RASTREAMENTO DE PACOTES IP
CONTRA ATAQUES DE NEGAÇÃO DE SERVIÇO

Rafael Pinaud Laufer

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:

Prof. Otto Carlos Muniz Bandeira Duarte, Dr.Ing.

Prof. Jussara Marques de Almeida, Ph.D.

Prof. Luís Henrique Maciel Kosmalski Costa, Dr.

Prof. Valmir Carneiro Barbosa, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

SETEMBRO DE 2005

LAUFER, RAFAEL PINAUD

Rastreamento de Pacotes IP contra Ataques de Negação de Serviço [Rio de Janeiro] 2005

XIII, 93 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2005)

Dissertação - Universidade Federal do Rio de Janeiro, COPPE

1. Rastreamento de Pacotes IP

2. Ataques de Negação de Serviço

3. Filtro de Bloom

I. COPPE/UFRJ II. Título (série)

À minha família e a Flavia.

Agradecimentos

Agradeço aos meus pais, Julio e Klaudia, por todo amor, incentivo e orientação passados em todos os anos de minha vida. Aos meus irmãos, Gustavo e Gabriela, por sempre reagirem aos meus implicantes comentários. À minha avó, Maria da Penha, pelos cuidados com seu neto. À minha namorada Flavia, pelo amor, carinho e enorme compreensão nas horas de estudo.

Ao professor e orientador Otto, responsável por grande parte da minha formação pessoal, acadêmica e profissional, por sua amizade, conselhos e orientação. Aos professores Luís Henrique, Rezende e Rubi, pela amizade, ensinamentos e dicas.

Aos amigos e colegas de graduação, Bruno, Guilherme, Igor e Marco, pelos bons momentos passados tanto nas horas de trabalho quanto nas de lazer. Aos amigos e colegas de trabalho, Aurélio, Bernardo, Daniel, Italo, Kleber, Miguel e Pedro, pelas risadas e pela troca de experiências. Ao meu grande amigo Felipe que, mesmo distante, está sempre presente e disposto a ajudar.

Agradeço em particular aos professores Jussara Almeida, Luís Henrique Costa e Valmir Barbosa pela presença na banca examinadora e contribuição neste trabalho.

Ao CNPq, à FAPERJ e ao UOL pelo financiamento da pesquisa.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

RASTREAMENTO DE PACOTES IP
CONTRA ATAQUES DE NEGAÇÃO DE SERVIÇO

Rafael Pinaud Laufer

Setembro/2005

Orientador: Otto Carlos Muniz Bandeira Duarte
Programa: Engenharia Elétrica

Em grande parte dos ataques de negação de serviço, pacotes IP com endereços de origem forjados são usados para ocultar a verdadeira origem do atacante. Uma possível estratégia de defesa é rastrear a origem do ataque, de forma a penalizar o atacante ou isolá-lo da rede. Até o momento, os sistemas propostos para o rastreamento requerem o armazenamento de um grande volume de informação nos roteadores ou um número mínimo de pacotes de ataque recebidos. Neste trabalho, é proposto um novo sistema de rastreamento de pacotes IP capaz de determinar a origem de um único pacote recebido pela vítima sem armazenar estado na infra-estrutura de rede. Para seu funcionamento, é desenvolvida uma generalização da teoria de Filtro de Bloom e realizada sua respectiva análise matemática. Um procedimento aprimorado de reconstrução de rota também é proposto de forma a melhorar o rastreamento. Resultados analíticos e de simulação são apresentados para mostrar a eficácia do sistema proposto. As simulações são realizadas em uma topologia baseada na própria Internet e os resultados mostram que o sistema proposto sempre localiza a verdadeira rota de ataque. Em um cenário com rotas de 15 saltos e usando-se somente 256 bits adicionais por pacote, o sistema proposto encontra em média 1,7 possíveis atacantes.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

IP TRACEBACK AGAINST DENIAL-OF-SERVICE ATTACKS

Rafael Pinaud Laufer

September/2005

Advisor: Otto Carlos Muniz Bandeira Duarte

Department: Electrical Engineering

On most denial-of-service attacks, packets with spoofed source addresses are employed in order to disguise the true origin of the attack. A defense strategy is to trace back the source of the attack for the sake of penalizing the attacker or isolating him from the network. To date, the proposed traceback systems require either large amounts of storage space on routers or a sufficient number of received attack packets. In this work, a new IP traceback system is proposed to determine the source of a single packet received by the victim without storing state in the network infrastructure. For practical purposes, a generalization of the theory of Bloom Filters is developed and a corresponding mathematical evaluation is derived. An enhanced reconstruction procedure is also introduced to improve the traceback process. Analytical and simulation results are presented to show the effectiveness of the proposed system. The simulations are performed in an Internet-based topology and the results show that the proposed system always locates the real attack path. In a simulation scenario with 15-hop routes and using 256 additional bits per packet, the proposed system locates on average 1.7 candidate attackers.

Sumário

Lista de Figuras

Lista de Acrônimos

BGP : *Border Gateway Protocol*;

DDoS : *Distributed Denial of Service*;

DNS : *Domain Name System*;

DoS : *Denial of Service*;

FBG : Filtro de Bloom Generalizado;

GBF : *Generalized Bloom Filter*;

ICMP : *Internet Control Message Protocol*;

IP : *Internet Protocol*;

IRC : *Internet Relay Chat*;

P2P : *Peer to Peer*;

PKI : *Public Key Infrastructure*;

REQ : Requisição;

RESP : Resposta;

TCP : *Transmission Control Protocol*;

UDP : *User Datagram Protocol*.

Capítulo 1

Introdução

A INFRA-ESTRUTURA de roteamento da Internet ainda é vulnerável a ataques anônimos de negação de serviço (*Denial of Service* - DoS) [1–4]. Tais ataques são caracterizados pelo completo desconhecimento da sua verdadeira origem e visam tornar inacessíveis os serviços oferecidos pela vítima. Este objetivo geralmente é alcançado através do envio de pacotes pelo atacante a uma taxa maior do que podem ser tratados pela vítima, fazendo com que legítimas requisições de serviço não sejam atendidas. Em sua versão distribuída (*Distributed* DoS - DDoS), os pacotes são enviados de diferentes origens e o tráfego agregado gerado é responsável pela total inutilização dos serviços da vítima [5].

1.1 Motivação

Ataques de negação de serviço têm ocorrido com freqüência na Internet. Moore *et al.* [6] fizeram um estudo para identificar a ocorrência de ataques de negação de serviço. Dentro das limitações técnicas de medida utilizadas, os autores observaram mais de 4.000 ataques por semana. Como o método empregado não consegue detectar todos os ataques que ocorrem na Internet, este valor é na verdade menor do que o número real de ataques. Um resultado importante mostra ainda que uma grande parte dos ataques são direcionados a vítimas brasileiras. De acordo com os autores, o domínio `.br` é o quarto domínio

mais atacado por inundações visando a negação de serviço. 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*) [7] 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.

Ultimamente, ataques de negação de serviço têm sido realizados de maneira distribuída, com diversos atacantes enviando tráfego para a vítima ao mesmo tempo. O número de ataques distribuídos a grandes sítios é cada vez mais alarmante, sendo inclusive desenvolvidas pragas digitais específicas para tal finalidade [8–12]. Embora menos comuns, também existem ataques de negação de serviço constituídos pelo envio de um único pacote [13–17]. Em ambos os casos, os resultados são desastrosos [18] e a necessidade de uma solução que identifique a verdadeira origem dos pacotes se torna evidente.

Devido à técnica datagrama usada no protocolo IP (*Internet Protocol*), o anonimato do atacante é facilmente mantido, pois é possível injetar pacotes na rede com endereço de origem forjado. Não existe uma entidade ou um mecanismo responsável pela verificação da autenticidade da fonte. Como toda infra-estrutura de roteamento é baseada exclusivamente no endereço de destino, pacotes com endereço de origem forjado geralmente alcançam a vítima sem dificuldades. Outra característica que permite a execução de ataques anônimos é a ausência de estado nos roteadores. Nenhuma informação relativa aos pacotes roteados é armazenada para consultas futuras. Em conseqüência, o encaminhamento de pacotes não deixa "rastros," tornando impossível a dedução da rota percorrida por um pacote.

A identificação da fonte também é dificultada caso ataques indiretos sejam empregados. Estes ataques são caracterizados pelo uso de estações intermediárias entre o atacante e a vítima, de forma a ocultar a sua verdadeira origem. A participação destas estações pode ser classificada como ativa ou passiva, dependendo do seu comportamento. No modo ativo, as estações intermediárias são antes comprometidas e nelas é implantado um programa malicioso cujo objetivo é transformá-las em "zumbis." Desta forma, elas ficam em controle de uma estação-mestre responsável pela coordenação do ataque. Uma vez

recebida a ordem, os zumbis geram um grande volume de tráfego em direção à vítima. Em ataques distribuídos, redes hierárquicas constituídas de diversos zumbis são formadas visando potencializar o efeito do ataque. Por outro lado, no modo passivo, as estações intermediárias atuam apenas como "refletores" do tráfego de ataque. O atacante envia falsas requisições em nome da vítima para estes refletores que, sem perceber que o endereço de origem é forjado, respondem inocentemente em direção à vítima [19]. Além disso, caso o tamanho da resposta seja maior do que o tamanho da requisição, os refletores também atuam como "amplificadores" do tráfego de ataque. Essas estratégias podem ainda ser combinadas para atuar simultaneamente de forma a dificultar a identificação do atacante.

1.2 Objetivo

Este trabalho propõe um sistema de rastreamento de pacotes IP, que objetiva identificar as estações que geram diretamente o tráfego de ataque e a respectiva rota usada por este tráfego. O rastreamento de pacotes é fundamental em qualquer tipo de ataque que emprega técnicas para esconder a sua verdadeira origem. No caso de ataques anônimos de negação de serviço, o primeiro passo a ser realizado é identificar os possíveis atacantes para que outras contramedidas, como a filtragem do tráfego de ataque, possam ser tomadas em seguida. Mesmo sem uma identificação completa da fonte, informações parciais que permitam identificar um roteador mais próximo da origem do ataque são úteis para controlar o tráfego desnecessário na rede. Diversos sistemas de rastreamento propostos sugerem que os roteadores notifiquem a vítima sobre a sua presença no rota de ataque [20–22]. Esta notificação pode ser realizada pelos roteadores através da inserção de informação sobre si próprio nos pacotes roteados. Dessa forma, ao receber um pacote de ataque, a vítima reconhece a presença daquele roteador específico na rota. Depois de recebidas essas informações, a vítima inicia um procedimento de reconstrução de rota, onde as informações recebidas são utilizadas para determinar a verdadeira origem de ataque.

1.3 Contribuições

Neste trabalho, é introduzida uma nova abordagem para o rastreamento de pacotes IP [23–28]. O sistema de rastreamento proposto possui a vantagem de rastrear um ataque a partir de informações contidas nos pacotes de ataques e até mesmo encontrar a rota de ataque a partir de um único pacote. Até o momento, sistemas de rastreamento propostos capazes de identificar a origem do ataque a partir de somente um pacote exigem uma alta capacidade de armazenamento na infra-estrutura de rede. Entretanto, a proposta deste trabalho não armazena nenhuma informação na rede. A idéia do sistema de rastreamento proposto é usar uma estrutura de dados, denominada Filtro de Bloom [29], integrada a cada pacote para armazenar os endereços dos roteadores atravessados. O uso de um Filtro de Bloom no sistema de rastreamento é justificado tanto pela sua eficiência no armazenamento de informações quanto pelo baixo processamento adicional inserido no processo de roteamento. Desta forma, cada roteador insere o seu endereço nos filtros dos pacotes roteados. A partir desta informação, um procedimento de reconstrução pode ser iniciado pela vítima visando determinar a verdadeira origem de cada pacote recebido. Além disso, todo o processo de rastreamento pode ser efetuado após a finalização do ataque e sem nenhuma ajuda de operadores de rede.

Apesar das vantagens apresentadas, o uso de Filtros de Bloom deixa o sistema suscetível a uma simples técnica de evasão que pode ser empregada por atacantes experientes. Por isso, também é proposta neste trabalho uma nova estrutura de dados, denominada Filtro de Bloom Generalizado. As expressões analíticas que ditam o comportamento desta estrutura são derivadas e as suas propriedades no sistema de rastreamento são amplamente exploradas. Essa generalização surge como uma solução natural contra a facilidade de evasão do sistema de rastreamento. É provado tanto por resultados analíticos quanto por resultados de simulação que o uso de um Filtro de Bloom Generalizado para a representação da rota de ataque não pode ser burlado.

Uma outra contribuição deste trabalho é um procedimento aprimorado de reconstrução de rota para o sistema de rastreamento de pacotes. Este procedimento aprimorado é proposto para melhorar a acurácia da rota reconstruída. De forma a comprovar a sua eficácia, este procedimento é simulado em uma topologia baseada na Internet. Através dos

resultados dessas simulações, é mostrado que o procedimento proposto sempre encontra a verdadeira rota de ataque.

1.4 Organização

Este trabalho está organizado da seguinte forma. No Capítulo 2, são apresentadas as principais formas utilizadas por atacantes para que a vítima negue os seus serviços para usuários legítimos. Métodos usados para manter o anonimato durante a condução do ataque também são abordados, assim como as situações onde o rastreamento de pacotes pode ajudar a identificar os atacantes. Em seguida, o Capítulo 3 descreve o sistema de rastreamento proposto neste trabalho. Além disso, o rastreamento de pacotes IP é descrito e os principais trabalhos relacionados ao tema também são abordados nesse capítulo. No Capítulo 4, o Filtro de Bloom é explicado assim como a sua generalização proposta neste trabalho. Expressões analíticas são derivadas e resultados de simulação são apresentados para mostrar as vantagens do uso de um Filtro de Bloom Generalizado no rastreamento de pacotes. O procedimento aprimorado de reconstrução de rota é apresentado no Capítulo 5, apresentando ainda resultados analíticos e de simulação deste procedimento. Por fim, no Capítulo 6, são relatadas as conclusões e os trabalhos futuros.

Capítulo 2

Negação de Serviço

ATAQUES de negação de serviço (*Denial of Service* - DoS) visam prejudicar ou interromper completamente uma atividade legítima [30]. Tal atividade pode ser, por exemplo, o uso de um buscador de páginas, a compra de um determinado produto ou simplesmente a troca de mensagens entre duas entidades. Este objetivo é alcançado através do envio de pacotes específicos para a vítima, de forma a interferir na atividade alvo escolhida. O resultado de um ataque pode ser inesperado, como o congelamento ou a reinicialização da vítima, ou ainda o esgotamento completo de recursos necessários para a oferta do seu serviço.

Neste capítulo, são introduzidas as condições e os principais ataques que levam uma determinada vítima a negar o seu serviço para usuários legítimos. Além disso, também são discutidas as maneiras usadas pelos atacantes para permanecerem anônimos durante a condução do ataque e como o rastreamento de pacotes pode ajudar na identificação destes malfeitores.

2.1 Introdução

Empresas que disponibilizam serviços através da Internet podem sofrer sérios prejuízos devido a ataques de negação de serviço. Em sítios de vendas, um longo período inacessível significa que usuários deixam de comprar os produtos disponibilizados. De-

pendendo da duração do ataque, a reputação da empresa pode ser afetada e, como conseqüência, problemas para atrair novos clientes ou investidores podem surgir no futuro. Caso o sítio obtenha sua renda através de propagandas, por exemplo, tais ataques impossibilitam a sua visualização pelos usuários. Como conseqüência, quebras de contrato, danos à imagem da empresa e até mesmo processos judiciais podem causar grandes prejuízos financeiros.

Uma maneira comum de atingir a negação de um determinado serviço é através do consumo de recursos essenciais para o seu funcionamento, como memória, processamento, espaço em disco ou banda passante. Como esses recursos são limitados, sempre é possível inundar a vítima com um grande número de mensagens de forma a consumir exageradamente estes recursos. O único pré-requisito desse ataque é a existência de uma infra-estrutura que consiga produzir mensagens a uma taxa maior do que podem ser tratadas. Desta forma, a aplicação, a vítima ou a sua infra-estrutura de rede fica sobrecarregada com o tratamento dessas mensagens e não consegue atender ao tráfego de usuários legítimos. Este tipo de ataque é denominado ataque por inundação. Um dos fatores que dificultam muito a adoção de contramedidas 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, o atacante precisa gerar mensagens a uma taxa superior à taxa na qual a vítima, ou a sua infra-estrutura de rede, consegue tratar estas mensagens. Em vítimas com poucos recursos, a negação do serviço consegue ser realizada sem dificuldades, uma vez que mensagens geradas a uma taxa baixa podem tornar o serviço da vítima indisponível. Entretanto, sendo a vítima superdimensionada, como é o caso de servidores famosos, esse resultado não é alcançado tão facilmente. Muitas vezes, o tráfego gerado por somente uma estação de ataque não é suficiente para exaurir os recursos da vítima. Nesses casos, são necessárias diversas estações atacando em conjunto para que os recursos da vítima se esgotem. Estas estações devem estar sincronizadas de alguma forma de modo a iniciarem o ataque ao mesmo tempo para potencializar o seu efeito. Quando estações agem com o objetivo comum de atacar uma

determinada vítima, é constituído o chamado ataque de negação de serviço distribuído (*Distributed* DoS - DDoS).

Mesmo com um superdimensionamento de recursos da vítima, não é possível garantir a sua imunidade quando diversas estações são usadas como geradoras do tráfego de ataque. Na verdade, sempre é possível inundar a vítima e negar o seu serviço desde que um número suficiente de estações participe do ataque. Conseqüentemente, mesmo sítios superdimensionados podem sofrer prejuízos consideráveis. Como exemplo, ataques de negação de serviço distribuídos já foram realizados com sucesso contra sítios bem conhecidos, como Yahoo, Ebay, Amazon.com e CNN.com [18, 31]. Além disso, diversas ferramentas já automatizam importantes fases de ataques distribuídos, como o controle e o sincronismo das estações [32–34]. Tal fato diminui drasticamente o nível de conhecimento necessário para se iniciar ataques distribuídos, fazendo com que atacantes inexperientes consigam inundar suas 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, esses ataques se aproveitam de determinadas vulnerabilidades para tornar seus serviços inacessíveis. Geralmente, estas vulnerabilidades são o resultado de falhas no projeto de determinada aplicação ou do próprio sistema operacional. Diversas vulnerabilidades deste gênero têm sido identificadas e exploradas ao longo dos anos. Um caso particular dos ataques por vulnerabilidades são aqueles que exigem somente um pacote para causar a negação do serviço [13–17]. Neste caso, ao receber um pacote contendo determinadas características que exploram a vulnerabilidade, a vítima se encontra em uma situação imprevista e reage diferentemente em cada caso. Como resultado, o serviço pode, por exemplo, ser indefinidamente interrompido até que uma intervenção manual ocorra. Através do envio periódico do mesmo pacote para a vítima, estes ataques podem ser prolongados até que um método de correção da vulnerabilidade seja aplicado. Estes ataques apresentam um grande desafio para sistemas de defesa, especialmente para os sistemas de rastreamento, uma vez que eles dispõem apenas de um pacote para identificar o atacante. É possível ainda que esses ataques não explorem a vulnerabilidade de uma aplicação, mas sim de um protocolo em uso. Por exemplo, Watson [35] pesquisou a possibilidade de ataques a conexões TCP

(*Transmission Control Protocol*) já estabelecidas. O ataque consiste no envio de pacotes que sinalizam o fim de conexão para uma das partes comunicantes. Estes pacotes têm o endereço IP de origem preenchido com o endereço da outra entidade de forma a fingir que a requisição de fim de conexão veio realmente dela. Desta forma, a conexão é finalizada. Uma das conseqüências deste ataque é que o protocolo de roteamento BGP (*Border Gateway Protocol*), baseado em conexões TCP de longa duração, poderia ser afetado e prejudicar todo o roteamento da Internet. Gont [16] identificou um ataque semelhante usando pacotes ICMP (*Internet Control Message Protocol*) para sinalizar erros e finalizar uma conexão em andamento.

2.2 Motivação

Existem diversos motivos usados por um atacante para se iniciar um ataque de negação de serviço direcionado a uma determinada vítima. Um motivo pode ser, por exemplo, para provar que é possível inundar a vítima ou explorar alguma de suas vulnerabilidades, de forma que seus serviços tornem-se inacessíveis. Ao observar que o ataque foi bem sucedido, o seu autor pode tentar encontrar uma solução para evitar que ataques reais aconteçam. Uma vez encontrada a solução, ela pode ser amplamente divulgada para acelerar o processo de proteção.

Outras vezes, o atacante pode receber algo em troca pelo sucesso no ataque, como o código-fonte da exploração de uma nova vulnerabilidade, o acesso a alguma máquina invadida, ou ainda um documento contendo informações confidenciais. Um acordo pode ser negociado previamente em um canal privado entre o atacante e a outra parte e acertado depois que o ataque surtir efeito. Entretanto, nem sempre informações concretas são usados como moeda de troca para motivar um atacante. Outro motivo bastante comum para iniciar um ataque é a busca pelo reconhecimento da 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. Com a automação de fases importantes de ataques de negação de serviço distribuídos, tal tarefa ficou ainda mais fácil e atacantes com pouca experiência são capazes de

deixar suas vítimas inacessíveis, ganhando a reputação desejada.

