante a reconstrução, tornando impraticável a descoberta da rota verdadeira.

Para minimizar a possibilidade do atacante burlar o sistema e torná-lo menos dependente da condição inicial do filtro, uma generalização do Filtro de Bloom também é proposta. O chamado Filtro de Bloom Generalizado (FBG) estaria integrado a cada pacote, de forma a armazenar os roteadores atravessados. A idéia básica do FBG é utilizar tanto funções *hash* que preenchem bits como funções que zeram bits. Desta forma, é possível mostrar que a probabilidade de falso positivo é limitada e depende pouco da condição inicial do filtro. Por outro lado, falsos negativos que eram inexistentes no Filtro de Bloom convencional são introduzidos com esta generalização. Na Subseção 1.5.1, a idéia do FBG é formalizada e uma análise das probabilidades de falso negativo e falso positivo é desenvolvida a fim de mostrar a eficácia dessa nova abordagem.

1.5.1. O Filtro de Bloom Generalizado

Assim como o filtro convencional, o Filtro de Bloom Generalizado é uma estrutura de dados usada para representar de forma compacta um conjunto $S = \{s_1, s_2, \ldots, s_n\}$ de n elementos. Ele é constituído por um vetor de m bits e por $k_0 + k_1$ funções *hash* independentes $g_1, g_2, \ldots, g_{k_0}, h_1, h_2, \ldots, h_{k_1}$ cujas saídas variam uniformemente no espaço discreto $\{0, 1, \ldots, m-1\}$. O vetor de bits é calculado de maneira semelhante ao do caso convencional. Entretanto, não há mais a restrição de que seus bits são inicialmente zerados. No FBG, os bits do vetor podem assumir qualquer valor inicial. Para cada elemento $s_i \in S$, os bits do vetor correspondentes às posições $g_1(s_i), g_2(s_i), \ldots, g_{k_0}(s_i)$ são zerados e os bits correspondentes às posições $h_1(s_i), h_2(s_i), \ldots, h_{k_1}(s_i)$ são preenchidos com 1. No caso de uma colisão entre uma função g_i com uma função h_j em um mesmo elemento, arbitra-se que o bit é zerado $\forall i, j$. O mesmo bit pode ser zerado ou preenchido diversas vezes sem restrições. A Figura 1.23 ilustra a inserção de um elemento em um filtro generalizado. Posteriormente, testes de pertinência podem ser realizados visando determinar a afiliação de um elemento ao conjunto. Para verificar se um elemento x pertence a S, é preciso checar se os bits $g_1(x), g_2(x), \ldots, g_{k_0}(x)$ estão zerados e se os bits $h_1(x), h_2(x), \ldots, h_{k_1}(x)$ estão preenchidos com 1. Se pelo menos um bit está invertido, então assume-se que $x \notin S$. Diferentemente do filtro convencional, agora há uma possibilidade de um elemento $x \in S$ não

ser reconhecido pelo filtro, criando um falso negativo. Tal anomalia ocorre quando pelo menos um dos bits $g_1(x), g_2(x), \ldots, g_{k_0}(x)$ é preenchido ou quando pelo menos um dos bits $h_1(x), h_2(x), \ldots, h_{k_1}(x)$ é zerado por algum outro elemento inserido posteriormente. Por outro lado, se nenhum bit está invertido, então assume-se que $x \in S$. Na verdade, um elemento externo $x \notin S$ pode ser reconhecido como um autêntico elemento do conjunto, criando um falso positivo. Um falso positivo ocorre quando os bits $g_1(x), g_2(x), \ldots, g_{k_0}(x)$ estão zerados e os bits $h_1(x), h_2(x), \ldots, h_{k_1}(x)$ estão preenchidos com 1 em virtude de um subconjunto dos elementos de S ou da própria condição inicial do vetor.

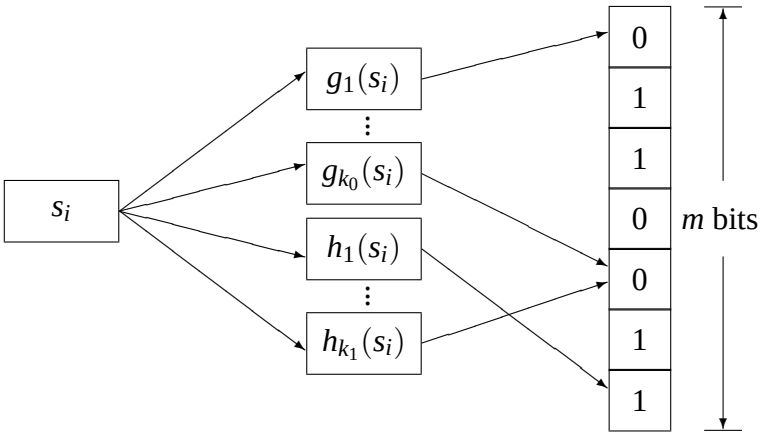

Figura 1.23. Inserção de um elemento em um Filtro de Bloom Generalizado.