Além destes motivos, a execução de ataques de negação de serviço também pode ser motivada por dinheiro. Uma pessoa com habilidade suficiente para iniciar esses ataques pode, por exemplo, deixar um determinado servidor fora do ar por um certo período em troca de uma remuneração considerada adequada. Uma empresa poderia contratar esses "serviços" e usar esses ataques para prejudicar a imagem de seus concorrentes e se beneficiar diretamente. Recentemente, redes inteiramente formadas por computadores comprometidos, os chamados zumbis, estão sendo alugadas de forma a serem usadas em ataques de negação de serviço [36]. 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 são colocados 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 [37]. 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 rede e os servidores do seu adversário 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 [38]. Aparentemente, o ataque foi direcionado aos servidores DNS (*Domain Name System*), cuja função é converter o nome do sítio no seu respectivo endereço IP. Como conseqüência, não era possível determinar o seu endereço IP e o acesso era negado.

2.3 Ataques de Negação de Serviço

2.3.1 Ataques por Inundação

Ataques por inundação têm o objetivo de impedir o acesso a um determinado serviço através da sobrecarga de recursos da vítima. A partir deste ataque, os recursos da vítima podem ser explorados de diferentes maneiras. 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 requisição (REQ) enviada pelo atacante, t_p o tempo necessário para que a vítima processe a requisição e envie uma resposta (RESP), e t_m o tempo durante o qual um determinado recurso de memória fica alocado para atender uma requisição.

Um ataque ao processamento da vítima pode ser realizado quando o atacante consegue gerar requisições a uma taxa mais rápida do que a vítima consegue processá-las, ou seja, quando $t_a < t_p$. A Figura 2.1 mostra um ataque de negação de serviço realizado para sobrecarregar o processamento da vítima. Neste caso, a vítima não consegue processar as requisições em tempo hábil, o que faz com que a fila de requisições encha e que muitas delas sejam descartadas. Desta forma, caso um usuário legítimo tente acessar o mesmo serviço que está sendo atacado, é muito provável que sua requisição seja descartada junto com o tráfego de ataque. Isso ocorre porque o tráfego do usuário legítimo precisa disputar o mesmo recurso com as inúmeras requisições enviadas pelo atacante. É importante ressaltar ainda que 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 da requisição 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 é a memória da vítima. Dependendo do protocolo utilizado, ao receber uma requisição, o servidor aloca um espaço de memória para armazenar determinadas informações sobre o requerente. Esta alocação é geralmente empregada em protocolos que exigem a manutenção

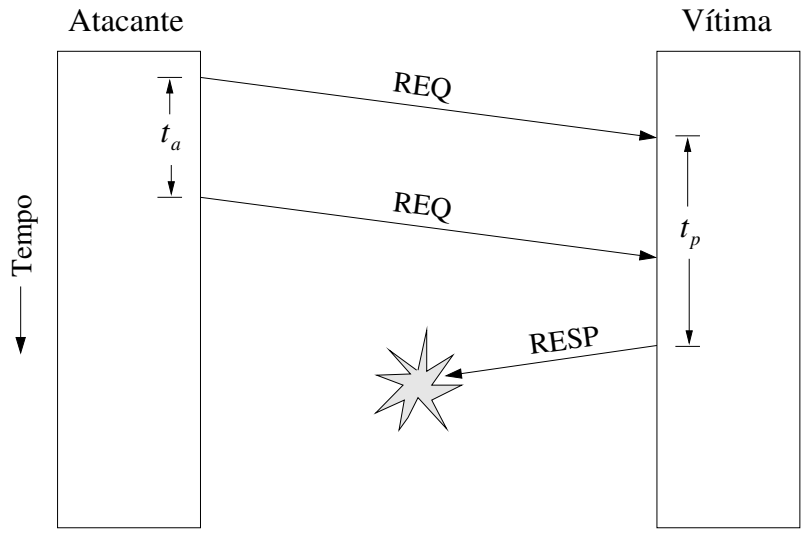


Figura 2.1: Ataque de negação de serviço consumindo o processamento da vítima.

de estados para o seu funcionamento adequado. Em seguida, uma resposta é enviada de volta para o requerente. Os recursos reservados inicialmente ficam então alocados até que um outro pacote seja retornado, ou até que um temporizador estoure. Neste caso, a memória alocada é finalmente liberada para ser aproveitada em requisições 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 consumir a sua memória. Para que este tipo de ataque tenha sucesso, é necessário que o atacante consiga gerar requisições a uma taxa maior do que a vitima consegue liberar recursos para as novas requisições, ou seja, é preciso que $t_a < t_m$. A Figura 2.2 ilustra o caso onde o recurso atacado é a memória da vítima. Nesta situação, a mesma disputa entre o tráfego legítimo e o tráfego de ataque ocorrem na vítima. À medida que uma requisição libera sua memória, uma outra requisição é processada. Devido ao volume de tráfego de ataque enviado para a vítima, muito provavelmente esta nova requisição processada não é legítima e só serve para ocupar a memória da vítima por um determinado período. Quando a memória se esgota, novas requisições não são atendidas.

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 so-

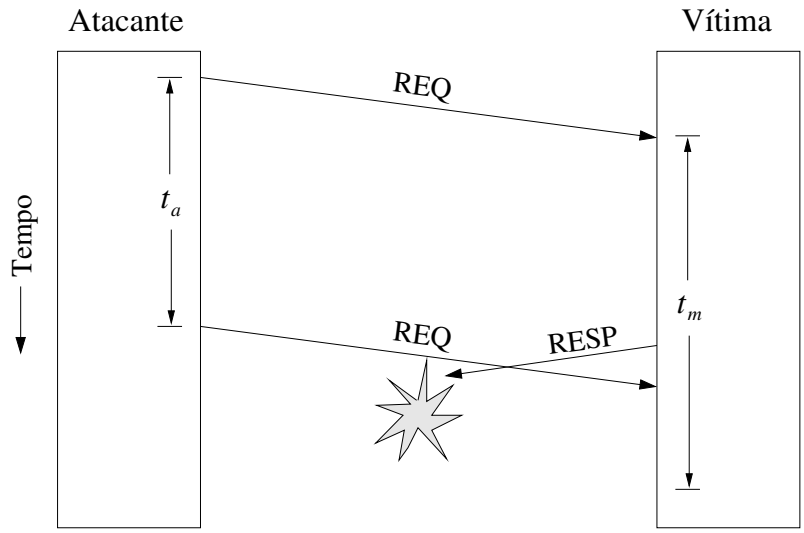


Figura 2.2: Ataque de negação de serviço consumindo a memória da vítima.

brecarregado. A segunda possibilidade ocorre quando $t_p \leq t_a < t_m$. Neste caso, a vítima consegue processar os pacotes, mas espaços de memória vão sendo consumidos por cada nova requisição e não são liberados a tempo para atender a todas as requisições. Portanto, novas requisições 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.

Um dos ataques de negação de serviço por inundação mais conhecido é o ataque 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 uma entrega confiável de dados, como a transferência de arquivos, por exemplo. Uma conexão TCP se inicia com a negociação de determinados parâmetros entre o cliente e o servidor. A Figura 2.3 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 (x). O número de seqüência é um parâmetro carregado pelos segmentos TCP que 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. Neste ponto, é dito que a conexão TCP se encontra semi-aberta. O segmento de resposta reconhece o número de seqüência do cliente e envia o número de seqüência inicial do servidor (y). 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*). Caso o atacante envie segmentos TCP SYN de forma que $t_a < t_p$, o processamento da vítima é afetado. Se por outro lado o tempo entre cada segmento de ataque está dentro do intervalo $t_p \leq t_a < t_m$, a memória da vítima é sobrecarregada por diversas conexões semi-abertas e o serviço é negado.

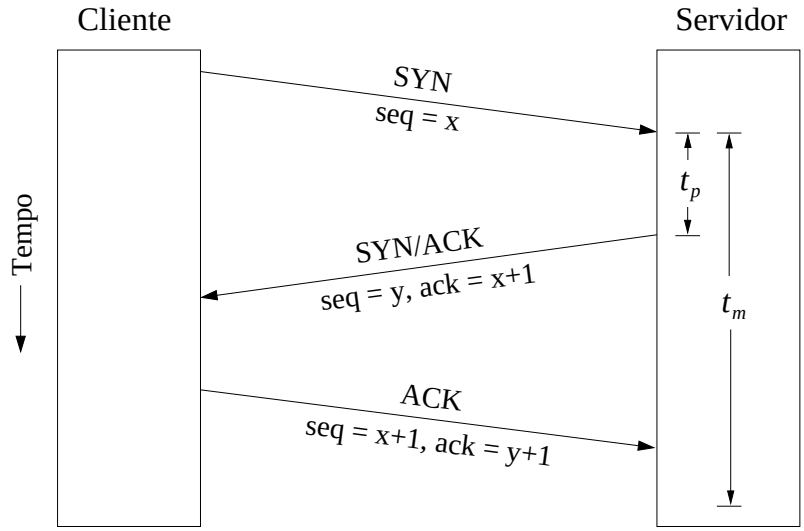


Figura 2.3: Abertura de uma conexão TCP.

Uma solução proposta por Schuba *et al.* [39] 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, este 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.

Outro ataque ao protocolo TCP pode ser realizado se o atacante está na mesma rede que a vítima e tem acesso aos pacotes enviados por ela. A idéia é esgotar a memória da vítima através da abertura de diversas conexões simultâneas aparentemente legítimas. Ao contrário dos ataques detalhados anteriormente, o processo de abertura de conexão é finalizado neste caso. De forma a que somente os recursos da vítima sejam esgotados, é necessário que o atacante não aloque recursos para cada conexão aberta. Na verdade, o atacante envia diversos pacotes TCP SYN para a vítima, com o endereço de origem forjado. Como ele consegue observar os segmentos TCP SYN/ACK respondidos pela vítima, ele constrói segmentos TCP ACK a partir destes segmentos e os reenvia para a vítima de forma a finalizar o acordo de três vias. Desta forma, conexões vão sendo abertas até esgotar o limite máximo de conexões abertas simultaneamente. Neste ponto, qualquer novo pedido de abertura de conexão é negado. Este ataque é denominado Naptha [40].

Um outro tipo de ataque de serviço por inundação que visa consumir a memória da vítima pode ser realizado com o uso de fragmentos 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 o 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. O ataque é conduzido da seguinte forma. 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. 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 ainda afetar o seu processamento. Dependendo da freqüência com que esses fragmentos são enviados para a vítima, a condição para a negação do seu serviço pode ser atingida.

2.3.2 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 à 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 ataque envie algum tipo de requisição (REQ) para o refletor, usando como endereço de origem o próprio endereço da vítima. Ao receber uma requisição, o refletor não consegue verificar a sua autenticidade e, conseqüentemente, envia uma resposta (RESP) diretamente para a vítima. A Figura 2.4 mostra este procedimento de maneira sucinta. 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 é sobrecarregado e o tráfego excedente é descartado, não apresentando efeito na vítima. Portanto, o ideal para o atacante é gerar os pacotes de ataque de acordo com a desigualdade $t_r \leq t_a < t_p$.

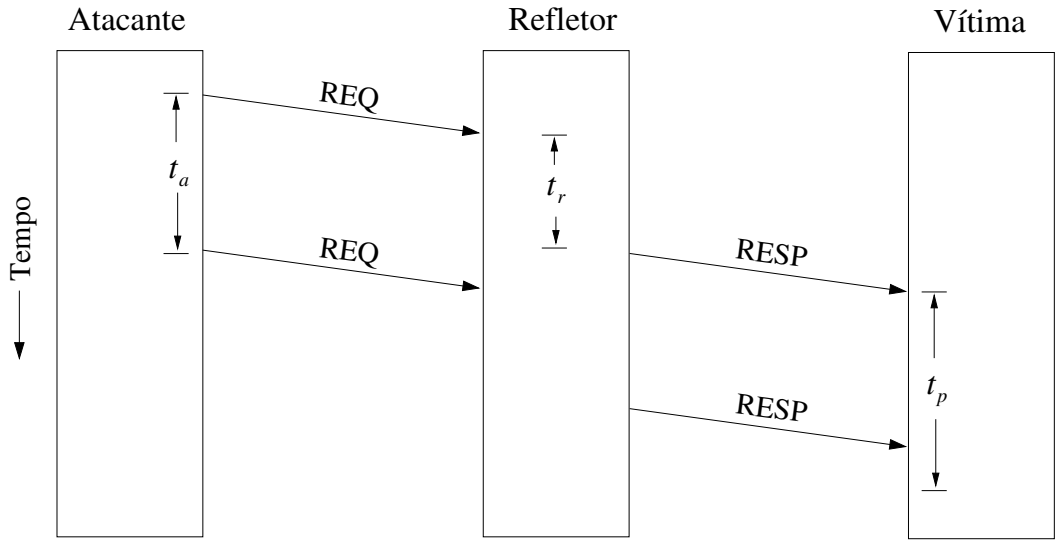


Figura 2.4: Ataque de negação de serviço por um refletor.

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 atenda a algum tipo de requisição e envie uma resposta. 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*) para este ataque. Uma aplicação que utiliza o UDP como protocolo de transporte pode ser usada para enganar o refletor. Por exemplo, 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 a resposta enviada pelo refletor é maior que a requisição enviada pelo atacante. 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 [3]. Neste caso, pacotes ICMP são enviados para o endereço IP de difusão de uma determinada rede. 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 pacote. Outro exemplo que pode ser aproveitado para amplificação do tráfego de ataque é o serviço de geração de caracteres (*chargen*) [41], transportado pelo UDP. 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.

2.3.3 Ataques à Infra-estrutura de Rede

Ataques de negação de serviço muitas vezes são direcionados a sítios conhecidos na Internet. Desta forma, os seus autores conseguem uma fama maior como resultado do ataque. Nestes casos, é provável que a própria vítima seja superdimensionada, ou seja, possua recursos como processamento e memória em abundância. Como conseqüência, ataques distribuídos de pequena escala não conseguem consumir estes 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 crucial para o funcionamento do serviço que não dependa da vítima. Um exemplo é tentar consumir toda a banda passante da rede da vítima com o tráfego de ataque. Desta forma, por mais que a vítima consiga atender a todas as requisições que chegam a ela, muitas requisições ainda são 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. Estes ataques são difíceis de combater, já que os pacotes não precisam ter nenhum padrão 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 [42]. 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 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 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.

2.3.4 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 na implementação do protocolo TCP em sistemas operacionais Windows recentemente [43]. 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. O ataque consistia no envio de diversos fragmentos IP pertencentes ao mesmo pacote com números de seqüência que se sobrepunham [14]. Desta forma, a vítima recebia uns poucos fragmentos intencionalmente mal formados e entrava em um estado imprevisto, resultando no seu congelamento ou reinicialização.

Gont [16] 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.

2.4 Ataques Distribuídos

Ataques de negação de serviço distribuídos são geralmente usados em ataques por inundação, quando uma estação sozinha não é capaz de consumir completamente 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 nomes como 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 a cada dia, é possível encontrar uma série de estações com versões ultrapassadas que possuem determinadas vulnerabilidades. Algumas dessas vulnerabilidades, por exemplo, 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 conseguir privilégios de administrador, obtendo controle total da estação invadida. Outra possibilidade que vem sendo muito usada atualmente é adivinhar a senha de uma determinada conta do sistema através 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 [30], 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 automatização. Por isso, usuários maliciosos criaram ferramentas capazes de 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 2.5 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 em 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 na Seção 2.3, 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.

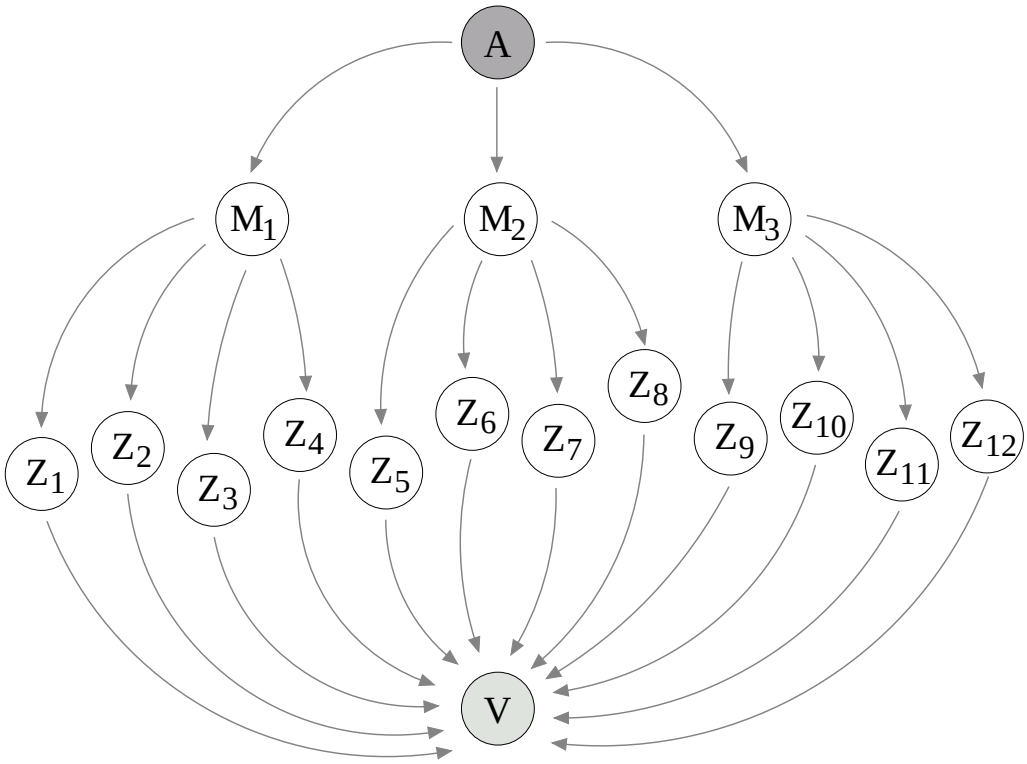


Figura 2.5: Ataque de negação de serviço distribuído.

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 camadas existirem nesta hierarquia, mais protegido estará o atacante. A Figura 2.6 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. Esse protocolo permite a troca de mensagens entre entidades remotas a partir de um servidor central. 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 [12].

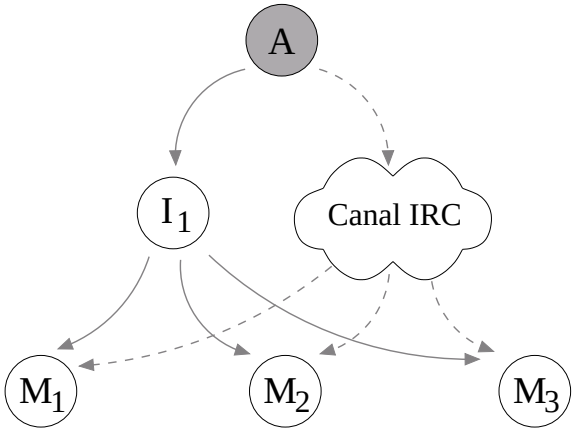


Figura 2.6: Estação intermediária ou um canal IRC para controlar as estações-mestre.

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. 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.

A topologia da Internet é formada por nós localizados no núcleo, composto por enlaces com altas taxas de transmissão, e por nós nas bordas, geralmente providos com enlaces de baixas taxas que se conectam ao núcleo. Esta assimetria permite que os nós localizados no núcleo encaminhem pacotes de diversas origens distintas para diferentes destinos a altas velocidades. Como resultado, ela também possibilita que nós localizados

nas bordas sejam inundados pelo tráfego agregado de outros nós. Desta forma, 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.

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 contactar 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 cessar a atividade de uma estação-mestre diretamente. Porém, é preciso antes identificar estas 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. A partir deste zumbi, tenta-se 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 da estação-mestre ou do zumbi.

2.5 Endereços IP de Origem Forjados

Uma das maneiras de garantir o anonimato em ataques de negação de serviço é através do uso de endereços IP de origem forjados. Em comunicações convencionais, pacotes IP são trocados entre duas entidades que desejam compartilhar determinadas informações. De forma a permitir uma comunicação bidirecional, tanto o endereço de origem como o endereço de destino são necessários em cada pacote de dados. Entretanto, 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 ataques por vulnerabilidades, atacantes podem forjar o endereço IP de origem e enviar estes pacotes para a vítima sem deixar nenhum rastro que os identifique.

Embora dificulte muito a identificação do atacante, o uso de endereços de origem forjados não é necessário para o sucesso de ataques de negação de serviço. A vítima pode ser inundada com pacotes com endereços legítimos e ter algum recurso sobrecarregado. Em ataques por vulnerabilidade, pacotes com endereços de origem reais também podem ser usados para atingir o mesmo resultado. Entretanto, a partir do endereço IP real do atacante, é possível determinar a sua localização geográfica e interromper o ataque em andamento.