Falsos Positivos

A probabilidade de falso positivo de um FBG é calculada de maneira similar à do caso convencional. Entretanto, é necessário primeiramente calcular a probabilidade de um bit ser preenchido ou zerado durante a inserção de um elemento. Dado que no caso de uma colisão as funções g_i sempre têm prioridade sobre as funções h_j, a probabilidade q_0 de um determinado bit ser zerado durante a inserção de um elemento é expressa pela probabilidade de pelo menos uma das k_0 funções zerar o bit. Assim, q_0 é expresso por

$$q_0 = \left[1 - \left(1 - \frac{1}{m}\right)^{k_0}\right] \approx \left(1 - e^{-k_0/m}\right). \tag{15}$$

Por outro lado, a probabilidade q_1 de um determinado bit ser preenchido durante a inserção de um elemento é a probabilidade de pelo menos uma das k_1 funções preencher o bit e nenhuma das k_0 funções o zerar. Dessa forma, q_1 é definido como

$$q_1 = \left[1 - \left(1 - \frac{1}{m}\right)^{k_1}\right]\left(1 - \frac{1}{m}\right)^{k_0} \approx \left(1 - e^{-k_1/m}\right)e^{-k_0/m}. \tag{16}$$

Por fim, a probabilidade de um bit específico permanecer intocado, isto é, não ser nem preenchido e nem zerado por um elemento, é simplesmente

$$(1-q_0-q_1) = \left(1-\frac{1}{m}\right)^{k_0+k_1} \approx e^{-(k_0+k_1)/m}. \quad (17)$$

Como o mesmo cálculo se aplica para todos os bits do vetor, na média, uma fração q_0 de bits é zerada, uma fração q_1 de bits é preenchida e uma fração $(1-q_0-q_1)$ de bits permanece intocada a cada inserção de elemento. Seguindo o mesmo raciocínio, tem-se na média $b_0 = m.q_0$ bits zerados, $b_1 = m.q_1$ bits preenchidos e $(m-b_0-b_1)$ bits intocados a cada inserção.

A partir destes valores, é possível determinar a fração de bits zerados e a fração de bits preenchidos do vetor depois das n inserções. A probabilidade $p_0(n)$ de um bit estar em 0 após as n inserções é calculada através das probabilidades de $n+1$ eventos mutuamente exclusivos. O primeiro evento é aquele onde o bit está inicialmente em 0 e permanece intocado pelos n elementos inseridos. Notando que $p_0(0)$ representa a probabilidade do bit estar inicialmente em 0, a probabilidade de tal evento é $p_0(0)\,(1-q_0-q_1)^n$. Os outros n eventos são aqueles onde o bit é zerado pelo $(n-i)$-ésimo elemento e não é mais tocado pelos i elementos seguintes, para $0 \leq i \leq n-1$. A probabilidade de ocorrência de cada um destes eventos é expressa por $q_0\,(1-q_0-q_1)^i$. Dessa forma, a probabilidade $p_0(n)$ pode ser expressa por

$$\begin{aligned} p_0(n) &= p_0(0)\,(1-q_0-q_1)^n + \sum_{i=0}^{n-1} q_0\,(1-q_0-q_1)^i \\ &= p_0(0)\,(1-q_0-q_1)^n + q_0\left[\frac{1-(1-q_0-q_1)^n}{1-(1-q_0-q_1)}\right] \\ &= p_0(0)\,(1-q_0-q_1)^n + \frac{q_0}{q_0+q_1}\Big[1-(1-q_0-q_1)^n\Big]. \end{aligned} \quad (18)$$

Analogamente, a probabilidade $p_1(n)$ de um bit estar em 1 depois das n inserções é

$$\begin{aligned} p_1(n) &= p_1(0)\,(1-q_0-q_1)^n + \sum_{i=0}^{n-1} q_1\,(1-q_0-q_1)^i \\ &= p_1(0)\,(1-q_0-q_1)^n + q_1\left[\frac{1-(1-q_0-q_1)^n}{1-(1-q_0-q_1)}\right] \\ &= p_1(0)\,(1-q_0-q_1)^n + \frac{q_1}{q_0+q_1}\Big[1-(1-q_0-q_1)^n\Big], \end{aligned} \quad (19)$$

e obviamente $p_0(n)+p_1(n)=1$. Dado que o mesmo cálculo pode ser aplicado para todos os bits do vetor, na média, uma fração $p_0(n)$ dos bits fica em 0 e uma fração $p_1(n)$ dos bits fica em 1 depois de n inserções.

A partir das proporções de bits no vetor, a probabilidade de falso positivo é calculada de maneira trivial. Dado que na média b_0 bits são zerados e b_1 bits são preenchidos a cada inserção de um elemento, a probabilidade de falso positivo f_p de um FBG é calculada da seguinte forma

$$f_p = p_0(n)^{b_0}\,p_1(n)^{b_1} = p_0(n)^{b_0}\,[1-p_0(n)]^{b_1} = [1-p_1(n)]^{b_0}\,p_1(n)^{b_1}. \quad (20)$$

Falsos Negativos

Falsos positivos ocorrem para elementos externos com a mesma probabilidade para cada elemento checado. Por outro lado, falsos negativos ocorrem somente para elementos *inseridos* com uma probabilidade diferente para cada elemento. Na verdade, um fator que afeta diretamente a probabilidade de falso negativo é a ordem de inserção. Por exemplo, os elementos inseridos primeiro têm uma maior probabilidade de serem falsos negativos do que os elementos inseridos por último. Isso ocorre porque mais inserções são realizadas depois dos primeiros elementos e, portanto, é mais provável que um dos seus bits marcados seja invertido.

A probabilidade de falso negativo pode ser calculada se antes for determinada a probabilidade de um bit do $(n-i)$-ésimo elemento inserido não ser invertido pelos i elementos seguintes, para $0 \leq i \leq n-1$. Como nos falsos positivos, a probabilidade $p_{00}(n-i)$ de um bit zerado pelo $(n-i)$-ésimo elemento permanecer zerado ao fim das i inserções é calculada a partir das probabilidades de $i+1$ eventos mutuamente exclusivos. O primeiro evento é aquele em que o bit permanece intocado por todas as i inserções subseqüentes; tal evento ocorre com probabilidade $(1-q_0-q_1)^i$. Os outros i eventos são aqueles onde o bit é zerado pelo $(n-j)$-ésimo elemento e não é mais tocado pelos j elementos seguintes, para $0 \leq j \leq i-1$. Dessa maneira, a probabilidade $p_{00}(n-i)$ é expressa por

$$\begin{aligned} p_{00}(n-i) &= (1-q_0-q_1)^i + \sum_{j=0}^{i-1} q_0 (1-q_0-q_1)^j \\ &= (1-q_0-q_1)^i + q_0 \left[\frac{1-(1-q_0-q_1)^i}{1-(1-q_0-q_1)} \right] \\ &= (1-q_0-q_1)^i + \frac{q_0}{q_0+q_1} \left[1-(1-q_0-q_1)^i \right]. \end{aligned} \tag{21}$$

Analogamente, a probabilidade $p_{11}(n-i)$ de um bit preenchido pelo $(n-i)$-ésimo elemento permanecer preenchido após as i inserções seguintes é

$$\begin{aligned} p_{11}(n-i) &= (1-q_0-q_1)^i + \sum_{j=0}^{i-1} q_1 (1-q_0-q_1)^j \\ &= (1-q_0-q_1)^i + q_1 \left[\frac{1-(1-q_0-q_1)^i}{1-(1-q_0-q_1)} \right] \\ &= (1-q_0-q_1)^i + \frac{q_1}{q_0+q_1} \left[1-(1-q_0-q_1)^i \right]. \end{aligned} \tag{22}$$

A partir das Equações 21 e 22, a probabilidade de falso negativo de qualquer elemento inserido no filtro pode ser determinada. A probabilidade de falso negativo $f_n(n-i)$ do $(n-i)$-ésimo elemento não ser reconhecido pelo filtro após as i inserções subseqüentes é calculada tomando-se o complemento da probabilidade de nenhum de seus bits serem

invertidos. Logo, esta probabilidade é igual a

$$f_n(n-i) = 1 - p_{00}(n-i)^{b_0} p_{11}(n-i)^{b_1}. \tag{23}$$

O Filtro de Bloom como um Caso Particular

Como esperado, o Filtro de Bloom é na verdade um caso particular do FBG. Para os falsos positivos, a Equação 20 se reduz à probabilidade do filtro convencional quando seus parâmetros são usados, isto é, $k_0 = 0$, $p_0(0) = 1$ e $p_1(0) = 0$. Nesse caso, tem-se $p_0(n) = e^{-k_1 n/m}$, $p_1(n) = 1 - e^{-k_1 n/m}$, $b_0 = 0$ e $b_1 = m(1 - e^{-k_1/m})$. Expandindo a expressão de b_1 em série e notando que $m \gg k_1$, pode-se chegar à simplificação realizada para o caso convencional

$$\begin{aligned} b_1 &= m(1 - e^{-k_1/m}) \\ &= m\left[1 - \left(1 - \frac{k_1}{m} + \frac{k_1^2}{2m^2} - \cdots\right)\right] \approx k_1. \end{aligned} \tag{24}$$

Logo, a probabilidade de falso positivo f_p se reduz à probabilidade do Filtro de Bloom convencional, conforme a Equação 14.

De forma semelhante, a probabilidade de falso negativo na Equação 23 se reduz a zero para o caso do Filtro de Bloom convencional. Nesse caso, $k_0 = 0$ e, portanto, $b_0 = 0$ e $p_{11} = 1$, para $0 \leq i \leq n-1$. Assim, independentemente de outros parâmetros, a probabilidade de falso negativo é zero. Para o último elemento inserido, ou seja, para o n-ésimo elemento, esta probabilidade também é zero dado que nenhum outro elemento pode inverter os seus bits. Neste caso, $i = 0$ e $p_{00} = p_{11} = 1$; logo, a probabilidade de falso negativo também se reduz a zero.