2.5.1 Objetivos

Ataques de negação de serviço utilizam endereços forjados com objetivos diferentes [30]. Obviamente, o motivo mais comum do uso de endereços forjados é para esconder a localização do atacante ou dos zumbis. Em ataques de negação de serviço simples, formados por um único atacante que gera tráfego diretamente para a vítima, estes endereços são usados para que a vítima não consiga identificar o atacante a partir da análise do tráfego recebido. Desta forma, a vítima reconhece que está sendo atacada, mas não consegue distinguir o responsável pelo ataque. Em ataques distribuídos, endereços de origem forjados podem ser usados para proteger a identidade dos zumbis. Caso algum zumbi seja identificado, é possível encontrar a estação-mestre que o controla remotamente. A partir da estação-mestre, novos esforços podem ser realizados para identificar o próprio atacante. Portanto, o envio de pacotes com endereço de origem legítimo pode comprometer a identidade do atacante.

Com um objetivo distinto, endereços de origem forjados também são usados em ataques por refletores. Neste caso, o uso de endereços forjados não é usado somente para garantir o anonimato do atacante, mas também para garantir que o refletor encaminhe o pacote de resposta para a vítima. É importante notar que o uso de endereços forjados é obrigatório neste ataque.

Um outro motivo para empregar endereços de origem forjados é para contornar determinados métodos de defesa. Um exemplo típico de um método de defesa empregado é filtrar todos os pacotes destinados a uma aplicação que estejam fora de um conjunto de endereços de origem pré-estabelecidos. Através do uso de pacotes contendo endereços de origem dentro deste conjunto especificado, é possível passar pelo filtro e atingir a aplicação desejada. Entretanto, neste caso, é necessário que o atacante tenha algum conhecimento prévio dos endereços liberados pelo filtro de forma a enviar pacotes contendo somente estes endereços.

2.5.2 Escolha de Endereços

Com relação à escolha do endereço de origem usado nos pacotes de ataque, técnicas distintas são empregadas pelos atacantes ou pelos zumbis [44]. A técnica mais simples consiste em gerar números aleatórios de 32 bits e usá-los como endereços de origem nos pacotes usados para o ataque. Neste caso, isto é equivalente a atribuir uma probabilidade de 2^{-32} a cada um dos endereços IP existentes e escolher um aleatoriamente para cada pacote enviado. Desta forma, não é possível filtrar os pacotes de ataque a partir do endereço de origem. Apesar de simples, esta técnica pode gerar endereços IP usados somente em redes privadas, endereços de difusão, endereços de *multicast* ou ainda endereços inválidos. Estes endereços podem ser filtrados sem dificuldade, uma vez que são conhecidos. Entretanto, grande parte dos pacotes gerados alcança a vítima sem maiores problemas.

Uma outra técnica possível é escolher endereços da própria sub-rede para o endereço de origem dos pacotes de ataque. Muitas vezes, roteadores de saída descartam pacotes cujo endereço de origem não pertence a faixa de endereços da sub-rede de forma a impossibilitar a saída de pacotes forjados. Esta técnica é denominada filtragem de ingresso

e é explicada com detalhes mais adiante na Subseção 2.5.3. De forma a contornar esta medida preventiva, zumbis e atacantes podem escolher endereços de origem de outras estações dentro da própria sub-rede. Esta estratégia possibilita a saída de pacotes com endereços de origem forjados e ainda mantém o anonimato durante o ataque. Uma vez tendo sido originados e possuindo endereços válidos da sub-rede, esses pacotes forjados não podem mais ser diferenciados de pacotes legítimos ao longo do trajeto até a vítima.

Endereços válidos de determinadas estações também podem ser usados nos pacotes de ataque com dois objetivos distintos. Primeiramente, estes endereços podem ser usados para que, através de contramedidas tomadas pela vítima, o serviço seja efetivamente negado às estações que usam estes endereços legitimamente. Basicamente, o atacante inunda a vítima usando um determinado grupo de endereços nos pacotes de ataque. A vítima então começa a filtrar todo o tráfego originado por este grupo até que o ataque termine. Neste caso, por mais que as estações que empregam estes endereços legitimamente sejam inocentes, seu tráfego sofrerá com a filtragem e o serviço lhes será negado. Um segundo motivo para o uso de um endereço válido é no caso de ataques por refletores.

2.5.3 Medidas Preventivas

Uma técnica preventiva, denominada filtragem de ingresso (*ingress filtering*) [45], foi sugerida para evitar que pacotes com endereços forjados trafeguem na rede. 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 2.7 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 na interface da Rede 1 do roteador R_1, somente aqueles pacotes cujo endereço de origem pertence à 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 en-

dereço 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.

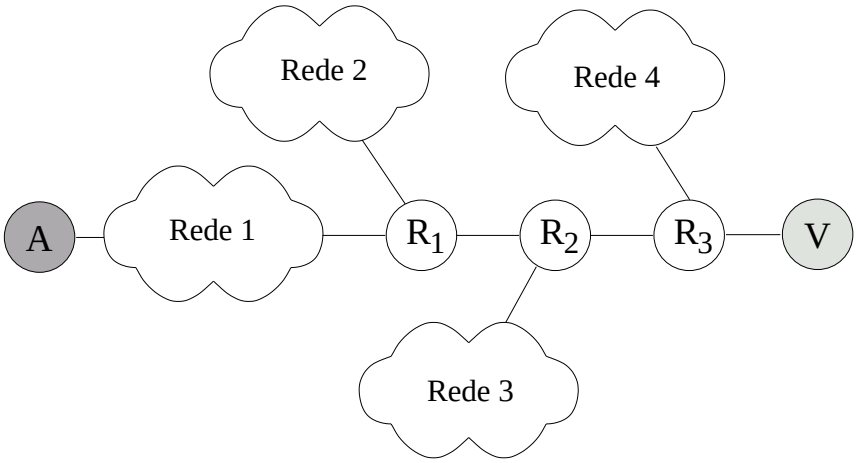


Figura 2.7: Filtragem de ingresso.

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 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 possui. Existem ainda tecnologias que podem ser afetadas pela filtragem de ingresso, como o IP Móvel [46], pois utilizam legitimamente pacotes com endereços de origem "forjados."

Uma generalização da filtragem de ingresso foi proposta por Li *et al.* [47]. 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 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 a 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.

2.6 Aplicação do Rastreamento de Pacotes

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.

Nos ataques de negação de serviço descritos nas Seções 2.3 e 2.4, o rastreamento de pacotes é útil em diversas situações. Em ataques que utilizam pacotes com endereço de origem forjado, a vítima não consegue extrair nenhuma informação relevante dos pacotes de ataque que forneça alguma pista sobre a origem dos atacantes. O rastreamento pode então ser usado neste caso para, a partir dos pacotes recebidos pela vítima, determinar a verdadeira fonte do tráfego de ataque.

Um outro tipo de ataque onde o rastreamento de pacotes é fundamental é aquele realizado através de um refletor. Neste caso, os pacotes de ataque recebidos pela vítima possuem o endereço de origem do refletor. Embora o endereço da origem do ataque possa ser legítimo, o rastreamento empregado a partir do refletor permite determinar a verdadeira fonte das requisições enviadas e a identificação do atacante. Desta forma, a vítima consegue identificar o refletor a partir dos pacotes de ataque recebidos e, a partir do refletor, é possível encontrar o atacante.

Em ataques distribuídos, o atacante pode configurar os zumbis para que eles enviem somente pacotes com endereços de origem forjados. Desta forma, a identidade dos zumbis é protegida e fica ainda mais difícil identificar o próprio atacante. Entretanto, com o uso do rastreamento, a identificação dos zumbis pode ser realizada a partir da vítima. Caso os zumbis usem ainda refletores para atacar a vítima, estes refletores precisam ser usados como ponto de partida do processo de rastreamento. A partir das requisições enviadas aos refletores, algum zumbi pode ser identificado e usado para encontrar a estação-mestre. O mesmo processo pode então ser empregado sucessivamente em cada camada da hierarquia até a identificação final do próprio atacante.

O rastreamento de pacotes é um primeiro passo importante e necessário contra ataques de negação de serviço, pois procura identificar os criminosos que iniciam estes ataques. A partir desta identificação, outras medidas devem tomadas para que o ataque possa ser finalmente interrompido e os seus autores punidos.

Capítulo 3

O Sistema de Rastreamento Proposto

NESTE capítulo, é introduzido o sistema de rastreamento de pacotes IP proposto neste trabalho. Esse sistema é motivado pela crescente ação de ataques anônimos de negação de serviço, que visam impossibilitar o uso de um determinado serviço sem deixar rastros. O sistema proposto é projetado para possibilitar o rastreamento de cada pacote recebido pela vítima. Desta forma, mesmo ataques por vulnerabilidade constituídos por um único pacote conseguem ter a sua origem revelada. Além disso, ataques distribuídos podem ser mais facilmente interrompidos, uma vez que todas as fontes de tráfego podem ser identificadas. Até o momento, o rastreamento de ataques a partir de um único pacote só é realizado através do armazenamento de informações de auditoria nos roteadores [48]. Entretanto, o método empregado no sistema proposto não armazena nenhum estado na infra-estrutura de rede.

Primeiramente, conceitos importantes como a definição do rastreamento de pacotes IP e os trabalhos relacionados são detalhados. Em seguida, as suposições realizadas neste trabalho são abordadas e, por fim, o sistema proposto é apresentado.

3.1 O Rastreamento de Pacotes IP

O rastreamento de pacotes IP tem o objetivo final de determinar a verdadeira origem e a rota percorrida por um pacote ou um conjunto de pacotes IP. Ou seja, a partir de pa-

cotes recebidos pela vítima, deseja-se encontrar quem realmente os enviou. Uma maneira interessante de se realizar este rastreamento é através da marcação do pacote por cada roteador atravessado. Desta forma, cada roteador insere uma marca nos pacotes roteados de forma a notificar a vítima sobre sua presença na rota de ataque. Em posse das informações recebidas dentro dos pacotes, a vítima pode reconstruir a rota percorrida até a sua origem.

3.1.1 Definições

A Figura 3.1 mostra um exemplo de rede, do modo como é vista pela vítima V. Cada atacante A_i é visto como um nó que gera tráfego em direção à vítima, conforme mostrado pela linha tracejada. Os nós internos R_i representam roteadores ao longo do caminho entre um determinado atacante e a vítima. Uma rota de ataque é definida como uma lista ordenada de roteadores entre um atacante A_i e a vítima V que descreve o caminho percorrido pelos pacotes enviados por A_i. Na Figura 3.1, por exemplo, a rota de ataque de A_2 é representada por (R_5, R_2, R_1). Através da composição de todas as rotas de ataque provenientes de cada atacante, um grafo de ataque é construído.

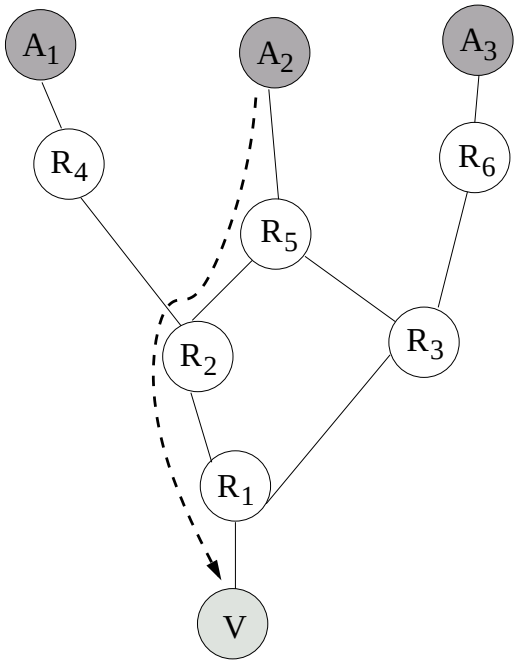


Figura 3.1: Uma rede conforme a visão da vítima.

A partir destes conceitos, Savage *et al.* [20] introduziram o chamado problema do rastreamento. De acordo com a definição, o problema do rastreamento "perfeito" é determinar a rota real de ataque para cada um dos atacantes. Entretanto, este problema não é facilmente resolvido e, portanto, um problema mais restrito foi formulado. O problema do rastreamento "aproximado" é definido como a determinação de uma rota que contenha a rota real de ataque como um sufixo. Por exemplo, (R_6, R_5, R_2, R_1) é uma solução válida para o problema do rastreamento aproximado, uma vez que contém a rota de ataque real (R_5, R_2, R_1) como um sufixo. Além disso, a solução é definida como robusta se não existe maneira do atacante evitar que a vítima encontre uma rota que possua a rota real de ataque como um sufixo.

Sistemas de rastreamento baseados na marcação de pacotes possuem dois procedimentos básicos. Um primeiro procedimento, denominado procedimento de marcação de pacotes, é realizado pelos roteadores durante a travessia dos pacotes. Cada roteador simplesmente insere algumas informações sobre si mesmo nos pacotes roteados. O segundo procedimento, denominado procedimento de reconstrução de rota, é realizado pela vítima em conjunto com os roteadores. O seu objetivo é encontrar a rota de ataque percorrida pelos pacotes a partir das informações recebidas.

3.1.2 Trabalhos Relacionados

Existem basicamente duas abordagens distintas para o rastreamento de pacotes. A primeira abordagem visa identificar os atacantes sem o armazenamento de estado. Por outro lado, a segunda abordagem propõe a coleta de rastros de auditoria durante o ataque de forma a permitir que o rastreamento seja realizado após o ataque, entre outras vantagens.

Rastreamento Sem Estado

Uma estratégia intuitiva para rastrear a fonte de um ataque em andamento é chamada depuração de entrada (*input debugging*) [49]. Ela consiste primeiramente na identificação de uma assinatura do ataque, que é alguma característica apresentada em todos os seus pacotes. A partir da assinatura, operadores da rede podem observar o roteador mais

próximo da vítima para identificar a interface de entrada dos pacotes de ataque e, conseqüentemente, o roteador adjacente que os enviou. Esta consulta, feita recursivamente em cada roteador, revela a rota do ataque até o próprio autor ou até a borda de outro provedor de serviços. Neste caso, este segundo provedor necessita ser contatado para dar continuidade ao processo. Embora simples, essa aplicação depende da comunicação entre provedores de serviço para alcançar um sucesso no rastreamento. Além disso, o ataque precisa ser suficientemente longo para permitir o rastreamento completo, não sendo possível realizá-lo após o término do ataque. Um sistema denominado *CenterTrack* [49] emprega esta técnica através de túneis. Depois de detectado o ataque, todo o tráfego da vítima é redirecionado de forma a passar por um roteador central de rastreamento com o qual outros roteadores já possuem túneis pré-estabelecidos. A partir deste roteador, a depuração de entrada é usada visando descobrir por qual túnel o pacote chegou.

Sob uma outra abordagem, uma técnica de rastreamento baseada em testes de enlaces foi desenvolvida por Burch e Cheswick [50] . Seu funcionamento é baseado na observação do fluxo de ataque recebido pela vítima à medida que cada enlace é inundado por curtas rajadas de tráfego. Inicialmente, todos os enlaces do roteador mais próximo da vítima são testados. Aquele que, quando inundado, causa uma queda no recebimento do tráfego de ataque é selecionado como componente da rota de ataque. O processo é então repetido recursivamente em cada roteador até que a fonte seja identificada. Apesar de não necessitar de nenhuma intervenção por parte dos operadores da rede, esta técnica exige um mapa topológico apurado assim como possíveis nós dispostos a sofrerem curtas inundações. Além disso, todo o sistema de rastreamento é baseado no serviço de geração de caracteres (*chargen*), o qual por padrão vem desabilitado na maioria dos roteadores [51]. Como na técnica anterior, é possível que as verdadeiras fontes de um ataque nem cheguem a ser reveladas se o ataque for subitamente interrompido no meio do processo de rastreamento. Além destas características, a variação observada na vítima para o caso de ataques distribuídos ou ainda para fluxos curtos pode ser imperceptível.

Rastreamento Baseado em Auditoria

A vantagem das abordagens descritas até o momento reside na absoluta ausência de estados. Entretanto, a sua conseqüência direta é a impossibilidade de rastrear a origem de um ataque após o seu término. Além disso, estas propostas lidam apenas com ataques baseados em inundação e não ataques de um único pacote. Por isso, uma segunda abordagem defende a criação de informações de auditoria à medida que os pacotes atravessam a rede. Estas informações podem ser consultadas mais tarde visando reconstruir a rota de ataque. Existem propostas que sugerem a sua localização tanto na vítima quanto nos próprios roteadores.

Um método simples para rastrear a verdadeira origem de um pacote é adicionar o endereço IP de cada roteador ao fim do pacote à medida que ele atravessa a rede. A idéia é bastante semelhante à opção IP Record Route [52], porém sem as suas restrições de espaço. Desta forma, cada pacote recebido pela vítima apresenta toda a rota de ataque embutida em si. Este algoritmo consegue rastrear a origem do ataque a partir de um único pacote, apresentando a alta escalabilidade necessária em ataques distribuídos. Entretanto, algumas desvantagens o desqualificam como um algoritmo de rastreamento ideal [20]. Em primeiro lugar, o ato de adicionar dados ao pacote durante o roteamento é custoso e implica um processamento adicional. Além disso, o tamanho do pacote varia ao longo da rota, uma vez que cada roteador acrescenta seu endereço IP ao reencaminhá-lo. Como não há como garantir espaço suficiente para todos os endereços da rota de ataque, pacotes sendo rastreados podem sofrer fragmentações desnecessárias e causar um maior processamento aos roteadores.

Savage *et al.* [20] introduzem um sistema baseado em auditoria, onde a informação necessária para o rastreamento é encontrada na vítima. Os roteadores a notificam sobre sua presença na rota inserindo informações nos pacotes roteados. Visando reduzir o processamento no roteador e o espaço necessário em cada pacote, técnicas de amostragem e codificação são empregadas. Desta forma, cada roteador probabilisticamente insere informações parciais sobre si mesmo no pacote roteado para a vítima. Posteriormente, a vítima consegue reconstruir a rota se um número suficiente de pacotes de um mesmo fluxo de ataque for recebido. Embora inovadora, a proposta necessita de um alto poder

computacional durante a reconstrução da rota de ataque pela vítima e gera diversos falsos positivos mesmo em ataques distribuídos de pequeno porte [53]. Park *et al.* [54] mostram ainda uma vulnerabilidade interessante deste sistema. O atacante pode inserir informações falsas no campo de marcação dos pacotes de forma a criar falsos rastros para levar a vítima a crer que o ataque partiu de outra origem. Com alta probabilidade, esta informação chega à vítima intacta. Por exemplo, usando a probabilidade de marcação de pacotes de $0,04$ sugerida pelos autores e uma rota de ataque típica de 15 roteadores, a probabilidade da vítima receber um pacote marcado somente pelo atacante é aproximadamente $(1-0,04)^{15} \approx 54\%$, que é um valor muito elevado.

Outro método baseado em auditoria foi proposto por Bellovin *et al.* [21]. 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, o atacante pode enviar pacotes ICMP forjados e, com isto, inviabilizar a descoberta da rota real de ataque. Logo, a autenticação das mensagens é necessária de forma a evitar mensagens forjadas pelo atacante e a adoção de uma infra-estrutura de chave pública (*Public Key Infrastructure* - PKI) se torna inevitável. Em uma extensão desse trabalho, Mankin *et al.* [55] introduzem novos conceitos como "utilidade" e "valor" das mensagens de rastreamento, assim como uma proposta para melhorá-los.

Outro sistema baseado em auditoria proposto por Dean *et al.* [22] se serve de técnicas probabilísticas e algébricas em conjunto para o rastreamento de pacotes IP. A idéia é cada pacote trazer consigo uma representação da rota percorrida como o resultado de um polinômio conhecido, cujas incógnitas são os endereços dos roteadores. Desta forma, tendo a vítima recebido pacotes suficientes da mesma rota, um sistema de equações pode ser construído e resolvido com solução única. Entretanto, diferente do sistema proposto por Savage *et al.* [20], 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. Além disso, sendo os algoritmos propostos baseados em marcações probabilísticas, a mesma vulnerabilidade do atacante

burlar o sistema citada anteriormente se aplica nesta situação [54].

Sob a perspectiva de armazenamento das informações de auditoria na própria infra-estrutura de rede, a maneira mais simples de recolher rastros é cada roteador registrar todos os pacotes que o atravessam, como o proposto por Stone [49]. Porém, recursos excessivos são requisitados tanto para armazenagem quanto para a mineração dos dados. Por exemplo, o armazenamento de cada pacote de um enlace OC-24 (1,244 Gbps) saturado necessitaria de 9,3 GB por minuto ou, equivalentemente, 13,4 TB por dia. 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 [29] (Capítulo 4). Ultimamente, estes filtros têm sido amplamente usados em redes de computadores [56]. Snoeren *et al.* [48] propõem 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. 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 recursivo 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. Melhorias propostas por Li *et al.* [57] diminuem drasticamente o espaço necessário para o armazenamento embora a capacidade de se rastrear um único pacote seja comprometida.