Aplicação

No sistema de rastreamento proposto, os elementos inseridos no FBG são na verdade os endereços IP dos roteadores. Portanto, o número de elementos n representa o número de roteadores atravessados pelo pacote de ataque. Além disso, o tamanho do vetor de bits m é exatamente o número de bits alocados no cabeçalho do pacote para o FBG. Os parâmetros k_0 e k_1 são o número de funções *hash* que zeram e preenchem bits em cada roteador, respectivamente. Estas funções devem ser as mesmas em cada roteador de forma a permitir a reconstrução da rota de ataque posteriormente. Por fim, $p_0(0)$ e $p_1(0)$ representam a fração inicial de bits zerados e a fração inicial de bits preenchidos, respectivamente. Esses dois parâmetros estão sob o controle do atacante, que é responsável pela criação do pacote de ataque e pela inicialização do conteúdo do FBG.

A implementação do Filtro de Bloom nos roteadores muda um pouco com a generalização. Existem bits que são zerados e bits que são preenchidos a cada inserção e, portanto, uma única operação OU bit-a-bit não é suficiente para atualizar o filtro. Dessa forma, é necessária uma operação OU bit-a-bit seguida de uma operação E bit-a-bit para

preencher e zerar os bits indicados, respectivamente. Assim, dois registradores são necessários para cada interface do roteador. Para configurar esses registradores, o endereço IP da interface é usado como entrada para as funções *hash*. O primeiro registrador tem todos os seus bits inicializados com 0 e os bits indicados pelas funções h_j são preenchidos com 1. O segundo registrador tem os seus bits inicializados com 1 e os bits indicados pelas funções g_i são colocados em 0. Quando o pacote vai ser reencaminhado, o FBG do pacote é atualizado inicialmente com o resultado de uma operação OU bit-a-bit do próprio FBG com o primeiro registrador da interface. Em seguida, uma operação E bit-a-bit do FBG com o segundo registrador é realizada para finalizar a atualização.

A ordem das operações é um resultado da prioridade adotada. Na verdade, se for decidido que as funções h_j têm precedência sobre as funções g_i, a única modificação a ser realizada é que a operação E deve ser realizada antes da operação OU.

1.5.2. Um Procedimento Aprimorado de Reconstrução de Rota

Se por um lado o uso de um FBG limita a ação do atacante na geração de falsos positivos, por outro, ele introduz falsos negativos no procedimento de reconstrução de rota. Um falso negativo significa não detectar um roteador por onde o pacote realmente passou. Ao deixar de detectar um roteador atravessado, também não são detectados todos os roteadores cuja rota depende deste roteador. Por exemplo, suponha que na reconstrução ilustrada na Figura 1.22 há um falso negativo no roteador R_4. Ou seja, o roteador R_4 não é reconhecido pelo filtro generalizado durante a verificação realizada por R_2 (3). Neste caso, os roteadores R_3 e R_5 não são checados, uma vez que R_4 não foi integrado à rota de ataque. Em conseqüência, o roteador R_5 também não é integrado à rota de ataque. Desta forma, somente um falso negativo pode ser suficiente para interromper o procedimento de reconstrução e evitar a determinação da verdadeira rota de ataque.

Para resolver este problema, é proposto um procedimento aprimorado de reconstrução que elimina os falsos negativos ao custo de uma maior probabilidade de falso positivo. Este novo procedimento de reconstrução é baseado no fato que, durante a travessia do pacote pela rede, roteadores subseqüentes podem inverter os bits marcados por roteadores anteriores e causar falsos negativos durante a reconstrução. No exemplo anterior, durante o percurso do pacote de ataque por (R_5, R_4, R_2, R_1), ou R_1 ou R_2 inverteu algum bit marcado previamente por R_4. Em conseqüência, o filtro do pacote recebido não reconhece R_4 durante o procedimento de reconstrução de rota. De forma a evitar este problema, cada roteador envia junto com o FBG algumas informações adicionais para os seus vizinhos durante a reconstrução. Esta informação contém os bits marcados pelo próprio roteador e pelos roteadores anteriores no procedimento de reconstrução. A Figura 1.24 ilustra brevemente este conceito. Na figura, o roteador R_1 primeiramente reconhece o roteador R_2 no FBG recebido pela vítima e, portanto, R_1 lhe repassa o FBG de forma a continuar a reconstrução. Entretanto, R_1 também envia outros dois vetores de bits, m_0 e m_1, indicando quais bits foram zerados e preenchidos, respectivamente, por R_1 durante a travessia do pacote. Por sua vez, o roteador R_2, ao reconhecerem algum roteador vizinho, também lhe repassa o próprio FBG e os dois vetores de bits adicionais indicando as marcações feitas por ele e por R_1. Desta forma, um roteador R_i sempre repassa para os próximos roteadores o FBG junto com dois vetores de bits, m_0 e m_1, indicando os bits zerados e preenchidos por R_i e pelos roteadores anteriores no procedimento de reconstrução.