O sistema de rastreamento proposto neste trabalho consegue rastrear a origem do ataque a partir de um único pacote sem armazenar estado na infra-estrutura da rede. Até o momento, estas duas vantagens não foram apresentadas juntas em nenhum sistema de rastreamento. Ao invés de usar um Filtro de Bloom nos roteadores para armazenar os pacotes roteados como feito por Snoeren *et al.* [48], o sistema proposto usa um Filtro de Bloom nos pacotes para armazenar os roteadores atravessados. Desta forma, com alguns bits adicionais por pacote, é possível determinar a rota percorrida por cada pacote.

3.2 Premissas

Algumas suposições são realizadas antes do projeto do sistema de rastreamento de forma a estabelecer princípios práticos e restrições:

1. ataques podem ser constituídos de um único pacote;
2. atacantes podem gerar pacotes com conteúdo arbitrário;
3. atacantes sabem que estão sendo rastreados;
4. atacantes podem agir em conjunto;
5. roteadores podem ser comprometidos, mas não freqüentemente;
6. roteadores possuem recursos escassos;
7. o tamanho do pacote não deve aumentar à medida que atravessa a rede.

Ataques de negação de serviço constituídos pelo envio de um único pacote já existiram no passado [13, 14] e alguns ainda existem até o momento atual [15–17]. Estes ataques comprovam a veracidade da primeira suposição e alertam para o perigo de novos ataques do mesmo gênero. Ao invés de inundar a vítima com diversas requisições de serviço falsas, estes ataques são baseados em vulnerabilidades ativadas por um pacote com determinadas características. Portanto, é mais fácil negar o serviço da vítima com um ataque deste tipo do que com ataques por inundação. É também mais difícil rastrear a sua origem, uma vez que a maioria dos sistemas de rastreamento propostos somente lida com ataques baseados em um fluxo de pacotes [20–22, 49, 50]. Além disso, ataques distribuídos podem ser conduzidos através do envio de um só pacote. Com um grande número de atacantes, a negação do serviço pode ser atingida se cada atacante envia um pacote para a vítima em um determinado período de tempo. O tráfego agregado é então responsável por desabilitar os serviços providos pela vítima. Desta forma, sistemas de rastreamento ideais devem ser capazaes de rastrear ataques de um único pacote. Até o momento, sistemas com esta característica exigem um alta capacidade de armazenamento na infra-estrutura de rede [48, 49].

A Suposição 2 reflete a habilidade de atacantes experientes. Estes atacantes conseguem injetar na rede pacotes IP com conteúdo arbitrário. Desta forma, determinados campos de controle carregados pelos cabeçalhos dos protocolos, como endereços e números de seqüência, podem ser alterados de acordo com a vontade do atacante. Não há nenhum mecanismo que impeça a construção e o envio destes pacotes na rede. Como conseqüência, sistemas de rastreamento não podem ser projetados baseando-se na hipótese de que os pacotes apresentam um determinado conteúdo inicial.

A Suposição 3 garante que o projeto do sistema deve ser seguro o suficiente para garantir que, mesmo sabendo que estão sendo rastreados, os atacantes não tenham como se evadir do sistema. Além disso, sistemas de rastreamento devem ser totalmente abertos para atingir uma implementação em escala global. Desta forma, os atacantes não só sabem que estão sendo rastreados, mas também sabem como estão sendo rastreados. Mesmo assim, o sistema deve ser eficaz a ponto de rastrear a origem de qualquer ataque.

Atacantes com objetivos em comum podem cooperar em um ataque distribuído para alcançar melhores resultados, conforme estabelece a Suposição 4. Neste tipo de ataque, um atacante sozinho não é capaz de consumir os recursos da vítima. Portanto, vários atacantes são escalados para inundar a vítima. Mesmo sem a cooperação direta de vários atacantes distintos, estações podem ser comprometidas por um mesmo atacante de forma a serem usadas em um ataque distribuído.

Um atacante pode ganhar o acesso a roteadores de diversas maneiras diferentes, mas é assumido que este evento não é freqüente, de acordo com a Suposição 5. Entretanto, caso isso aconteça, este roteador também é tratado como um atacante, uma vez que pode alterar as marcações realizadas pelos roteadores anteriores. De qualquer forma, a invasão deve ser imediatamente remediada após a sua detecção de forma a possibilitar o rastreamento completo da rota de ataque.

A Suposição 6 lida com as restrições dos roteadores. É assumido que a infra-estrutura de rede tem recursos limitados e, portanto, é incapaz de manter estado para cada pacote. Na verdade, no sistema de rastreamento proposto, a atividade de armazenamento é deixada como uma opção para a vítima enquanto somente um pequeno processamento adicional é incluído no roteamento.

A suposição final de que o tamanho dos pacotes não deve aumentar durante a travessia pela rede é necessária para evitar a fragmentação. A fragmentação de pacotes afeta o desempenho da rede porque exige processamento tanto do roteador que realiza a fragmentação quanto dos roteadores subseqüentes que encaminham os pacotes adicionais. Medidas realizadas indicam que uma causa significativa da fragmentação de pacotes é a inserção do cabeçalho adicional de 40 bytes usado em túneis [58]. Portanto, sistemas de rastreamento ideais não podem aumentar o tamanho do pacote a cada salto.

3.3 O Sistema de Rastreamento

Esta seção apresenta brevemente o sistema de rastreamento de pacotes IP proposto. 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 é proposta para evitar que o atacante possa forjar as "assinaturas" dos roteadores e prejudicar o rastreamento.

O Filtro de Bloom [29] é uma estrutura de dados usada para representar um conjunto de elementos de forma compacta. Ele é abordado com mais detalhes no Capítulo 4, mas uma visão geral do seu funcionamento é importante para o entendimento da proposta. O filtro é composto por um vetor de bits e por um determinado número de funções *hash* independentes. Primeiramente, o vetor de bits é inteiramente zerado e, em seguida, os elementos do conjunto são inseridos no filtro. Para isso, cada elemento é usado como

entrada para as funções *hash*. Os resultados destas funções são então usados como índices no vetor de bits e cada bit indicado é então preenchido com 1. Após as inserções de todos os elementos, testes de pertinência podem ser conduzidos para verificar se um determinado elemento pertence ou não ao conjunto. Para isso, o elemento em questão é usado como entrada para as mesmas funções *hash* e verifica-se se os bits indicados estão preenchidos com 1. Se pelo menos um bit encontra-se zerado, então com certeza o elemento não pertence ao conjunto. Caso contrário, é assumido que o elemento pertence ao conjunto. Entretanto, existe uma pequena probabilidade de que esta suposição esteja errada e de que o elemento seja na verdade um falso positivo. Este evento ocorre quando todos os bits indicados foram previamente preenchidos com 1 por outros elementos realmente inseridos. Embora apresente esta desvantagem, o Filtro de Bloom pode ser bem eficiente para a representação de um conjunto se a probabilidade de falso positivo for mantida suficientemente baixa.

3.3.1 O Procedimento de Marcação de Pacotes

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, no próprio pacote, de um Filtro de Bloom que permite identificar os endereços de todos os roteadores da rota de ataque. Uma vantagem importante desse procedimento de marcação é o seu baixo processamento adicional. Ao encaminhar um pacote, cada roteador atualiza o filtro contido no pacote com resultado de uma simples operação OU bit-a-bit do próprio filtro do pacote com um registrador da interface de saída do roteador. É importante observar que os valores *hash* do endereço IP de cada interface do roteador são calculados inicialmente e armazenados em um registrador existente para cada interface. Este cálculo é feito uma única vez e não pacote a pacote. Esse registrador pode ser interpretado como um Filtro de Bloom com somente um elemento inserido: o endereço IP da interface. Dessa forma, o endereço IP da interface de saída do roteador é adicionado ao filtro do pacote de maneira eficiente.

3.3.2 O Procedimento de Reconstrução de Rota

Para reconstruir a rota de ataque, o seguinte algoritmo é utilizado. Inicialmente, a vítima verifica qual dos seus roteadores vizinhos está presente 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, a vítima envia o filtro para o roteador vizinho 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 por onde veio o ataque e, como conseqüência, o próximo roteador 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 3.2 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.

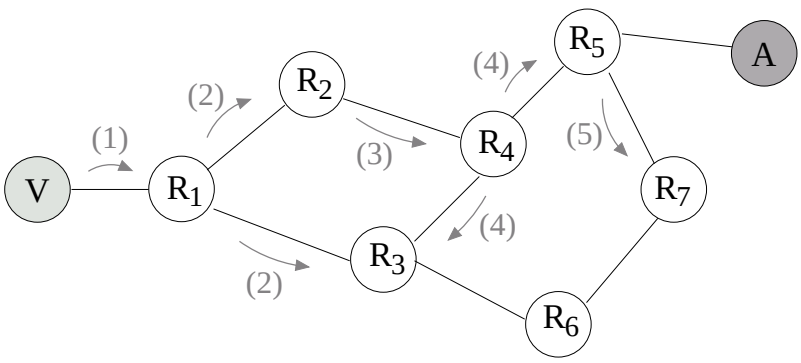


Figura 3.2: Exemplo do procedimento de reconstrução de rota.

3.3.3 Vantagens e Desvantagens

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 ou em um ataque por vulnerabilidade. 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 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 algumas desvantagens presentes em outros sistemas de rastreamento [20–22, 48, 53]. 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 roteadores 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 *hash* 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. Entretanto, um procedimento aprimorado de reconstrução de rota pode ser utilizado para eliminar os falsos negativos. No Capítulo 4, a idéia do FBG é formalizada e uma análise das probabilidades de falso negativo e falso positivo é desenvolvida. No Capítulo 5, o procedimento aprimorado de reconstrução de rota é abordado e resultados de simulação mostram a eficácia dessa nova abordagem.

Capítulo 4

O Filtro de Bloom Generalizado

UMA generalização do Filtro de Bloom é proposta neste capítulo como forma de evitar que o atacante burle o sistema de rastreamento proposto. Como o filtro convencional é carregado pelos pacotes IP, o atacante pode simplesmente preencher com 1 os bits dos pacotes de ataque correspondentes ao filtro. Dessa forma, a vítima recebe um filtro saturado e não consegue distinguir os roteadores realmente atravessados pelo pacote de outros roteadores, falsos positivos, durante a reconstrução de rota. Com o uso de um Filtro de Bloom Generalizado, é mostrado que o ajuste da condição inicial do filtro pelo atacante é ineficaz e que a probabilidade de falso positivo é limitada. Este limite é ainda controlado por determinados parâmetros do filtro que o atacante não consegue modificar. O custo destas vantagens é a inserção de falsos negativos nos testes de pertinência. Entretanto, também é mostrado que a probabilidade de falso negativo é consideravelmente reduzida usando-se um filtro com um tamanho maior.

Primeiramente, o Filtro de Bloom é explicado seguindo a análise de Fan *et al.* [59], Margoliash [60] e Mitzenmacher [61]. Em seguida, são detalhados alguns trabalhos que modificam este filtro de forma a atender alguma aplicação ou necessidade específica. Posteriormente, a proposta do Filtro de Bloom Generalizado é apresentada, juntamente com uma análise das probabilidades de falso positivo e falso negativo. Por fim, resultados analíticos e de simulação são apresentados para confirmar a eficiência e a robustez desta nova estrutura de dados.

4.1 O Filtro de Bloom

O Filtro de Bloom [29] 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 4.1 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 elementos de S inseridos no filtro.

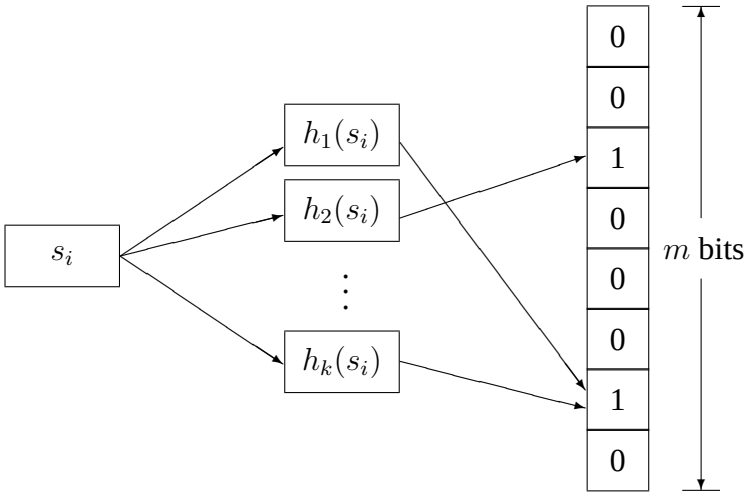


Figura 4.1: Inserção de um elemento em um Filtro de Bloom convencional.

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

$$p = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{-kn/m}, \tag{4.1}$$

sendo a aproximação acima válida somente para valores grandes de m. 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 [61]. Dessa 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 todas as k posições indicadas, ou seja

$$f = \left(1 - e^{-kn/m}\right)^{k}. \tag{4.2}$$

É importante mencionar que na Equação 4.2 é assumido que o número médio de colisões por elemento é próximo de zero. Uma colisão ocorre quando duas ou mais funções *hash* mapeiam um elemento para o mesmo valor de saída. A Figura 4.1 ilustra uma colisão entre duas funções *hash* (h_1 e h_k) para um mesmo elemento. De acordo com esta suposição, para cada elemento, todas as funções *hash* indicam posições diferentes no vetor de bits. É fácil verificar que esta aproximação é válida somente quando $m \gg k$. Caso esta condição não seja satisfeita, também é preciso levar em consideração as probabilidades de falso positivo para os casos de $1, 2, \ldots, k-1$ colisões.

A partir da Equação 4.2, duas observações podem ser feitas à probabilidade de falso positivo f com relação aos seus parâmetros k e m/n . Primeiramente, um aumento na relação m/n sempre reduz a probabilidade de falso positivo. Através da alocação de mais bits por elemento, a fração $(1 - e^{-kn/m})$ é reduzida e, portanto, a probabilidade de falso positivo diminui. Isto pode ser observado na Figura 4.2. É importante ressaltar que este comportamento é válido para qualquer valor escolhido para k. Além disso, existe um compromisso importante entre o número de funções *hash* e a probabilidade de falso positivo. À medida que mais funções são usadas, maior é a fração de bits preenchidos no filtro, colaborando para uma maior probabilidade de falso positivo. Por outro lado, com maiores valores de k, menor é a chance de se encontrar todos os bits indicados pelas funções *hash* em 1. Matematicamente, este compromisso é observado na Equação 4.2 ao perceber que a fração de bits preenchidos $(1 - e^{-kn/m})$ aumenta com valores maiores

de k e que a expressão a^k diminui à medida que k aumenta, para $0 < a < 1$. Este compromisso é ilustrado na Figura 4.3, onde um valor ótimo que minimiza f pode ser claramente observado.

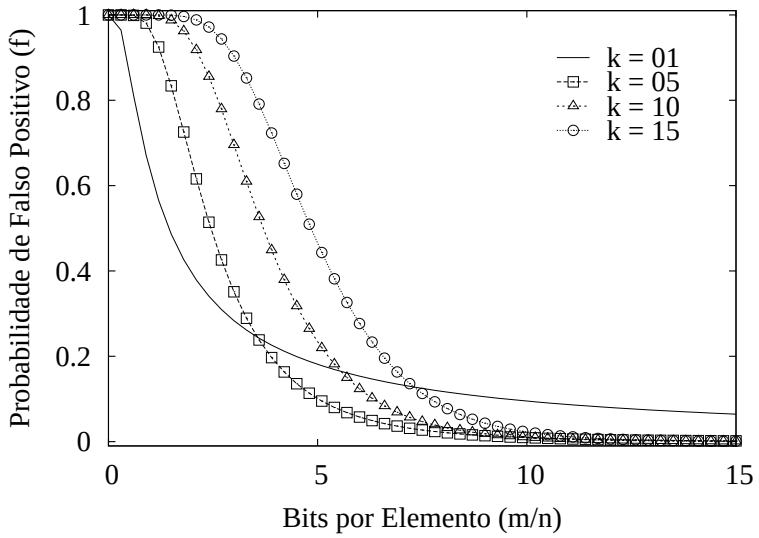


Figura 4.2: Probabilidade de falso positivo de um Filtro de Bloom em função de m/n.

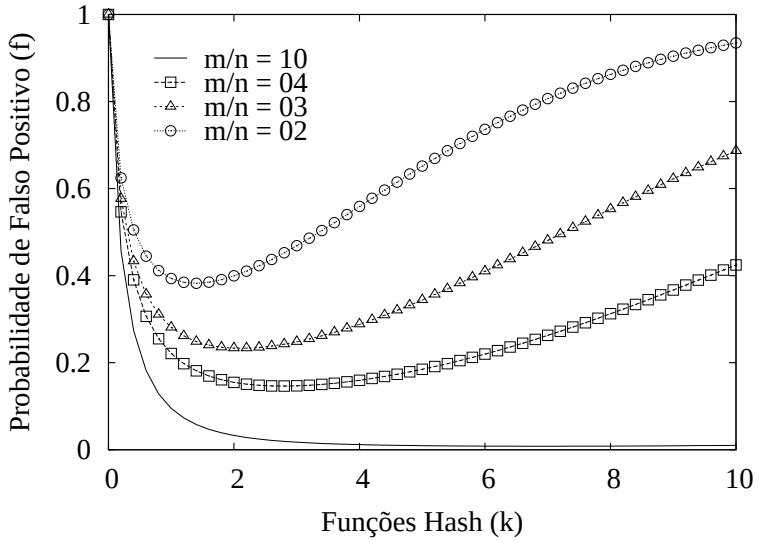


Figura 4.3: Probabilidade de falso positivo de um Filtro de Bloom em função de k.

O número ótimo de funções *hash* é encontrado através da derivação simples de f em relação a k, que fornece

$$\frac{df}{dk} = \left[\ln\left(1 - e^{-kn/m}\right) + \frac{kn}{m}\frac{e^{-kn/m}}{1 - e^{-kn/m}}\right]\left(1 - e^{-kn/m}\right)^k. \tag{4.3}$$

Igualando a derivada a zero e percebendo que $(1 - e^{-kn/m})^k$ nunca se anula, resulta em

$$\begin{aligned}(1-p)\ln(1-p) &= p\ln p \\ (1-p)^{(1-p)} &= p^p. \end{aligned} \tag{4.4}$$

Supondo que k é finito e positivo, a única solução para a Equação 4.4 é $p = 0,5$. Este resultado significa que a probabilidade de falso positivo é minimizada quando a fração de bits em 0 é igual a fração de bits em 1. Neste caso, $k = (m/n)\ln 2$ e a probabilidade de falso positivo f é expressa por $(0,5)^k = (0,6185)^{m/n}$. Na prática, o valor de k, assim como o de n e m, é inteiro.

De forma a diferenciá-lo da versão generalizada proposta na Seção 4.3, o Filtro de Bloom mostrado nesta seção também é chamado de Filtro de Bloom convencional, ou simplesmente filtro convencional no restante deste trabalho.

4.2 Trabalhos Relacionados

O Filtro de Bloom foi inicialmente empregado em uma aplicação de separação de sílabas automatizada [29]. A abordagem é usada para reduzir o número de vezes que uma palavra é procurada em um dicionário residente no disco. Como a maioria das palavras pode ser separada através da aplicação de regras simples, um Filtro de Bloom é usado para representar somente as palavras que precisam de uma verificação no dicionário. Desta forma, cada palavra é testada contra o filtro e somente aquelas reconhecidas pelo filtro são verificadas no dicionário; o resto das palavras são separadas com regras simples. Neste caso, os falsos positivos levam a uma verificação desnecessária no dicionário para uma palavra que pode ser separada facilmente.

Seguindo a idéa de filtragem antes de uma busca, Severance e Lohman [62] propõem o uso de Filtros de Bloom para melhorar o acesso a arquivos diferenciais. Atualizações de banco de dados podem ser realizadas de maneira eficiente se um arquivo de dados separado for usado para armazenar as alterações ocorridas em um determinado período. Mais tarde, as atualizações armazenadas neste arquivo diferencial são aplicadas ao banco de dados. Entretanto, o custo desta abordagem é que o acesso ao banco de dados não é

mais imediato. Para acessar um determinado dado, é preciso sempre verificar se existe alguma atualização daquele dado específico no arquivo diferencial. Caso exista, é preciso enviar o dado atualizado para o usuário; caso não exista, o próprio dado armazenado no banco de dados é retornado. Desta forma, melhorias no processo de atualização são alcançadas ao custo de um maior atraso no acesso aos dados. Para melhorar o tempo de acesso, os autores sugerem o uso de um Filtro de Bloom para manter o registro dos dados que foram modificados e que se encontram no arquivo diferencial. Assim, antes de um acesso, uma verificação é realizada no filtro para determinar se aquele dado foi alterado ou não. Desta forma, somente no caso dos dados terem sido alterados ou no caso de falsos positivos é que estes dados são procurados no arquivo diferencial.

Outras aplicações também conseguiram economias significativas de espaço com os Filtros de Bloom [60, 63]. Recentemente, estes filtros têm sido amplamente empregados em redes de computadores [56]. No entanto, o Filtro de Bloom original tem sido constantemente adaptado para atender aos requisitos de diversas aplicações. Por exemplo, Fan *et al.* [59] modificaram o filtro convencional para permitir não somente a inserção de elementos, mas também a retirada de elementos do filtro. A proposta é usar um contador ao invés de um simples bit em cada posição do vetor do filtro. Desta forma, os contadores são incrementados durante uma inserção de um elemento e decrementados durante uma retirada. Esta estrutura, denominada Filtro de Bloom Contador (*Counting Bloom Filter*), é perfeitamente adequado para a representação de conjuntos dinâmicos, como o conteúdo de *caches*.

Uma outra variação, denominada Filtro de Bloom Comprimido (*Compressed Bloom Filter*), foi introduzido por Mitzenmacher [61] para permitir a transmissão eficiente de um filtro entre duas estações. Conforme visto na Seção 4.1, o Filtro de Bloom otimizado possui uma taxa de falso positivo mínima quando metade dos seus bits estão preenchidos e a outra metade está zerada. Portanto, a compressão do filtro neste caso é completamente ineficiente. Entretanto, Mitzenmacher sugere usar filtros subótimos maiores nas estações finais e comprimí-los antes da transmissão. O autor concluiu que a probabilidade de falso positivo dos filtros comprimidos é sempre menor ou igual a dos filtros convencionais, para um tamanho de transmissão fixo. O custo é uma maior quantidade de memória e o processamento adicional da compressão e descompressão nas estações finais.

Dharmapurikar *et al.* [64] descreve uma técnica de inspeção de pacotes baseada nos chamados Filtros de Bloom Paralelos. O objetivo desta inspeção é detectar de maneira eficiente a presença de determinadas cadeias de bytes, denominadas assinaturas, dentro dos pacotes da rede. Tal detecção poderia ser usada, por exemplo, em aplicações de segurança, bilhetagem e roteamento por conteúdo. Os autores propõem agrupar todas as assinaturas de acordo com os seus respectivos tamanhos e armazenar as assinaturas de cada grupo em um Filtro de Bloom convencional. Diversos filtros são usados, sendo que cada um armazena as assinaturas de um determinado tamanho. Desta forma, assinaturas de tamanhos diferentes podem ser verificadas em paralelo através dos respectivos filtros. Sob uma abordagem semelhante, os autores propõem o uso de Filtros de Bloom Paralelos para agilizar o processo de roteamento [65].

Uma proposta interessante é adaptar o Filtro de Bloom para representar multiconjuntos ao invés de conjuntos simples. Um multiconjunto é muito similar a um conjunto, com a exceção que um determinado elemento pode estar presente mais de uma vez no multiconjunto. Estan and Varghese [66] introduzem um Filtro de Bloom com suporte a multiconjuntos para identificar fluxos intensos de dados em um roteador. Este filtro também usa um vetor de contadores ao invés de um vetor de bits simples. Ao receber um pacote, o roteador incrementa os contadores do respectivo fluxo com o tamanho do pacote recebido. Quando todos os contadores do respectivo fluxo estão acima de um determinado limiar, o fluxo de dados é considerado intenso. Cohen and Matias [67] introduzem uma abordagem semelhante que possibilita a execução de testes de multiplicidade. Dado um elemento específico, o chamado Filtro de Bloom Espectral (*Spectral Bloom Filter*) retorna uma estimativa do número de ocorrências daquele elemento no filtro. Uma representação alternativa de um multiconjunto, o Filtro de Bloom Espaço-Código (*Space-Code Bloom Filter*), é introduzido por Kumar *et al.* [68]. A invés do filtro usar somente um grupo fixo de funções *hash*, os autores propõem o uso de grupos diferentes de funções *hash*. A cada inserção, um grupo é selecionado aleatoriamente e os bits indicados pelas funções do grupo escolhido são preenchidos no filtro. Neste caso, a multiplicidade é estimada pelo número de grupos cujas funções indicam somente bits preenchidos.

Shanmugasundaram *et al.* [69] introduzem uma estrutura de dados chamada Filtro de Bloom Hierárquico (*Hierarchical Bloom Filter*) para possibilitar a busca de sub-cadeias

de caracteres. Desta forma, é possível saber se uma parte de uma cadeia de caracteres foi inserida no filtro com uma baixa probabilidade de falso positivo. Inicialmente, a cadeia a ser inserida é dividida em blocos de tamanho pré-definido b. Em seguida, é concatenado ao final de cada bloco o seu respectivo deslocamento (*offset*) em relação ao início da cadeia. Cada bloco com o seu respectivo deslocamento é então inserido em um Filtro de Bloom. Para verificar se uma subcadeia de caracteres foi inserida, ela é primeiramente quebrada em blocos de tamanho b e, ao final de cada bloco, são adicionados números de deslocamentos consecutivos. Desta forma, o primeiro bloco recebe o deslocamento 0, o segundo recebe o deslocamento 1, e assim por diante. Todos os blocos da subcadeia são testados contra o filtro e, caso um deles não seja reconhecido, o número de deslocamento inicial é incrementado. Ou seja, os blocos da sub-cadeia são então numerados a partir de 1. O mesmo procedimento é realizado até que o número máximo de deslocamento inicial seja alcançado ou que todos os blocos sejam reconhecidos. Nesse último caso, assume-se que a subcadeia foi realmente inserida no filtro. O problema desta abordagem é que falsos positivos podem ocorrer através da combinação de blocos provenientes de cadeias de caracteres diferentes. Para melhorar a acurácia da resposta, a mesma cadeia também é inserida no filtro usando-se blocos com tamanho $2b$. Desta forma, dois blocos consecutivos de tamanho b reconhecidos pelo filtro podem ser confirmados por uma verificação de um bloco de tamanho $2b$. Se mais acurácia é desejada, blocos de tamanho $4b$ também podem ser inseridos e assim por diante. No entanto, resultados mais precisos necessitam de filtros de tamanho maior.

Para melhorar a localização e o roteamento de documentos em redes *peer-to-peer* (P2P), Rhea and Kubiatowicz [70] introduziram o Filtro de Bloom Atenuado (*Attenuated Bloom Filter*). Um filtro atenuado de profundidade d é na verdade um vetor formado por d Filtros de Bloom convencionais, onde o i-ésimo filtro armazena os documentos que podem ser alcançados em i saltos. Os autores propõem que cada nó mantenha um filtro atenuado para cada nó vizinho na rede P2P. Desta forma, cada filtro atenuado armazena os documentos que podem ser alcançados por aquele vizinho em até d saltos. O algoritmo de verificação é descrito a seguir. Inicialmente, o nó que busca o documento examina o primeiro nível do filtro atenuado de cada vizinho. Se nenhuma resposta positiva é retornada, os filtros de níveis mais altos são verificados em ordem crescente. Uma vez

que uma resposta positiva é encontrada, o pedido do documento é encaminhado para o respectivo vizinho. Este vizinho realiza o mesmo procedimento de verificação nos seus filtros para encaminhar o pedido adequadamente. No caso de um falso positivo, o pedido chega a um nó que não contém o documento em questão. Um algoritmo determinístico é então usado para encontrar um nó que realmente atenda o pedido.

O filtro proposto neste trabalho visa deixar o Filtro de Bloom menos dependente da sua condição inicial. Como o filtro é carregado no cabeçalho do pacote e o seu estado inicial é configurado pelo próprio atacante, um filtro robusto a qualquer condição inicial se torna necessário.

4.3 O Filtro de Bloom Generalizado

Esta seção apresenta a proposta do Filtro de Bloom Generalizado (FBG). O objetivo da generalização proposta é tornar o Filtro de Bloom menos dependente da sua condição inicial, de forma que um atacante não consiga interferir no rastreamento de pacotes através da alteração do conteúdo inicial do filtro.

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 sejam inicialmente zerados. No Filtro de Bloom Generalizado, 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 4.4 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.

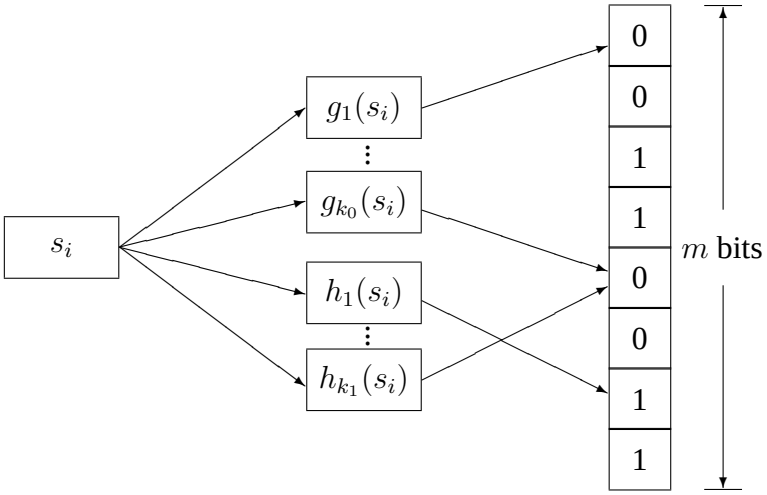


Figura 4.4: Inserção de um elemento em um Filtro de Bloom Generalizado.

4.3.1 Falsos Positivos

A probabilidade de falso positivo de um Filtro de Bloom Generalizado 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

$$q_0 = \left[1 - \left(1 - \frac{1}{m}\right)^{k_0}\right] \approx \left(1 - e^{-k_0/m}\right). \tag{4.5}$$

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

$$q_1 = \left[1 - \left(1 - \frac{1}{m}\right)^{k_1}\right]\left(1 - \frac{1}{m}\right)^{k_0} \approx \left(1 - e^{-k_1/m}\right)e^{-k_0/m}. \tag{4.6}$$

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

$$(1 - q_0 - q_1) = \left(1 - \frac{1}{m}\right)^{k_0+k_1} \approx e^{-(k_0+k_1)/m}. \tag{4.7}$$

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 de 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

$$p_0(n) = p_0(0)\,(1 - q_0 - q_1)^n + \sum_{i=0}^{n-1} q_0\,(1 - q_0 - q_1)^i$$

$$= p_0(0)\,(1-q_0-q_1)^n + q_0\left[\frac{1-(1-q_0-q_1)^n}{1-(1-q_0-q_1)}\right]$$
$$= p_0(0)\,(1-q_0-q_1)^n + \frac{q_0}{q_0+q_1}\Big[1-(1-q_0-q_1)^n\Big]. \tag{4.8}$$

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

$$p_1(n) = p_1(0)\,(1-q_0-q_1)^n + \sum_{i=0}^{n-1} q_1\,(1-q_0-q_1)^i$$
$$= p_1(0)\,(1-q_0-q_1)^n + q_1\left[\frac{1-(1-q_0-q_1)^n}{1-(1-q_0-q_1)}\right]$$
$$= p_1(0)\,(1-q_0-q_1)^n + \frac{q_1}{q_0+q_1}\Big[1-(1-q_0-q_1)^n\Big], \tag{4.9}$$

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 Filtro de Bloom Generalizado é calculada da seguinte forma

$$f_p = p_0(n)^{b_0} p_1(n)^{b_1} = p_0(n)^{b_0}\,[1-p_0(n)]^{b_1} = [1-p_1(n)]^{b_0}\,p_1(n)^{b_1}. \tag{4.10}$$

4.3.2 Falsos Negativos

Falsos positivos ocorrem para elementos externos com a mesma probabilidade para cada elemento checado. Ou seja, cada elemento que não foi inserido no Filtro de Bloom Generalizado tem a mesma probabilidade de ser um falso positivo. Por outro lado, falsos negativos ocorrem somente para elementos realmente *inseridos* com uma probabilidade diferente para cada elemento. Um fator que afeta diretamente a probabilidade de falso negativo é a ordem de inserção. No caso, 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.

Para se calcular a probabilidade de falso negativo, é necessário determinar 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

$$\begin{aligned} p_{00}(n-i) &= (1-q_0-q_1)^i + \sum_{j=0}^{i-1} q_0 \, (1-q_0-q_1)^j \\ &= (1-q_0-q_1)^i + q_0 \left[\frac{1-(1-q_0-q_1)^i}{1-(1-q_0-q_1)} \right] \\ &= (1-q_0-q_1)^i + \frac{q_0}{q_0+q_1} \left[1-(1-q_0-q_1)^i \right]. \end{aligned} \tag{4.11}$$

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

$$\begin{aligned} p_{11}(n-i) &= (1-q_0-q_1)^i + \sum_{j=0}^{i-1} q_1 \, (1-q_0-q_1)^j \\ &= (1-q_0-q_1)^i + q_1 \left[\frac{1-(1-q_0-q_1)^i}{1-(1-q_0-q_1)} \right] \\ &= (1-q_0-q_1)^i + \frac{q_1}{q_0+q_1} \left[1-(1-q_0-q_1)^i \right]. \end{aligned} \tag{4.12}$$

A partir das Equações 4.11 e 4.12, 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

$$f_n(n-i) = 1 - p_{00}(n-i)^{b_0} p_{11}(n-i)^{b_1}. \tag{4.13}$$

4.3.3 O Filtro de Bloom como um Caso Particular

Como esperado, o Filtro de Bloom convencional é na verdade um caso particular do Filtro de Bloom Generalizado. Para os falsos positivos, a Equação 4.10 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

$$\begin{aligned} b_1 &= m(1 - e^{-k_1/m}) \\ &= m\left[1 - \left(1 - \frac{k_1}{m} + \frac{k_1^2}{2m^2} - \cdots\right)\right] \approx k_1. \end{aligned} \tag{4.14}$$

Logo, a probabilidade de falso positivo f_p se reduz à probabilidade do Filtro de Bloom convencional, dada pela Equação 4.2.

De forma semelhante, a probabilidade de falso negativo na Equação 4.13 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.

4.3.4 Aplicação

No sistema de rastreamento proposto, os elementos inseridos no Filtro de Bloom Generalizado 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 filtro generalizado. 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 Filtro de Bloom Generalizado.

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, são necessárias 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 Filtro de Bloom Generalizado do pacote é atualizado inicialmente com o resultado de uma operação OU bit-a-bit do próprio filtro do pacote com o primeiro registrador da interface. Em seguida, uma operação E bit-a-bit do Filtro de Bloom Generalizado 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.

4.4 Resultados Analíticos e de Simulação

De forma a analisar o comportamento de um Filtro de Bloom Generalizado e comprovar as expressões analíticas desenvolvidas na Seção 4.3, foi desenvolvido um simulador em C++ [71]. O simulador é baseado em uma classe denominada GBF (*Generalized Bloom Filter* - GBF) que contém os métodos para inserção e verificação de elementos em um vetor de bits. Para cada rodada de simulação, novas funções *hash* são selecionadas, o vetor de bits é inicializado de acordo com um dada condição inicial e, em seguida, os elementos de um determinado conjunto são inseridos no filtro. Após as inserções, testes

de pertinência usando elementos externos e elementos inseridos são realizados para medir respectivamente as taxas de falsos positivos e falsos negativos.

Para obter as funções *hash* independentes e uniformes assumidas nas Seções 4.1 e 4.3, é usada uma classe universal de funções *hash* [72]. Esta classe é definida da seguinte maneira. Sejam os elementos definidos como números inteiros do universo $S = \{1, 2, \ldots, p-1\}$ e seja $\{0, 1, \ldots, m-1\}$ o espaço de saída de cada função. A classe H de funções *hash* $h_{c,d}$ que mapeiam um elemento $x \in S$ no respectivo espaço de saída é definida como

$$H = \{h_{c,d}(\cdot) \mid 0 < c < p,\ 0 \leq d < p\}, \tag{4.15}$$

onde

$$h_{c,d}(x) = [(cx + d) \bmod p] \bmod m. \tag{4.16}$$

Os números c e d são inteiros dentro do intervalo definido pela Equação 4.15. Para cada função *hash* usada na simulação, c e d são escolhidos arbitrariamente. O número p é definido como um número primo.

Além disso, para mostrar as vantagens do FBG em relação ao filtro convencional, também é apresentada nesta seção uma comparação analítica entre os dois filtros: o Filtro de Bloom convencional e o novo Filtro de Bloom Generalizado. A análise aborda três métricas distintas: falsos positivos, falsos negativos e a interferência da condição inicial.

Deve ser ressaltado que os resultados obtidos por simulação foram muito próximos dos valores analíticos, o que valida as expressões analíticas desenvolvidas. Para uma confiabilidade de 95%, o maior intervalo de confiança obtido foi 0,003. Este intervalo é muito pequeno e não é mostrado nos gráficos. Todos os gráficos apresentados nesta seção se referem a resultados de simulação, mostrados como pontos discretos. Entretanto, os respectivos resultados analíticos também são apresentados como curvas contínuas nos gráficos mostrados.

4.4.1 Falsos Positivos: Um Limitante Superior

Um falso positivo implica um teste de pertinência reconhecer um elemento externo (não pertencente ao conjunto) como um autêntico elemento do conjunto. À medida que a probabilidade de falso positivo aumenta, mais elementos externos são identificados como elementos do conjunto. Logo, sempre se busca uma baixa probabilidade de falso positivo.

Primeiramente, uma simplificação das Equações 4.8, 4.9 e 4.10 é realizada. Supondo que $m \gg k_0$ e $m \gg k_1$, as Equações 4.8 e 4.9 podem ser reescritas da seguinte maneira

$$p_0(n) = p_0(0)\left(1 - q_0 - q_1\right)^n + \frac{k_0}{k_0 + k_1}\Big[1 - \left(1 - q_0 - q_1\right)^n\Big], \tag{4.17}$$

$$p_1(n) = p_1(0)\left(1 - q_0 - q_1\right)^n + \frac{k_1}{k_0 + k_1}\Big[1 - \left(1 - q_0 - q_1\right)^n\Big]. \tag{4.18}$$

Além disso, neste caso, as aproximações $b_0 \approx k_0$ e $b_1 \approx k_1$ também são válidas. Dessa forma, a Equação 4.10 também pode ser simplificada conforme a expressão

$$f_p \approx p_0(n)^{k_0} p_1(n)^{k_1} = \left[1 - p_1(n)\right]^{k_0} p_1(n)^{k_1} = p_0(n)^{k_0}\left[1 - p_0(n)\right]^{k_1}. \tag{4.19}$$

As Figuras 4.5 e 4.6 mostram como varia a probabilidade de falso positivo f_p em função de $p_1(n)$, de acordo com a Equação 4.19 e resultados de simulação. A probabilidade $p_1(n)$ também pode ser encarada como a fração de bits do vetor em 1 depois de inseridos os n elementos. Na Figura 4.5, a curva onde $k_0 = 0$ representa o Filtro de Bloom convencional. Para este caso, pode-se constatar que a probabilidade de falso positivo aumenta à medida que $p_1(n)$ aumenta. Este resultado está de acordo com a Equação 4.2. As curvas com outros valores de k_0 representam o Filtro de Bloom Generalizado. Pode-se observar um compromisso claro entre a distribuição final de bits e a probabilidade de falso positivo nestas curvas. Este compromisso pode ser entendido ao notar que, na média, k_0 bits em 0 e k_1 bits em 1 são necessários para causar um falso positivo. Entretanto, se $p_1(n)$ é baixo, é fácil encontrar um bit em 0 e difícil encontrar um bit em 1. Por outro lado, se $p_1(n)$ é alto, é fácil encontrar um bit em 1 e difícil encontrar um bit em 0. Na Figura 4.6, um comportamento similar é observado. A curva para $k_1 = 0$ representa o dual do Filtro de Bloom convencional, com somente funções que zeram bits. Para este filtro, quanto maior a fração de bits em 0, maior é a probabilidade de falso positivo. As outras curvas representam os duais dos filtros generalizados apresentados na Figura 4.5.

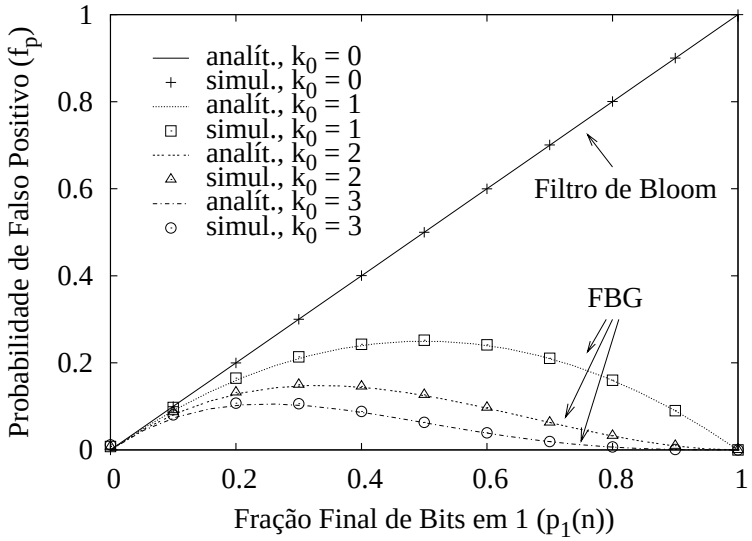


Figura 4.5: Probabilidade de falso positivo de um FBG em função de $p_1(n)$, para $k_1 = 1$.