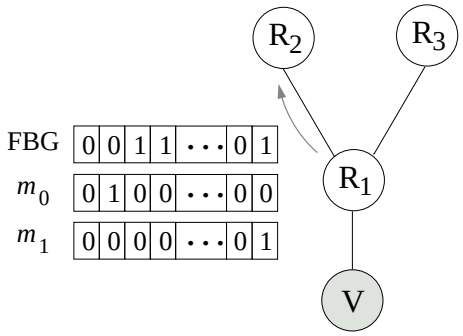

Figura 1.24. O procedimento aprimorado de reconstrução de rota.

O processo de verificação dos vizinhos se altera um pouco em virtude da informação adicional recebida por cada roteador. Primeiramente, antes de checar a presença de cada vizinho no FBG, o roteador atualiza os vetores m_0 e m_1 usando o endereço IP da interface pela qual a requisição de reconstrução chegou. Após a atualização, os testes dos vizinhos são iniciados. Quando um vizinho não é reconhecido pelo FBG, o roteador não descarta aquele vizinho diretamente. Ao invés disso, o roteador verifica se algum bit marcado por este vizinho foi invertido por algum roteador seguinte durante a travessia do pacote. Para isso, ele utiliza os vetores m_0 e m_1 recebidos. Desta forma, caso os bits do vizinho que estão invertidos no FBG estiverem marcados como reescritos, então o vizinho é reconhecido como um elemento do filtro. Caso contrário, aquele vizinho é então descartado. No exemplo dado anteriormente, ao não reconhecer o roteador R_4 no FBG, o roteador R_2 verifica se os bits invertidos de R_4 estão marcados ou no vetor m_0 ou no vetor m_1. Como certamente eles estão, o roteador R_4 é integrado à rota de ataque e o procedimento de reconstrução continua.

Uma importante vantagem do procedimento aprimorado de reconstrução é que falsos negativos não podem mais ocorrer. Dado que os bits reescritos de cada roteador agora podem ser checados a cada salto, os roteadores realmente atravessados são sempre integrados à rota reconstruída. Portanto, a rota de ataque real está *sempre* no grafo de ataque reconstruído. Por outro lado, a probabilidade de falso positivo aumenta à medida que um roteador é testado mais longe da vítima e mais perto do atacante. Isso ocorre porque a fração dos bits marcados em m_0 e m_1 aumenta para cada roteador integrado à rota de ataque. Assim, roteadores que não eram antes reconhecidos porque algum dos seus bits estava invertido, agora podem ser reconhecidos com maior probabilidade. Entretanto, resultados experimentais comprovam que isto não é um problema grave e que o atacante pode ser facilmente identificado [Laufer et al., 2005a], [Laufer, 2005].

A probabilidade de falso positivo do procedimento aprimorado de reconstrução de rota pode ser calculada de uma maneira simples. Primeiramente, é calculada a probabilidade de um bit específico de m_0 estar preenchido em um determinado roteador durante a reconstrução. É importante ressaltar que um bit preenchido em um roteador não significa necessariamente que o bit foi preenchido por este roteador. No caso, o bit pode ter sido preenchido pelo próprio roteador ou por algum roteador anterior no procedimento de reconstrução. Este evento ocorre com probabilidade q_0, uma vez que preencher um bit de m_0 é equivalente a zerar um bit no FBG. Portanto, a probabilidade $s_0(i)$ de um bit

específico de m_0 estar preenchido em um roteador a i saltos da vítima é a probabilidade de pelo menos um dos roteadores anteriores ou o próprio roteador preencher o bit. Desta forma, a probabilidade $s_0(i)$ é

$$s_0(i) = 1 - (1 - q_0)^i. \tag{25}$$

De maneira semelhante, a probabilidade $s_1(i)$ de um bit específico de m_1 estar preenchido em um determinado roteador a i saltos da vítima é a probabilidade de pelo menos um roteador anterior ou o próprio roteador preencher o bit. A probabilidade deste evento é q_1, uma vez que preencher um bit em m_1 é equivalente a preencher um bit no FBG. Portanto, a probabilidade $s_1(i)$ é definida como

$$s_1(i) = 1 - (1 - q_1)^i. \tag{26}$$

Como o mesmo cálculo pode ser realizado para cada bit dos dois vetores, na média, uma fração $s_0(i)$ de bits está preenchida em m_0 e uma fração $s_1(i)$ de bits está preenchida em m_1 durante o procedimento de reconstrução em um roteador a i saltos da vítima.