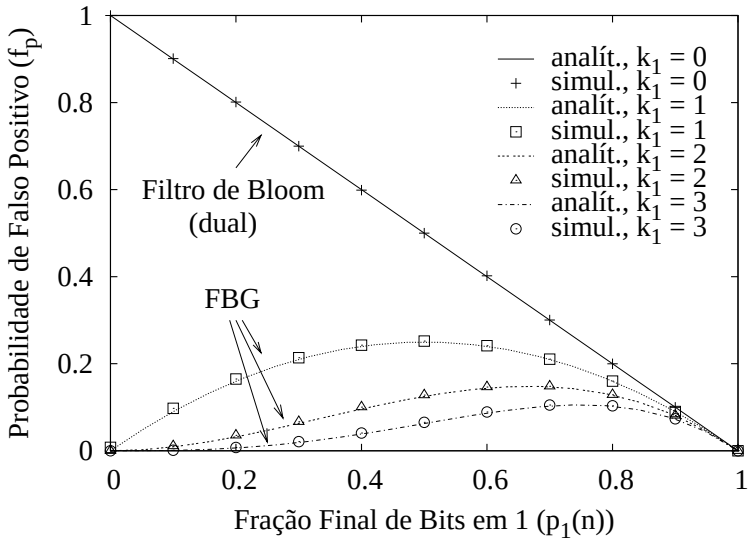


Figura 4.6: Probabilidade de falso positivo de um FBG em função de $p_1(n)$, para $k_0 = 1$.

O ponto máximo das Figuras 4.5 e 4.6 pode ser calculado tomando-se a derivada de f_p em relação a $p_1(n)$ e igualando-a a zero. Derivando a Equação 4.19 em relação a $p_1(n)$ e igualando-a a zero, chega-se a

$$k_0 \left[1 - p_1(n)\right]^{k_0 - 1} p_1(n)^{k_1} = k_1 \left[1 - p_1(n)\right]^{k_0} p_1(n)^{k_1 - 1}. \tag{4.20}$$

Supondo que $p_1(n) \neq 0$ e $p_1(n) \neq 1$, pode-se determinar a partir da Equação 4.20 que a probabilidade de falso positivo máxima de um FBG ocorre para

$$p_1(n) = \frac{k_1}{k_0 + k_1}. \tag{4.21}$$

A probabilidade de falso positivo máxima F_p é determinada ao substituir o resultado da Equação 4.21 na Equação 4.19 e seu valor é

$$F_p = \left(\frac{k_0}{k_0 + k_1}\right)^{k_0} \left(\frac{k_1}{k_0 + k_1}\right)^{k_1}. \tag{4.22}$$

Diferentemente do Filtro de Bloom convencional, o Filtro de Bloom Generalizado tem uma probabilidade de falso positivo limitada por F_p. Esta probabilidade máxima é exclusivamente determinada pelos parâmetros k_0 e k_1 escolhidos para o sistema. Em virtude desta limitação, a interferência da ação do atacante no processo de rastreamento também é restringida, como tratado na Seção 4.4.3.

4.4.2 Falsos Negativos: Um Limitante Superior

Com a adoção do FBG, falsos negativos podem ocorrer durante o processo de reconstrução da rota de ataque. Um falso negativo significa não detectar um roteador pelo qual o pacote rastreado realmente passou. Somente um falso negativo já é suficiente para interromper precocemente a reconstrução da verdadeira rota de ataque. Logo, a probabilidade de falsos negativos deve necessariamente ser mantida baixa. Vale ressaltar que o atacante não interfere na probabilidade de falso negativo, de acordo com as Equações 4.11, 4.12 e 4.13. Pode-se verificar que os termos $p_0(0)$ e $p_1(0)$ não aparecem nessas equações.

A Figura 4.7 mostra a probabilidade de falso negativo para cada elemento inserido, de acordo com a Equação 4.13 e resultados de simulação. De acordo com as definições realizadas, o elemento 1 representa o primeiro elemento inserido e o elemento 10 representa o último elemento inserido. No sistema de rastreamento, o primeiro elemento corresponde ao roteador mais perto do atacante e o n-ésimo elemento corresponde ao roteador mais próximo da vítima. Na Figura 4.7, a curva onde $k_0 = 0$ representa o Filtro de Bloom convencional. Conforme esperado, os falsos negativos para este caso são sempre zero independentemente do elemento. Como o filtro convencional não usa funções *hash* que zeram bits, a presença de falsos negativos é impossível. As outras curvas desta figura representam o FBG. Neste caso, verifica-se que elementos inseridos primeiro têm uma maior probabilidade de falso negativo. Isto acontece porque os primeiros elementos têm

mais elementos inseridos depois deles e, como conseqüência, uma inversão dos seus bits marcados é mais provável. Uma outra importante observação é que a probabilidade de falso negativo cresce conforme k_0 aumenta. Isto ocorre porque, quanto mais funções são usadas, maior é a probabilidade de um elemento ter um bit invertido por um elemento subseqüente. Ao fixar k_0 e variar k_1, um resultado semelhante é encontrado. A Figura 4.8 ilustra este caso e o mesmo comportamento da Figura 4.7 é apresentado.

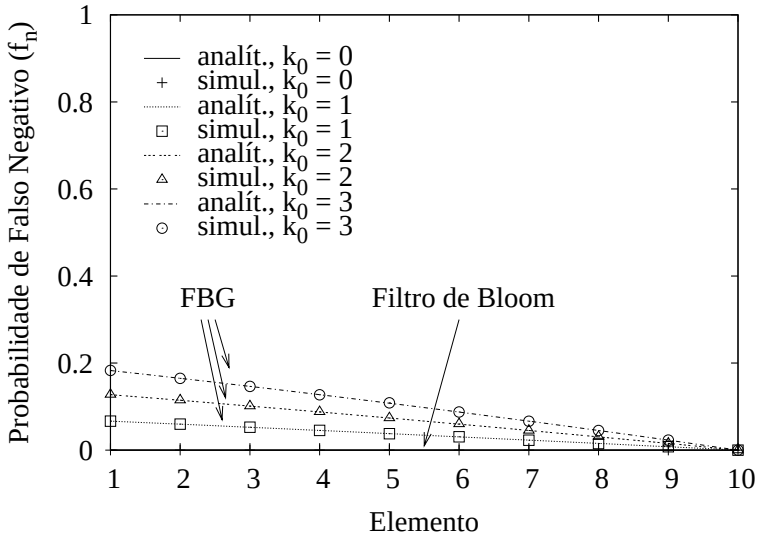


Figura 4.7: Probabilidade de falso negativo de um FBG em função de cada elemento inserido, para $k_1 = 1, m = 256$ e $n = 10$.

De acordo com as Figuras 4.7 e 4.8, a probabilidade de falso negativo é uma função não-crescente dos elementos inseridos. Na verdade, este comportamento é sempre válido e pode ser provado analiticamente se observado que $p_{00}(x) \leq p_{00}(y)$ e $p_{11}(x) \leq p_{11}(y)$, se $x < y$. A partir da Equação 4.13, é fácil verificar que $f_n(x) \geq f_n(y)$ é sempre verdade quando $x < y$. Portanto, é possível calcular um limitante superior para a probabilidade de falso negativo do FBG. Seja $f_n(0)$ a probabilidade de falso negativo de um elemento fictício inserido antes dos n elementos de um determinado conjunto. De acordo com o resultado anterior, tem-se $f_n(0) \geq f_n(1) \geq f_n(2) \geq \ldots \geq f_n(n)$. Dessa forma, uma probabilidade de falso negativo máxima de um FBG pode ser definida como sendo $f_n(0)$. Supondo que $m \gg k_0$ e $m \gg k_1$, as Equações 4.11 e 4.12 para o hipotético elemento 0 podem ser rescritas como

$$p_{00}(0) = (1 - q_0 - q_1)^n + \frac{k_0}{k_0 + k_1} \left[1 - (1 - q_0 - q_1)^n\right], \tag{4.23}$$

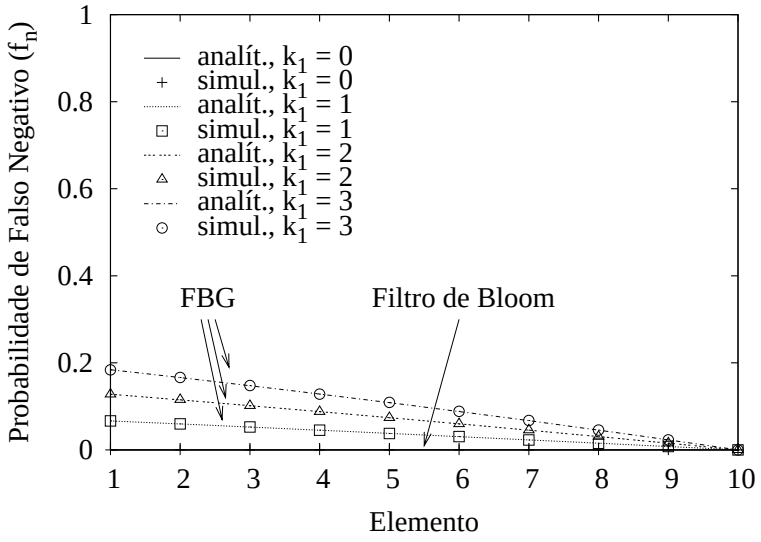


Figura 4.8: Probabilidade de falso negativo de um FBG em função de cada elemento inserido, para $k_0 = 1, m = 256$ e $n = 10$.

$$p_{11}(0) = (1 - q_0 - q_1)^n + \frac{k_1}{k_0 + k_1}\left[1 - (1 - q_0 - q_1)^n\right]. \tag{4.24}$$

Observando que neste caso $b_0 \approx k_0$ e $b_1 \approx k_1$, a Equação 4.13 pode ser simplificada por

$$f_n(n - i) = 1 - p_{00}(n - i)^{k_0} p_{11}(n - i)^{k_1}. \tag{4.25}$$

Substituindo as Equações 4.23 e 4.24 na Equação 4.25, a probabilidade de falso negativo máxima F_n é definida como

$$F_n = f_n(0) = 1 - p_{00}(0)^{k_0} p_{11}(0)^{k_1}. \tag{4.26}$$

De acordo com as simplificações realizadas, F_n é unicamente definida por k_0, k_1 e pela relação m/n. Portanto, o projetista do sistema pode primeiro arbitrar a probabilidade de falso positivo máxima desejada através da definição dos parâmetros k_0 e k_1. A probabilidade de falso negativo máxima pode ser então alcançada através de um ajuste na relação m/n. Entretanto, menores probabilidades de falso negativo são alcançadas ao custo de mais bits por elemento, ou seja, valores maiores para a relação m/n.

A Figura. 4.9 ilustra a probabilidade de falso negativo máxima F_n em função da relação m/n, de acordo com a Equação 4.26 e resultados de simulação. Para o caso do Filtro de Bloom convencional, a probabilidade de falso negativo máxima permanece zerada,

uma vez que não ocorrem falsos negativos neste caso. Para o FBG, pode ser visto que um aumento em m/n diminui a probabilidade de falso negativo máxima. Isso ocorre porque, aumentando o número de bits por elemento, aumenta-se o espaço de saída das funções *hash* e probabilidade de um bit ser invertido diminui. É também importante ressaltar que os resultados apresentados são as probabilidades de falso negativo *máximas*. Na verdade, a probabilidade de falso negativo individual de cada elemento inserido em um FBG varia entre 0 e aproximadamente F_n, dependendo da ordem de inserção. Fixando-se k_0 e variando k_1, o mesmo resultado é alcançado, como mostra a Figura 4.10. Isso ocorre porque k_0 e k_1 podem ser trocados na Equação 4.26 sem alterar o resultado final.

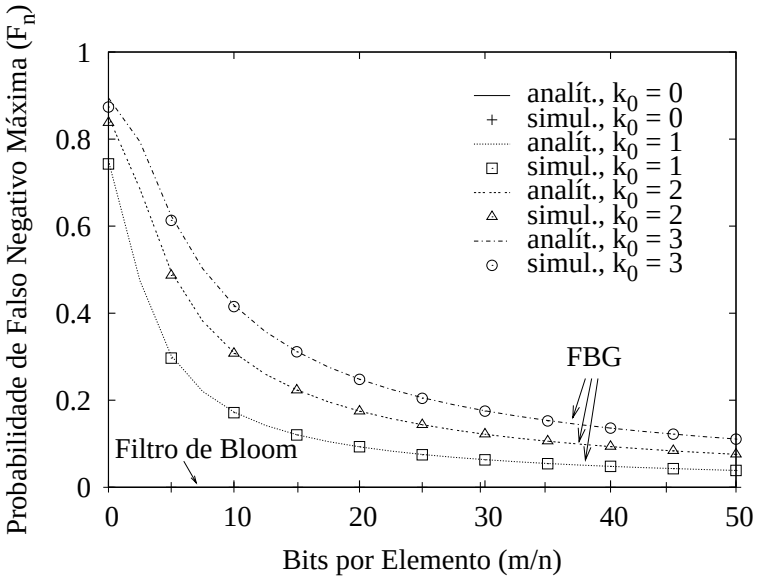


Figura 4.9: Probabilidade de falso negativo de um FBG em função de m/n, para $k_1 = 1$.

4.4.3 Interferência da Condição Inicial: Ação Limitada

A probabilidade de falso positivo do filtro convencional e do filtro generalizado varia com a condição inicial do vetor de bits. Primeiramente, o filtro convencional é abordado. A Figura 4.11 mostra a dependência da probabilidade de falso positivo de um filtro convencional em relação à sua condição inicial. Como visto, para qualquer número k de funções *hash* escolhido, a probabilidade de falso positivo f sempre tende a 1 à medida que mais bits são inicialmente preenchidos. Devido ao compromisso entre f e k já citado anteriormente, pode-se perceber que a probabilidade de falso positivo é menor para alguns

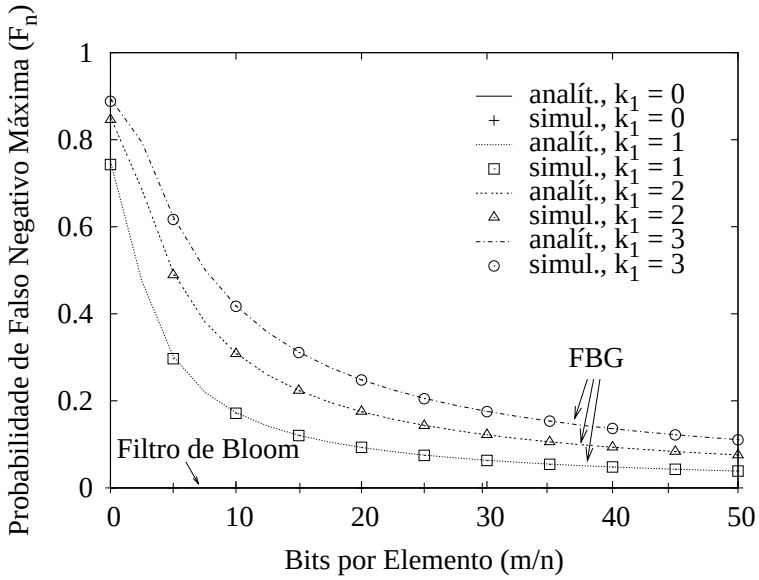


Figura 4.10: Probabilidade de falso negativo de um FBG em função de m/n, para $k_0 = 1$.

valores de k e maior para outros, mas em todos os casos o comportamento é o mesmo. A Figura 4.12 mostra um comportamento semelhante, porém variando-se a relação m/n. De forma geral, a probabilidade de falso positivo pode atingir 100% quando os todos os bits do filtro são inicialmente preenchidos com 1.

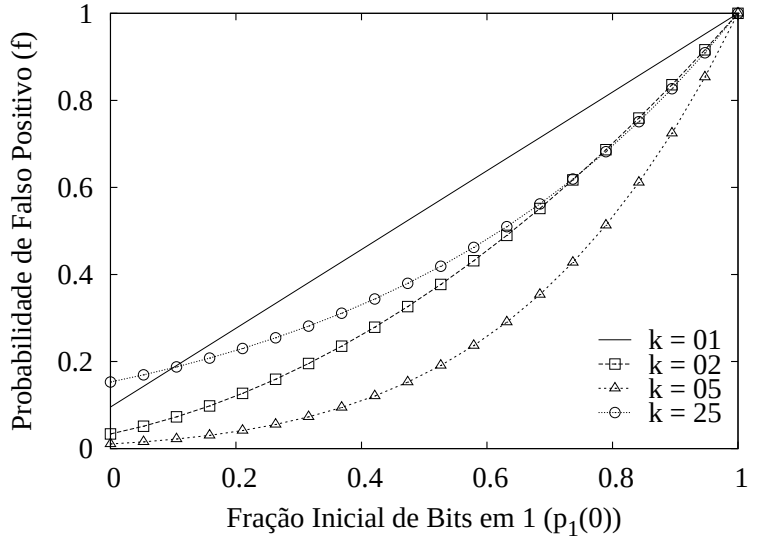


Figura 4.11: Probabilidade de falso positivo de um Filtro de Bloom em função de $p_1(0)$, para $m/n = 10$.

Sendo utilizado um FBG ao invés de um filtro convencional, a interferência da condição inicial diminui consideravelmente. A Figura 4.13 mostra a probabilidade de falso

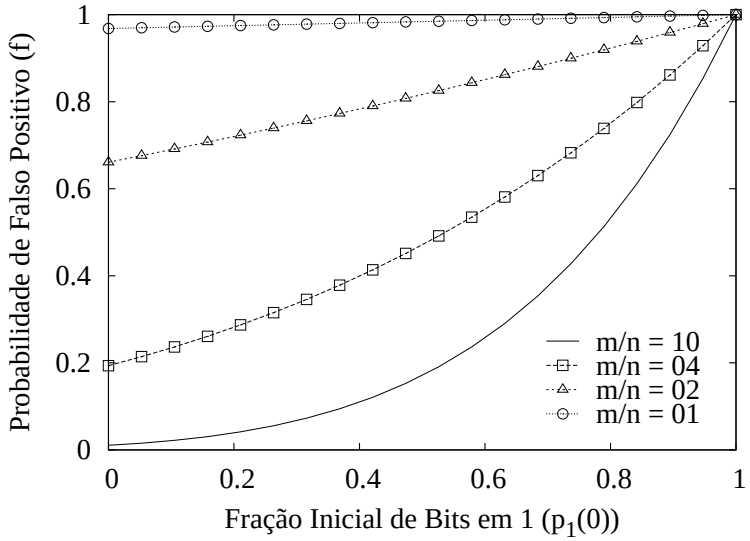


Figura 4.12: Probabilidade de falso positivo de um Filtro de Bloom em função de $p_1(0)$, para $k = 5$.

positivo do FBG em função da fração inicial de bits em 1, $p_1(0)$, de acordo com a Equação 4.19 e resultados de simulação. De acordo com a Figura 4.13, é possível perceber que a condição inicial do filtro pode ser ajustada de forma a maximizar a probabilidade de falso positivo. Se k_0 for mantido fixo e k_1 variar, um resultado semelhante é alcançado, como mostrado na Figura 4.14. Nestes dois casos, o valor máximo da probabilidade de falso positivo F_p pode ser atingido se o valor encontrado na Equação 4.27 abaixo for substituído nas Equações 4.17, 4.18 e 4.19.

$$p_1(0) = \frac{k_1}{k_0 + k_1}. \tag{4.27}$$

Desta forma, a probabilidade de falso positivo é muito menos dependente da condição inicial do filtro quando um FBG é usado ao invés de um Filtro de Bloom convencional. Por exemplo, adotando-se um FBG, a probabilidade máxima de falso positivo é reduzida em pelo menos 75%, para o caso quando $k_0 = k_1 = 1$. Se for usado $k_0 = k_1 = 2$, a probabilidade máxima de falso positivo é reduzida em 93.75%.