Desta maneira, um bit é interpretado como zero nos testes de pertinência com os vizinhos se um bit estiver zerado no FBG ou se ele estiver preenchido tanto no FBG e no vetor m_1. Portanto, a fração $t_0(i)$ de bits interpretados como zero nos testes de pertinência realizados por um determinado roteador a i saltos da vítima é

$$t_0(i) = p_0(n) + p_1(n)s_1(i). \tag{27}$$

Analogamente, a fração $t_1(i)$ de bits interpretados como um é

$$t_1(i) = p_1(n) + p_0(n)s_0(i). \tag{28}$$

A partir das Equações (27) e (28), a probabilidade de um falso positivo $f_p(i)$ para um determinado roteador a i saltos da vítima no procedimento aprimorado de reconstrução de rota pode ser calculada e expressa por

$$f_p(i) = t_0(i)^{b_0} t_1(i)^{b_1}, \tag{29}$$

onde b_0 e b_1 são respectivamente o número médio de bits em 0 e o número médio de bits em 1 necessários para um falso positivo ocorra, conforme descrito na Subseção 1.5.1.

Referências

[Aljifri et al., 2003] Aljifri, H., Smets, M. e Pons, A. (2003). IP Traceback using Header Compression. *Computers & Security*, 22(2):136–151.

[Bai et al., 2004] Bai, C., Feng, G. e Wang, G. (2004). Algebraic Geometric Code Based IP Traceback. Em *IEEE International Conference on Performance, Computing, and Communications*, páginas 49–56, Phoenix, AZ, EUA.

[Belenky e Ansari, 2003a] Belenky, A. e Ansari, N. (2003a). Accommodating Fragmentation in Deterministic Packet Marking for IP Traceback. Em *IEEE GLOBECOM 2003 Conference*, páginas 1374–1378, San Francisco, CA, EUA.

[Belenky e Ansari, 2003b] Belenky, A. e Ansari, N. (2003b). IP Traceback With Deterministic Packet Marking. *IEEE Communications Letters*, 7(4):162–164.

[Bellovin et al., 2003] Bellovin, S. M., Leech, M. D. e Taylor, T. (2003). ICMP Traceback Messages. *Internet Draft: draft-ietf-itrace-04.txt*.

[Bloom, 1970] Bloom, B. H. (1970). Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, 7(13):442–426.

[Broder e Mitzenmacher, 2003] Broder, A. e Mitzenmacher, M. (2003). Network Applications of Bloom Filters: A Survey. *Internet Mathematics*, 1(4):485–509.

[Burch e Cheswick, 2000] Burch, H. e Cheswick, B. (2000). Tracing Anonymous Packets to their Approximate Source. Em *USENIX LISA'00*, páginas 319–327, Nova Orleans, LA, EUA.

[Büyükkokten, 2005] Büyükkokten, O. (2005). Orkut.com. http://www.orkut.com/.

[CERT, 1996] CERT (1996). *CERT Advisory CA-1996-26 Denial-of-Service Attack via ping*. http://www.cert.org/advisories/CA-1996-26.html.

[CERT, 1997] CERT (1997). *CERT Advisory CA-1997-28 IP Denial-of-Service Attacks*. http://www.cert.org/advisories/CA-1997-28.html.

[CERT, 1998] CERT (1998). *CERT Advisory CA-1998-01 Smurf IP Denial-of-Service Attacks*. http://www.cert.org/advisories/CA-1998-01.html.

[Choi e Dai, 2004] Choi, K. H. e Dai, H. K. (2004). A Marking Scheme Using Huffman Codes for IP Traceback. Em *7th International Symposium on Parallel Architectures, Algorithms and Networks - ISPAN'04*, páginas 421–428, Hong Kong, China.

[Cisco, 2003] Cisco (2003). *Cisco Security Advisory: Cisco IOS Interface Blocked by IPv4 Packets*. Cisco Systems, Inc.
http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml.

[CNN.com, 2000] CNN.com (2000). *Denial of service hackers take on new targets*.
http://www.cnn.com/2000/TECH/computing/02/09/denial.of.service.03.

[Dean et al., 2002] Dean, D., Franklin, M. e Stubblefield, A. (2002). An Algebraic Approach to IP Traceback. *ACM Transactions on Information and System Security*, 5(2):119–137.

[Dittrich, 1999a] Dittrich, D. (1999a). *The DoS Project's 'trinoo' distributed denial of service attack tool*. http://staff.washington.edu/dittrich/misc/trinoo.analysis.txt.

[Dittrich, 1999b] Dittrich, D. (1999b). *The 'stacheldraht' distributed denial of service attack tool*. http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.txt.

[Dittrich, 1999c] Dittrich, D. (1999c). *The 'Tribe Flood Network' distributed denial of service attack tool*. http://staff.washington.edu/dittrich/misc/tfn.analysis.txt.

[Ferguson e Senie, 2000] Ferguson, P. e Senie, D. (2000). Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. *RFC 2827*.

[FTC, 2005] FTC (2005). The CAN-SPAM Act: Requirements for Commercial Emailers. http://www.ftc.gov/bcp/conline/pubs/buspubs/canspam.htm.

[Garber, 2000] Garber, L. (2000). Denial-of-Service Attacks Rip the Internet. *IEEE Computer*, 4(33):12–17.

[Gibson, 2001] Gibson, S. (2001). *The Strange Tale of the Attacks Against GRC.COM*. Gibson Research Corporation. http://www.grc.com/dos/grcdos.htm.