De acordo com a Equação 4.18, sendo o FBG iniciado conforme a Equação 4.27, $p_1(n)$ permanece constante independentemente no número n de elementos inseridos. Além disso, a fração $p_1(n)$ também permanece constante quando muitos elementos são inseridos no filtro, ou seja, n é grande. Nos dois casos, este valor constante é o mesmo que

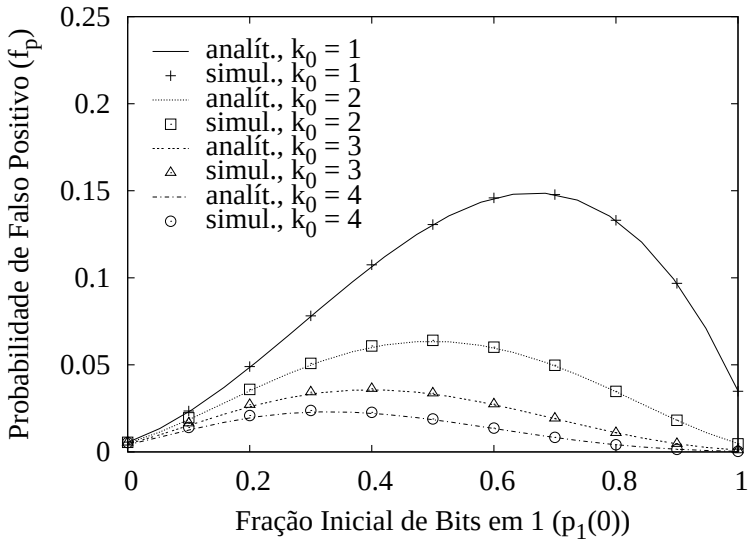


Figura 4.13: Probabilidade de falso positivo de um FBG em função de $p_1(0)$, para $k_1 = 2$, $m/n = 25$.

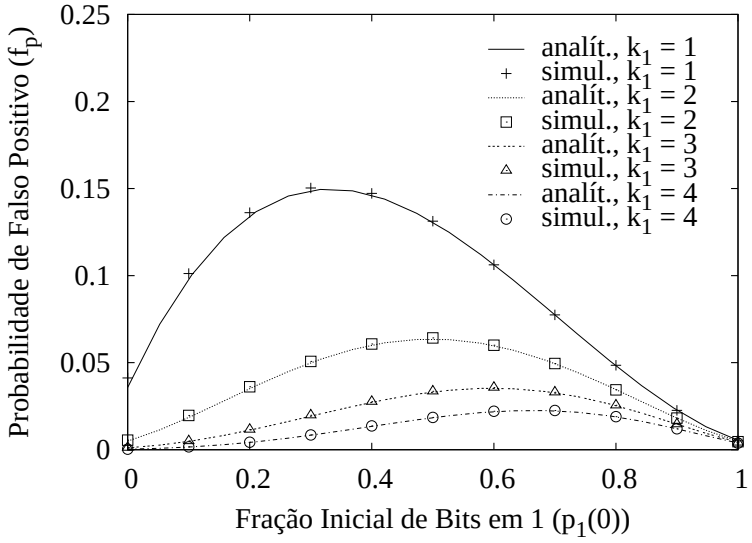


Figura 4.14: Probabilidade de falso positivo de um FBG em função de $p_1(0)$, para $k_0 = 2$, $m/n = 25$.

está descrito na Equação 4.21, o qual leva à máxima probabilidade de falso positivo F_p. Quando um FBG atinge a fração de bits descrita na Equação 4.21, diz-se que o filtro atingiu o chamado *estado estacionário*. Uma vez no estado estacionário, não importa quantos elementos são inseridos no FBG, a fração de bits em 0 e em 1 permanece constante e a probabilidade de falso positivo é F_p. A Figura 4.15 comprova este resultado. Na figura,

podem ser notadas as duas maneiras de um FBG atingir o estado estacionário. Pode-se configurar a condição inicial do filtro de acordo com a Equação 4.27 ou pode-se inserir muitos elementos no filtro. Em ambos os casos, a probabilidade de falso positivo atinge o seu ponto máximo.

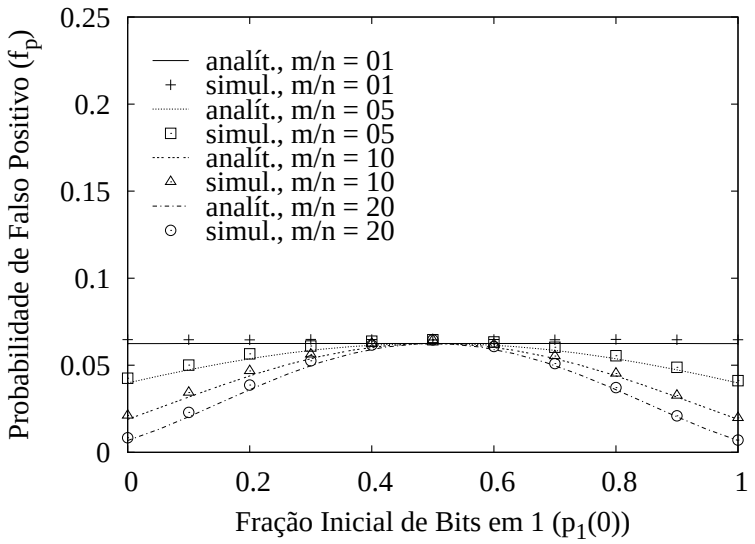


Figura 4.15: Probabilidade de falso positivo de um FBG em função de $p_1(0)$, usando $k_0 = k_1 = 2$.

Capítulo 5

Um Procedimento Aprimorado de Reconstrução de Rota

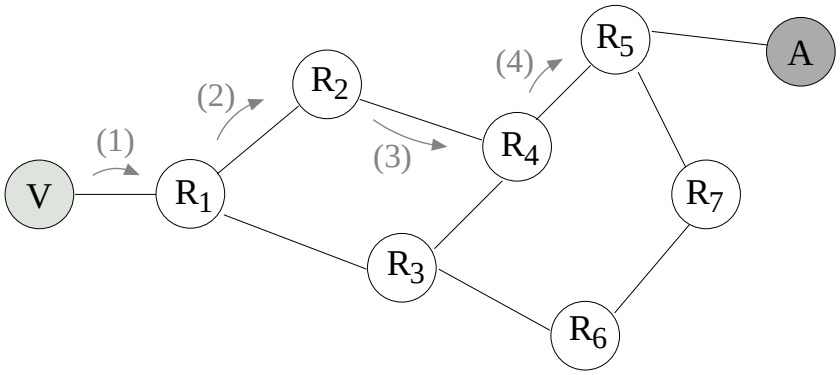
Se por um lado o uso de um Filtro de Bloom Generalizado (FBG) limita a ação do atacante na geração de falsos positivos, por outro, ele introduz falsos negativos no procedimento de reconstrução de rota. Um falso negativo significa não detectar um roteador por onde o pacote realmente passou. Ao deixar 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 3.2 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 precocemente 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 de que, durante o encaminhamento do pacote pela rede, roteadores subseqüentes podem inverter os bits marcados por roteadores anteriores e causar falsos negativos durante a reconstrução. No exemplo da Figura 3.2, durante o percurso do pacote de ataque por (R_5, R_4, R_2, R_1), ou R_1 ou R_2 inver-

teu 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 Filtro de Bloom Generalizado (FBG) algumas informações adicionais para os seus vizinhos durante a reconstrução. Essas informações contêm os bits marcados pelo próprio roteador e pelos roteadores anteriores no procedimento de reconstrução. Assim, cada roteador sabe se um determinado bit pode ter sido invertido por um roteador subseqüente durante a travessia do pacote.

A Figura 5.1 ilustra brevemente o conceito do procedimento aprimorado de reconstrução de rota. Na figura, suponha que o FBG utiliza $k_0 = k_1 = 1$. Primeiramente, a vítima V confirma a presença de R_1 no FBG recebido pelo pacote e, portanto, lhe repassa o FBG de forma a iniciar a reconstrução (1). Entretanto, V também envia outros dois vetores de bits, m_0 e m_1, inicialmente zerados, que possuem o mesmo tamanho que o FBG, conforme descrito em (a). Em seguida, o roteador R_1 atualiza m_0 e m_1 de acordo com os bits da interface por onde a requisição de reconstrução chegou. Desta forma, os bits que a interface zera durante o procedimento de marcação são preenchidos com 1 em m_0 e os bits que a interface preenche durante a marcação são colocados em 1 no vetor m_1. Os vetores atualizados se encontram em (b). Posteriormente, R_1 testa os seus vizinhos. O processo de verificação dos vizinhos se altera um pouco em virtude da informação adicional recebida. Quando um vizinho não é reconhecido pelo FBG, R_1 não descarta aquele vizinho diretamente. Ao invés disso, o roteador R_1 verifica se ele próprio inverteu algum bit marcado por este vizinho durante a travessia do pacote. Para isso, ele verifica se os bits invertidos deste vizinho estão marcados em m_0 ou m_1, dependendo se o bit do vizinho está em 1 ou em 0, respectivamente. Caso os bits invertidos do vizinho estejam marcados como reescritos, então o vizinho é reconhecido como um elemento do filtro. Caso contrário, aquele vizinho é então descartado. Desta forma, R_1 reconhece a presença de R_2. O roteador R_1 então lhe repassa o FBG junto com os dois vetores m_0 e m_1 atualizados (2). Por sua vez, o roteador R_2 atualiza estes vetores de acordo com a interface por onde a requisição chegou, conforme mostrado em (c). Em seguida, o roteador R_2 testa a presença de R_4 no FBG. Os vetores m_0 e m_1 são usados para verificar se algum bit marcado por R_4 foi invertido por R_1 ou R_2. Neste caso, verifica-se que um bit de R_4 está invertido. Um bit que supostamente deveria estar em 0, está em 1. Entretanto, este

bit está marcado no vetor m_1, o que significa que ele foi sobrescrito por R_1 ou R_2 durante a travessia do pacote. Assim, o roteador R_4 não é mais um falso negativo e é reconhecido como um elemento do filtro. O roteador R_2 então lhe repassa o próprio FBG e os dois vetores de bits adicionais indicando as marcações feitas por ele próprio e por R_1 (3). O roteador R_4 faz então o mesmo procedimento com seu vizinho R_5 (4). Pode ser visto em (d) os vetores atualizados por R_4. Neste caso, é fácil perceber o bit zerado por R_4 que foi preenchido por R_2 durante a travessia do pacote pela rede.



	(a)	(b)	(c)	(d)
FBG	0 0 1 1 0 1	0 0 1 1 0 1	0 0 1 1 0 1	0 0 1 1 0 1
m_0	0 0 0 0 0 0	0 1 0 0 0 0	1 1 0 0 0 0	1 1 0 1 0 0
m_1	0 0 0 0 0 0	0 0 0 0 0 1	0 0 0 1 0 1	0 0 1 1 0 1

Figura 5.1: O procedimento aprimorado de reconstrução de rota.

Uma importante vantagem do procedimente 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 de simulação comprovam que isto não é um problema grave e que o atacante pode ser facilmente identificado, conforme mostrado na Seção 5.3.

5.1 Falsos Positivos

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

$$s_0(i) = 1 - (1 - q_0)^i . \tag{5.1}$$

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

$$s_1(i) = 1 - (1 - q_1)^i . \tag{5.2}$$

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 quanto 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

$$t_0(i) = p_0(n) + p_1(n)s_1(i). \tag{5.3}$$

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

$$t_1(i) = p_1(n) + p_0(n)s_0(i). \tag{5.4}$$

A partir das Equações (5.3) e (5.4), 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

$$f_p(i) = t_0(i)^{b_0} t_1(i)^{b_1}, \tag{5.5}$$

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 que um falso positivo ocorra, conforme descrito no Capítulo 4.

5.2 Resultados Analíticos

A Figura 5.2 ilustra a probabilidade de um roteador ser incluído erroneamente, devido a um falso positivo do FBG, na rota de ataque em função da distância i, em número de saltos da vítima, de acordo com a Equação 5.5. Para cada curva, o pior caso para a condição inicial do filtro é suposto de acordo com a Equação 4.27. De forma a facilitar o entendimento, usou-se o mesmo número de funções que colocam bits em 0 e em 1 no filtro, ou seja, $k = k_0 = k_1$. Pela figura, pode ser visto que, quanto mais afastado um roteador se encontra da vítima, maior é a probabilidade de falso positivo. Isso ocorre porque, conforme citado anteriormente, as frações de bits preenchidos dos vetores m_0 e m_1 aumenta à medida que mais roteadores são integrados à rota de ataque. Além disso, é possível observar que um aumento no número de funções diminui a probabilidade de falso positivo até um certo ponto. A partir deste valor, a probabilidade f_p tende a aumentar. Na verdade, para cada tamanho do filtro usado, um valor ótimo para k_0 e k_1 existe. O compromisso neste caso é que um grande número de funções *hash* implica uma maior fração de bits marcados em m_0 e m_1 a cada salto e, portanto, uma maior probabilidade de falso positivo. Por outro lado, quanto maior é o número de funções *hash*, é mais provável que seja encontrado um bit invertido no FBG que não está marcado em m_0 ou m_1 e, portanto, uma menor probabilidade de falso positivo é esperada.

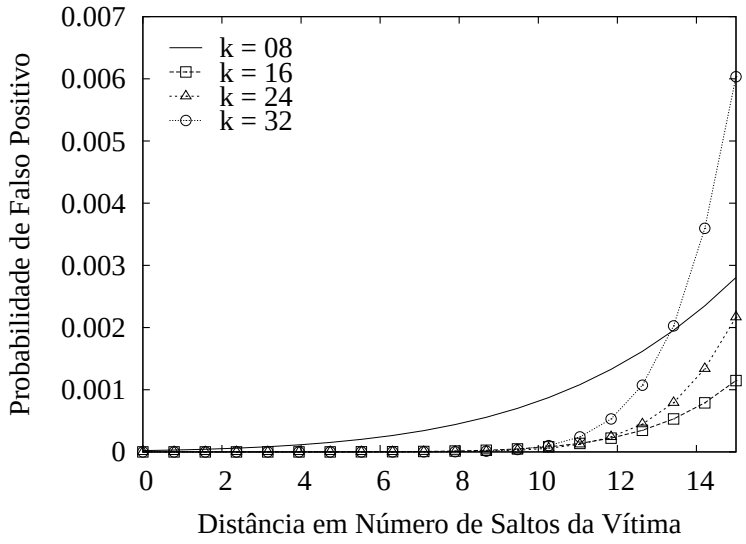


Figura 5.2: Probabilidade de falso positivo de um FBG em função da distância do roteador até a vítima, em número de saltos, para $m = 256$, $n = 15$ e $p_1(0) = 0.5$.

É importante ressaltar mais uma vez que, com o procedimento aprimorado de reconstrução de rota, a probabilidade de falso negativo é sempre zero, não importando a que distância o roteador está da vítima. Comparando a Figura 4.7 com a Figura 5.2, é possível perceber que os falsos negativos do procedimento padrão de reconstrução aumentam mais rapidamente que os falsos positivos do procedimento aprimorado. Além disso, os falsos negativos têm um impacto maior na reconstrução do que os falsos positivos. Portanto, o procedimento aprimorado de reconstrução de rota é a melhor escolha a ser adotada.

De forma a mostrar a eficiência do procedimento aprimorado de reconstrução contra ações do atacante, a Figura 5.3 ilustra a probabilidade um roteador ser incluído erroneamente, devido a um falso positivo do FBG, na rota de ataque em função da sua distância i, em número de saltos, da vítima usando diferentes condições iniciais do filtro. Conforme pode ser visto, a ação do atacante para aumentar os falsos positivos também é limitada. Esta limitação é causada pelas mesmas razões que limitam o atacante no procedimento de reconstrução padrão. Para que um falso positivo ocorra, são necessários bits em 0 e bits em 1 e, se eles estão distribuídos proporcionalmente conforme a Equação 4.21, a probabilidade de falso positivo aumenta.

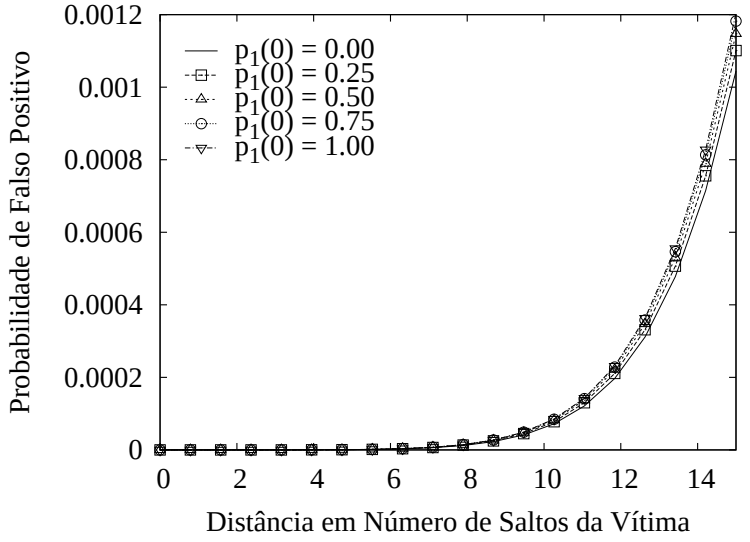


Figura 5.3: Probabilidade de falso positivo de um FBG em função da distância do roteador até a vítima, em número de saltos, para $m = 256$, $n = 15$ e $k_0 = k_1 = 16$.

5.3 Resultados de Simulação

De forma a analisar o comportamento do procedimento aprimorado de reconstrução, foi desenvolvido um simulador em C++ [73]. Inicialmente, para criar a topologia usada na simulação, foi usado o gerador de topologia *nem* [74], baseado em amostragem do mapa da Internet. O gerador *nem* extrai aleatoriamente um subgrafo de um mapa de rede, mantendo suas propriedades originais, como grau de vizinhos, distância média e diâmetro da topologia. A topologia empregada consiste de 10.000 roteadores com uma média de 3,15 vizinhos por roteador e é amostrada de uma topologia real da Internet. Uma vez gerada a topologia, um atacante é escolhido aleatoriamente de um conjunto de roteadores de borda. Isso reflete a suposição de que roteadores de núcleo não são comprometidos e que é mais provável que o(s) atacante(s) esteja(m) em uma rede local fora do *backbone* da Internet. Em seguida, é definida uma rota de ataque sem *loops* a partir do roteador de borda escolhido. Posteriormente, é simulado o envio de um pacote de ataque através da marcação do FBG do pacote de acordo com os roteadores escolhidos para compor a rota de ataque. A condição inicial de pior caso é sempre utilizada para inicializar o filtro, de acordo com a Equação 4.27. Sendo o filtro marcado, o processo de reconstrução de rota é iniciado a partir da vítima. Dois procedimentos de reconstrução são usados, de forma

a compará-los. O primeiro é o procedimento padrão, onde somente o filtro é usado para a verificação dos vizinhos pelos roteadores, e o segundo é o procedimento descrito neste capítulo, onde os vetores m_0 e m_1 adicionais também são repassados a cada salto. Para cada ponto medido, foi calculado o intervalo de confiança, representado por barras de erro verticais, usando uma confiabilidade de 95%.

A Figura 5.4 mostra os resultados do procedimento de reconstrução padrão para uma rota típica de quinze roteadores. Na figura, é mostrado o tamanho da rota rastreada em função do número de funções *hash* utilizadas no filtro ($k = k_0 = k_1$). Pode-se constatar que o procedimento de reconstrução padrão apresenta um tamanho de rota rastreada menor que o tamanho da rota original. Para rotas de ataque de quinze roteadores, só foram rastreados cerca de sete roteadores. Este comportamento ocorre devido à existência de falsos negativos na reconstrução da rota. No caso de um falso negativo, um roteador por onde o pacote realmente passou não é reconhecido na reconstrução e o procedimento é interrompido precocemente. Para reduzir a probabilidade de um falso negativo, é necessário aumentar muito o tamanho do filtro, o que aumenta a sobrecarga de dados necessários para o rastreamento. Pode-se constatar também que à medida que se aumenta o número k de funções *hash*, o tamanho da rota rastreada diminui. Tal comportamento ocorre porque a probabilidade de falso negativo aumenta conforme se aumenta o número de funções *hash*. Logo, a probabilidade do procedimento de reconstrução ser interrompido mais cedo é ainda maior.

O procedimento aprimorado de reconstrução de rota descrito neste capítulo não apresenta falsos negativos e, conseqüentemente, a rota de ataque é sempre encontrada. Entretanto, devido aos falsos positivos, outras rotas que levam a falsos atacantes também são encontradas. Conforme a definição anterior, um atacante é definido como um roteador onde o procedimento de reconstrução termina, ou seja, nenhum vizinho é reconhecido. Como forma de avaliar o desempenho do procedimento aprimorado, é usado como métrica o número médio de atacantes rastreados. Idealmente, a reconstrução de rota a partir de um pacote de ataque deve levar a somente um atacante, o verdadeiro atacante.