[Globo Online, 2005] Globo Online (2005). Brasil é 5º maior receptor de spam; spywares representam 22% das infecções. http://oglobo.globo.com/online/plantao/169450846.asp.

[Gont, 2004] Gont, F. (2004). ICMP Attacks Against TCP. *Internet Draft: draft-gont-tcpm-icmp-attacks-03.txt*.

[Goodrich, 2002] Goodrich, M. T. (2002). Efficient Packet Marking for LargeScale IP Traceback. Em *9th ACM Conference on Computer and Communications Security - CCS'02*, páginas 117–126, Washington, DC, EUA.

[Gordon et al., 2005] Gordon, L. A., Loeb, M. P., Lucyshyn, W. e Richardson, R. (2005). *2005 CSI/FBI Computer Crime and Security Survey*.

[Gray e Fried, 2003] Gray, P. e Fried, I. (2003). *Al-Jazeera suffers DoS attack*. ZDNet UK. http://news.zdnet.co.uk/business/0,39020645,2132585,00.htm.

[Grupo Brasil AntiSPAM, 2005] Grupo Brasil AntiSPAM (2005). Código de Ética AntiSPAM e Melhores Práticas de Uso de Mensagens Eletrônicas. http://brasilantispam.org/main/codigo.htm.

[Hilgenstieler e Duarte Jr., 2004] Hilgenstieler, E. e Duarte Jr., E. P. (2004). Uma Arquitetura para Rastreamento de Pacotes na Internet. Em *IV Workshop em Segurança de Sistemas Computacionais - WSeg 2004*, Gramado, RS, Brasil.

[Holmes, 2005] Holmes, N. (2005). In Defense of Spam. *IEEE Computer Magazine*, 38(4):86–88.

[Hormel Foods, 2000] Hormel Foods (2000). Your Use of Our Trademark SPAM on Your "Page-O-SPAM" Website. http://www.rsi.com/spam/.

[IBM Report, 2005] IBM Report (2005). Phishing attacks in May jumped more than 200 percent. Relatório técnico, IBM.

[ICQ, 2005] ICQ (2005). Icq.com - community, people search and messaging service! http://www.icq.com/.

[Jacobson, 1990] Jacobson, V. (1990). Compressing TCP/IP Headers for Low-Speed Serial Links. *RFC 1144*.

[Laufer, 2005] Laufer, R. P. (2005). Rastreamento de Pacotes IP contra Ataques de Negação de Serviço. Tese de mestrado, COPPE/UFRJ.

[Laufer et al., 2005a] Laufer, R. P., Velloso, P. B., de O. Cunha, D. e Duarte, O. C. M. B. (2005a). Um Procedimento Alternativo de Reconstrução de Rota para o Rastreamento de Pacotes IP. Em *XXII Simpósio Brasileiro de Telecomunicações - SBrT'05*, Campinas, SP, Brasil.

[Laufer et al., 2005b] Laufer, R. P., Velloso, P. B. e Duarte, O. C. M. B. (2005b). Defeating DoS Attacks with IP Traceback. Em *IFIP Open Conference on Metropolitan Area Networks - MAN'2005*, Ho Chi Minh, Vietnã.

[Laufer et al., 2005c] Laufer, R. P., Velloso, P. B. e Duarte, O. C. M. B. (2005c). Um Novo Sistema de Rastreamento de Pacotes IP contra Ataques de Negação de Serviço. Em *XXIII Simpósio Brasileiro de Redes de Computadores - SBRC'2005*, Fortaleza, CE, Brasil.

[Lee et al., 2004] Lee, T.-H., Wu, W.-K. e Huang, T.-Y. W. (2004). Scalable Packet Digesting Schemes for IP Traceback. Em *IEEE International Conference on Communications ICC'04*, páginas 1008–1013, Paris, França.

[Leech, 1996] Leech, M. (1996). DefUsername/Password Authentication for SOCKS V5. *RFC 1929*.

[Li et al., 2002] Li, J., Mirkovic, J., Wang, M., Reiher, P. e Zhang, L. (2002). SAVE: Source Address Validity Enforcement Protocol. Em *Proceedings of the IEEE INFOCOM 2002 Conference*, páginas 1557–1566, Nova Iorque, NY, EUA.

[Li et al., 2004] Li, J., Sung, M., Xu, J. e Li, L. (2004). Large-Scale IP Traceback in High-Speed Internet: Practical Techniques and Theoretical Foundation. Em *Proceedings of the 25th IEEE Symposium on Security and Privacy*, Oakland, CA, EUA.

[Liu et al., 2003] Liu, J., Lee, Z.-J. e Chung, Y.-C. (2003). Efficient Dynamic Probabilistic Packet Marking for IP Traceback. Em *IEEE International Conference on Networks - ICON'03*, páginas 475–480, Sydney, Austrália.

[Mankin et al., 2001] Mankin, A., Massey, D., Wu, C.-L., Wu, S. F. e Zhang, L. (2001). On Design and Evaluation of "Intention-Driven" ICMP Traceback. Em *Proceedings of the IEEE ICCCN 2001 Conference*, Scottsdale, AZ, EUA.

[M.Degermark et al., 1999] M.Degermark, Nordgren, B. e S.Pink (1999). IP Header Compression. *RFC 2507*.

[Microsoft, 2002] Microsoft (2002). *Stop 0A in Tcpip.sys When Receiving Out Of Band (OOB) Data*. Microsoft Corporation. http://support.microsoft.com/kb/q143478.

[Microsoft, 2005] Microsoft (2005). Microsoft Network - MSN. http://messenger.msn.com/.

[Mirkovic et al., 2004] Mirkovic, J., Dietrich, S., Dittrich, D. e Reiher, P. (2004). *Internet Denial of Service: Attack and Defense Mechanisms*. Prentice Hall PTR, 1ª edição.

[Mirkovic e Reiher, 2004] Mirkovic, J. e Reiher, P. (2004). A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *ACM SIGCOMM Computer Communications Review*, 34(2):39–53.

[Mitzenmacher, 2002] Mitzenmacher, M. (2002). Compressed Bloom Filters. *IEEE/ACM Transactions on Networking*, 10(5):604–612.

[Moore et al., 2001] Moore, D., Voelker, G. e Savage, S. (2001). Inferring Internet Denial of Service Activity. Em *Proceedings of the 2001 USENIX Security Symposium*, Washington, DC, EUA.

[Park e Lee, 2001] Park, K. e Lee, H. (2001). On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial of Service Attack. Em *Proceedings of the IEEE INFOCOM 2001 Conference*, Anchorage, AK, EUA.

[Perkins, 2002] Perkins, C. E. (2002). IP Mobility Support for IPv4. *RFC 3220*.

[Postel, 1981] Postel, J. (1981). Internet Protocol. *RFC 791*.

[Postel, 1983] Postel, J. (1983). Character Generator Protocol. *RFC 864*.

[Reuters, 2004] Reuters (2004). *Scotland Yard and the case of the rent-a-zombies*. ZDNet.com. http://news.zdnet.com/2100-1009_22-5260154.html.

[Sahami et al., 1998] Sahami, M., Dumais, S., Heckerman, D. e Horvitz, E. (1998). A bayesian approach to filtering junk E-mail. Em *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, WI, EUA. AAAI Technical Report WS-98-05.

[Savage et al., 2001] Savage, S., Wetherall, D., Karlin, A. e Anderson, T. (2001). Network Support for IP Traceback. *IEEE/ACM Transactions on Networking*, 9(3):226–237.

[Schuba et al., 1997] Schuba, C. L., Krsul, I. V., Kuhn, M. G., Spafford, E. H., Sundaram, A. e Zamboni, D. (1997). Analysis of a Denial of Service Attack on TCP. Em *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, páginas 208–223, Oakland, CA, EUA.

[Shachtman, 2003] Shachtman, N. (2003). *Porn Purveyors Getting Squeezed*. Wired News. http://wired-vig.wired.com/news/print/0,1294,59574,00.html.

[Snoeren et al., 2001] Snoeren, A. C., Partridge, C., Sanchez, L. A., Jones, C. E., Tchakountio, F., Kent, S. T. e Strayer, W. T. (2001). Hash-Based IP Traceback. Em *Proceedings of the ACM SIGCOMM'01 Conference*, páginas 3–14, San Diego, CA, EUA.

[Snoeren et al., 2002] Snoeren, A. C., Partridge, C., Sanchez, L. A., Jones, C. E., Tchakountio, F., Schwartz, B., Kent, S. T. e Strayer, W. T. (2002). Single-Packet IP Traceback. *IEEE/ACM Transactions on Networking*, 10(6):721–734.

[Song e Perrig, 2001] Song, D. X. e Perrig, A. (2001). Advanced and Authenticated Marking Schemes for IP Traceback. Em *Proceedings of the IEEE INFOCOM 2001 Conference*, Anchorage, AK, EUA.

[SpamAssassin, 2005] SpamAssassin (2005). The apache spamassassin project. http://spamassassin.apache.org/.

[Spammer-X et al., 2004] Spammer-X, Posluns, J. e Sjouwerman, S. (2004). *Inside the SPAM Cartel: Trade Secrets from the Dark Side*. Syngress Publishing, 1^a edição.

[Stone, 2000] Stone, R. (2000). CenterTrack: An IP Overlay Network for Tracking DoS Floods. Em *9th USENIX Security Symposium*, páginas 199–212, Denver, CO, EUA.

[US-CERT, 2005] US-CERT (2005). *Vulnerability Note VU#222750: TCP/IP Implementations Do Not Adequately Validate ICMP Error Messages*. http://www.kb.cert.org/vuls/id/222750.

[Wagner, 2002] Wagner, J. (2002). *Dealing With Massive Attack: DNS Protection*. Internetnews.com. http://www.internetnews.com/dev-news/article.php/1487331.

[Watson, 2004] Watson, P. A. (2004). Slipping in the Window: TCP Reset Attacks. Em *2004 CanSecWest Conference*, Vancouver, Canadá.