A Figura 5.5 mostra o número de atacantes encontrados pelo procedimento aprimorado em função do número de funções *hash* usadas ($k = k_0 = k_1$). Pode-se perceber que,

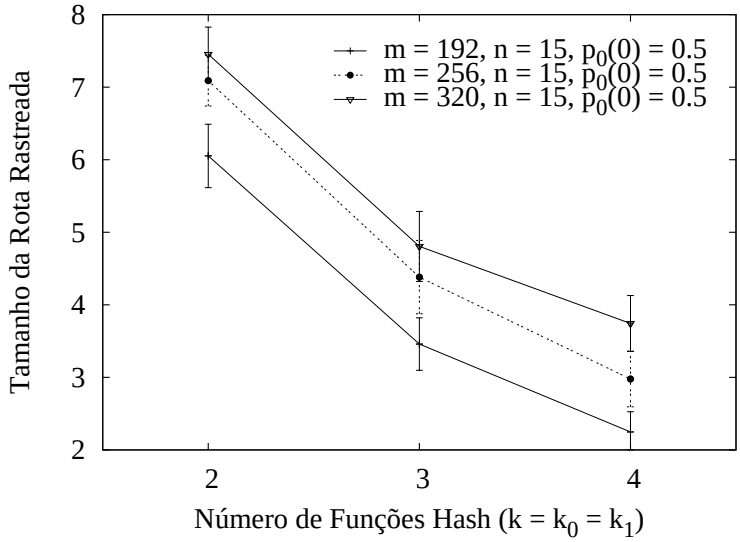


Figura 5.4: Tamanho da rota reconstruída em função das funções *hash*.

para o mesmo tamanho da rota de ataque e para os mesmos tamanhos de filtro usados no procedimento padrão, o número de possíveis atacantes é reduzido a menos de quatro. Além disso, este número se aproxima muito do caso ideal, onde somente um atacante é encontrado, para filtros de 256 e 320 bits. Nota-se também que existe um compromisso entre o número de funções usadas e o número de atacantes rastreados. É possível perceber que para pequenos valores de k o número de prováveis atacantes resultantes do rastreamento é grande. O mesmo ocorre à medida que aumentamos muito o valor de k. Tal compromisso é conseqüência direta dos falsos positivos no processo de reconstrução. Com poucas funções *hash*, poucos bits em 0 e em 1 precisam ser encontrados para que ocorra um falso positivo. Por outro lado, com muitas funções, a fração de bits marcados em m_0 e m_1 aumenta a cada roteador durante a reconstrução e também leva a uma maior probabilidade de falso positivo. Em ambos os casos, um alto número de falsos positivos leva a mais roteadores reconhecidos como componentes da rota de ataque e, conseqüentemente, a um maior número médio de prováveis atacantes resultantes do rastreamento.

Pode ser observado na Figura 5.6 o número de atacantes rastreados pelo procedimento aprimorado de reconstrução em função do tamanho da rota de ataque. É possível perceber que à medida que o tamanho da rota de ataque aumenta, também aumenta o número de atacantes encontrados. Isso ocorre porque, durante a reconstrução pelo procedimento aprimorado, a probabilidade de falso positivo aumenta à medida que os roteadores são

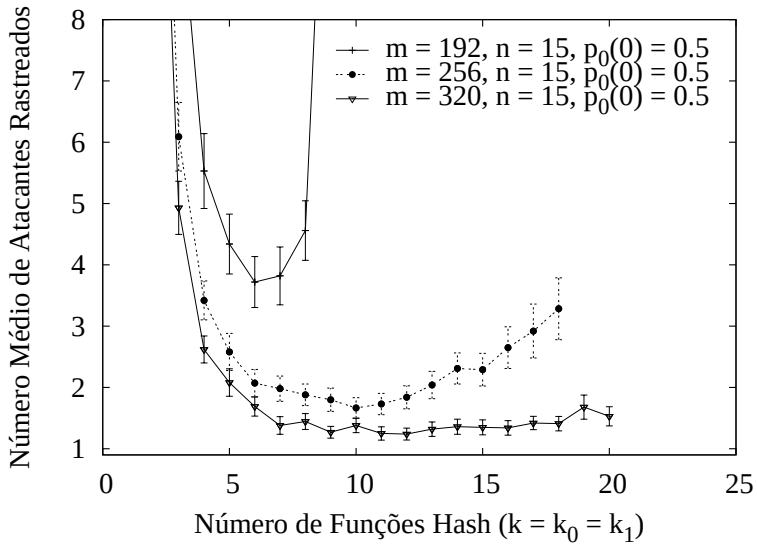


Figura 5.5: Número médio de prováveis atacantes rastreados em função das funções *hash*.

testados mais longe da vítima. Em conseqüência, mais roteadores são reconhecidos como componentes da rota de ataque e mais atacantes são encontrados. Entretanto, ainda é possível encontrar um número de atacantes muito próximo do ideal dependendo do tamanho do filtro utilizado. Com maiores filtros, rotas mais longas podem ser rastreadas.

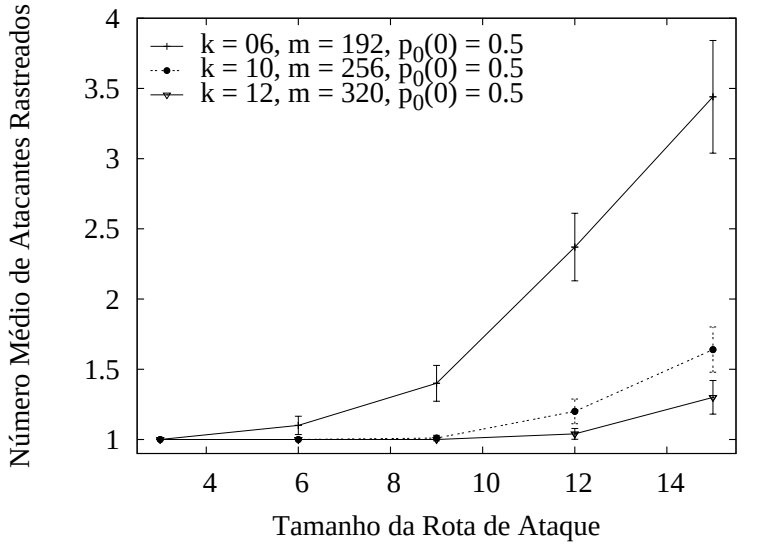


Figura 5.6: Número médio de prováveis atacantes rastreados em função do tamanho da rota de ataque.

Desta forma, para uma rota de 15 roteadores, um filtro de 256 bits identifica em média 1,7 possíveis atacantes. Considerando que o atacante real sempre está dentro dos pos-

síveis atacantes rastreados, esse valor pode ser considerado adequado. Comparando o espaço necessário para se armazenar o endereço IP de 15 roteadores (15 roteadores $\times$ 32 bits/roteador $=$ 480 bits), pode-se verificar que 256 bits apresenta uma economia de 47% de espaço no cabçalho do pacote.

Capítulo 6

Conclusões

ATAQUES de negação de serviço têm causado prejuízos consideráveis nos últimos anos. Estes ataques visam deixar os serviços oferecidos por uma determinada vítima inacessíveis para usuários legítimos. No caso de vítimas que dependem da Internet para a viabilidade de seus negócios, os ataques de negação de serviço podem ser a causa de danos financeiros irreparáveis. A motivação para estes ataques vai desde o prestígio e a fama dentro da comunidade virtual a outras razões mais sérias, como motivos financeiros e políticos.

Muitos ataques de negação de serviço são conduzidos a partir de pacotes com endereço de origem forjados, de forma a dificultar a identificação do atacante. Como o roteamento IP é baseado exclusivamente no endereço de destino e nenhum teste é realizado para verificar a autenticidade da origem, pacotes com endereços de origem forjados são roteados sem problemas até o seu destino. Além disso, os roteadores não armazenam nenhuma informação sobre os pacotes encaminhados, o que impossibilita a determinação da rota tomada por um pacote depois do seu recebimento. Atacantes se aproveitam destas características para enviarem pacotes de ataque para a vítima sem precisar se identificar.

Para revelar a identidade de atacantes que usam endereços de origem forjados, alguns sistemas de rastreamento de pacotes foram propostos. Estes sistemas visam identificar a rota percorrida por um determinado conjunto de pacotes, de forma que contramedidas possam ser tomadas o mais próximo possível da origem do ataque e que o tráfego de

usuários legítimos não seja afetado. Dentre os sistemas propostos, destacam-se aqueles capazes de rastrear ataques a partir de um único pacote recebido. Estes sistemas conseguem realizar a difícil tarefa de identificar a origem de ataques por vulnerabilidade, que só precisam de um pacote para negar o serviço da vítima. Além disso, tais sistemas são capazes de identificar todas as possíveis fontes de um ataque distribuído, tornando-se altamente escaláveis. Entretanto, os sistemas propostos que possuem essa capacidade necessitam armazenar informações sobre os pacotes roteados na própria infra-estrutura da rede, o que os torna pouco práticos.

Neste trabalho, é introduzida uma nova abordagem para o rastreamento de pacotes IP que insere dados no cabeçalho dos próprios pacotes. O sistema proposto é capaz de descobrir a origem de um ataque através do rastreamento de um único pacote, sem armazenar nenhuma informação na infra-estrutura da rede. Ao atravessar a rede, o pacote é marcado por cada roteador pertencente a sua rota com uma "assinatura." Dessa forma, a vítima consegue extrair informações do próprio pacote que ajudam na identificação de todos os roteadores componentes da rota de ataque. O sistema proposto usa um Filtro de Bloom no cabeçalho de cada pacote para que a informação inserida seja compacta e apresente um tamanho fixo, evitando assim a adição de dados aos pacotes e possíveis fragmentações. Desta forma, nenhum processamento do roteador é desperdiçado com essas ações. Além disso, com o uso de um Filtro de Bloom, o processamento adicional inserido no roteamento devido à marcação de pacotes é baixo. É mostrado que somente uma operação OU bit-a-bit adicional precisa ser realizada por pacote para que o pacote seja rastreado. Entretanto, o Filtro de Bloom é altamente dependente da sua condição inicial. É mostrado que, ao se usar um Filtro de Bloom no pacote para armazenar a rota de ataque, o atacante facilmente consegue dificultar o processo de rastreamento. Por isso, uma generalização do Filtro de Bloom foi proposta e empregada no sistema. O sistema possui a característica de ser resistente à burla do atacante.

Os dois filtros, o Filtro de Bloom convencional e o Filtro de Bloom Generalizado, foram comparados analiticamente e por meio de simulação a fim de comprovar a eficácia do filtro generalizado. Um importante resultado mostra que a capacidade do atacante de dificultar o processo de rastreamento é drasticamente diminuída. Enquanto para o filtro convencional o atacante consegue atingir uma probabilidade máxima de falso positivo de

100%, para o filtro generalizado, a probabilidade máxima de falso positivo depende exclusivamente de parâmetros escolhidos no projeto do sistema e é no máximo 25%. Ou seja, a probabilidade máxima de falso positivo é diminuída de pelo menos 75% e pode ser diminuída ainda mais dependendo dos parâmetros escolhidos. Em contrapartida, a possibilidade da ocorrência de falsos negativos é introduzida no sistema. Entretanto, também é encontrado um limitante superior para os falsos negativos. Desta forma, é mostrado que a probabilidade de falso negativo pode ser reduzida tanto quanto se queira usando-se filtros generalizados de tamanho maior. Além disso, o Filtro de Bloom Generalizado não é tão dependente da sua condição inicial como o Filtro de Bloom convencional. Com a generalização, é mostrado que a condição inicial afeta somente os falsos positivos e que a capacidade do atacante burlar o sistema de rastreamento é limitada.

De forma a minimizar a probabilidade de falso negativo introduzida pelo Filtro de Bloom Generalizado, um procedimento aprimorado de reconstrução de rota para o rastreamento de pacotes IP também é apresentado. Através deste novo procedimento, cada roteador repassa informações adicionais aos seus vizinhos de forma a notificá-los sobre as marcações realizadas pelos roteadores anteriores no processo de reconstrução. Desta forma, ao verificar a presença de algum roteador vizinho no filtro, cada roteador também verifica se as marcações daquele vizinho não foram alteradas por algum outro roteador. Caso tenham sido alteradas, o roteador é considerado um dos roteadores componentes da rota de ataque. Em caso contrário, ele é simplesmente retirado do procedimento de reconstrução. Este procedimento aprimorado de reconstrução foi comparado com o procedimento padrão por meios analíticos e de simulação. É mostrado que o procedimento proposto elimina a ocorrência de falsos negativos na reconstrução da rota ao custo de um pequeno aumento na taxa de falsos positivos. Além disso, é constatado por simulação que o procedimento aprimorado de reconstrução de rota consegue rastrear o atacante com excelente acurácia enquanto o mesmo não ocorre para o procedimento padrão. Em virtude de falsos negativos, o procedimento padrão torna-se limitado a rotas com um pequeno número de roteadores. Para uma rota com 15 roteadores, somente 7 foram rastreados em média. No procedimento aprimorado, como não há falsos negativos, é mostrado ainda que o atacante é sempre identificado e que esta identificação é muito próxima da ideal, onde somente um atacante é encontrado. Para uma rota com 15 roteadores e um filtro de

256 bits, chegou-se a um número médio de 1,7 possíveis atacantes. Além disso, é mostrado também que o atacante não consegue burlar o rastreamento quando o procedimento aprimorado de reconstrução de rota é usado.

Alguns compromissos interessantes puderam ser observados com os resultados de simulação do procedimento aprimorado de reconstrução de rota. Primeiramente, é mostrado que existe um compromisso entre o número de funções *hash* utilizadas pelo Filtro de Bloom Generalizado e o número médio de prováveis atacantes rastreados. Desta forma, existe um número ótimo de funções *hash* que minimiza a probabilidade de falso positivo e, conseqüentemente, o número médio de atacantes. Além disso, existe uma relação interessante entre o tamanho do filtro e o tamanho da rota que se quer rastrear. Para maiores rotas ou para uma maior acurácia, é necessário o uso de um filtro maior, enquanto que para rotas menores ou menor acurácia, um filtro menor é suficiente. Em ambos os casos, a economia de espaço devido ao uso do Filtro de Bloom Generalizado é significativa. Para uma rota de 15 roteadores, um filtro de 256 bits é suficiente para se rastrear o atacante, o que apresenta uma economia de 47% de espaço no cabeçalho do pacote.

Alguns trabalhos futuros incluem o estudo da compressão para o Filtro de Bloom Generalizado e de outras técnicas usadas na reconstrução para melhorar ainda mais a acurácia da rota de ataque reconstruída. Um importante trabalho explicado anteriormente sugere a compressão de Filtros de Bloom convencionais antes da sua transmissão. Um resultado interessante mostra que, para um tamanho de transmissão fixo, é sempre vantajoso usar filtros maiores nas estações finais e comprimí-los antes da transmissão. Um estudo sobre a possibilidade de compressão do Filtro de Bloom Generalizado seria interessante para melhorar a acurácia do resultado. Um filtro maior poderia ser usado para o rastreamento usando-se menos bits por pacote. O custo seria o processamento de compressão e descompressão a cada salto. Ou seja, cada roteador precisaria descomprimir o filtro carregado pelo pacote, atualizá-lo de acordo com o seu endereço IP e comprimí-lo novamente antes de reencaminhá-lo.

Uma outra pesquisa que poderia ser realizada para melhorar a acurácia da rota reconstruída é considerar a maior rota do grafo reconstruído como sendo a rota de ataque. Com o procedimento de reconstrução aprimorado, um grafo é reconstruído, contendo somente

um atacante e possivelmente outros falsos positivos. Entretanto, de acordo com observações realizadas a partir dos resultados de simulação, estes falsos positivos são geralmente roteadores a um salto de algum roteador componente da verdadeira rota de ataque. Ou seja, o grafo reconstruído é geralmente composto por uma longa cadeia de roteadores e por algumas arestas partindo desta cadeia que são falsos positivos. Desta forma, é possível aproximar-se ainda mais do caso ideal onde somente um atacante é rastreado.

Referências Bibliográficas

[1] CERT. *CERT Advisory CA-1996-01 UDP Port Denial-of-Service Attack*, fevereiro de 1996. http://www.cert.org/advisories/CA-1996-01.html.

[2] CERT. *CERT Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks*, setembro de 1996. http://www.cert.org/advisories/CA-1996-21.html.

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

[4] CERT. *CERT Advisory CA-2000-21 Denial-of-Service Vulnerabilities in TCP/IP Stacks*, novembro de 2000. http://www.cert.org/advisories/CA-2000-21.html.

[5] MIRKOVIC, J., ROBINSON, M., E REIHER, P. Alliance Formation for DDoS Defense. Em *Proceedings of the 2003 Workshop on New Security Paradigms* (Ascona, Suíça, 2003), pág. 11–18.

[6] MOORE, D., VOELKER, G., E SAVAGE, S. Inferring Internet Denial of Service Activity. Em *Proceedings of the 2001 USENIX Security Symposium* (Washington, DC, EUA, agosto de 2001), pág. 9–22.

[7] GORDON, L. A., LOEB, M. P., LUCYSHYN, W., E RICHARDSON, R. *2005 CSI/FBI Computer Crime and Security Survey*, 2005.

[8] CERT. *CERT Advisory CA-1999-17 Denial-of-Service Tools*, dezembro de 1999. http://www.cert.org/advisories/CA-1999-17.html.

[9] CERT. *CERT Advisory CA-2000-01 Denial-of-Service Developments*, janeiro de 2000. http://www.cert.org/advisories/CA-2000-01.html.

[10] CERT. *CERT Advisory CA-2003-20 W32/Blaster worm*, agosto de 2003. http://www.cert.org/advisories/CA-2003-20.html.

[11] SYMANTEC SECURITY RESPONSE. *W32.Mydoom.F@mm*, fevereiro de 2004. http://securityresponse.symantec.com/avcenter/venc/data/w32.mydoom.f@mm.html.

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

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

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

[15] CISCO SYSTEMS, INC. *Cisco Security Advisory: Cisco IOS Interface Blocked by IPv4 Packets*, julho de 2003. http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml.

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

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

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

[19] CHANG, R. K. C. Defending against Flooding-Based Distributed Denial-of-Service Attacks: A Tutorial. *IEEE Communications Magazine 40*, 10 (outubro de 2002), 42–51.

[20] SAVAGE, S., WETHERALL, D., KARLIN, A., E ANDERSON, T. Network Support for IP Traceback. *IEEE/ACM Transactions on Networking 9*, 3 (junho de 2001), 226–237.

[21] BELLOVIN, S. M., LEECH, M. D., E TAYLOR, T. ICMP Traceback Messages. *Internet Draft: draft-ietf-itrace-04.txt* (agosto de 2003).

[22] DEAN, D., FRANKLIN, M., E STUBBLEFIELD, A. An Algebraic Approach to IP Traceback. *ACM Transactions on Information and System Security 5*, 2 (maio de 2002), 119–137.

[23] LAUFER, R. P., VELLOSO, P. B., E DUARTE, O. C. M. B. A Stateless Scheme for Single-Packet IP Traceback. Em *IEEE Infocom 2005 - Student Workshop* (Miami, FL, EUA, março de 2005).

[24] LAUFER, R. P., VELLOSO, P. B., E DUARTE, O. C. M. B. Defeating DoS Attacks with IP Traceback. Em *IFIP Open Conference on Metropolitan Area Networks - MAN'2005* (Ho Chi Minh, Vietnã, abril de 2005), pág. 131–148.

[25] LAUFER, R. P., VELLOSO, P. B., DE O. CUNHA, D., MORAES, I. M., BICUDO, M. D. D., E DUARTE, O. C. M. B. A New IP Traceback System against Denial-of-Service Attacks. Em *12th International Conference on Telecommunications - ICT'2005* (Cidade do Cabo, África do Sul, maio de 2005).

[26] LAUFER, R. P., VELLOSO, P. B., E DUARTE, O. C. M. B. 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, maio de 2005).

[27] LAUFER, R. P., VELLOSO, P. B., DE O. CUNHA, D., E DUARTE, O. C. M. B. Um Procedimento Alternativo de Reconstrução de Rota para o Rastreamento de Pacotes IP. Em *XXII Simpósio Brasileiro de Telecomunicações - SBrT'05* (Campinas, SP, Brasil, setembro de 2005), pág. 1013–1018.

[28] LAUFER, R. P., MORAES, I. M., VELLOSO, P. B., BICUDO, M. D. D., CAMPISTA, M. E. M., DE O. CUNHA, D., COSTA, L. H. M. K., E DUARTE, O. C. M. B. Negação de Serviço: Ataques e Contramedidas. Em *Minicursos do V Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais - SBSeg'2005*. setembro de 2005, cap. 1.

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

[30] MIRKOVIC, J., DIETRICH, S., DITTRICH, D., E REIHER, P. *Internet Denial of Service: Attack and Defense Mechanisms*. Prentice Hall PTR, 2004.

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

[32] DITTRICH, D. *The DoS Project's 'trinoo' distributed denial of service attack tool*, outubro de 1999. http://staff.washington.edu/dittrich/misc/trinoo.analysis.txt.

[33] DITTRICH, D. *The 'Tribe Flood Network' distributed denial of service attack tool*, outubro de 1999. http://staff.washington.edu/dittrich/misc/tfn.analysis.txt.

[34] DITTRICH, D. *The 'stacheldraht' distributed denial of service attack tool*, dezembro de 1999. http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.txt.

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

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

[37] SHACHTMAN, N. *Porn Purveyors Getting Squeezed*. Wired News, julho de 2003. http://wired-vig.wired.com/news/print/0,1294,59574,00.html.

[38] GRAY, P., E FRIED, I. *Al-Jazeera suffers DoS attack*. ZDNet UK, março de 2003. http://news.zdnet.co.uk/business/0,39020645,2132585,00.htm.

[39] SCHUBA, C. L., KRSUL, I. V., KUHN, M. G., SPAFFORD, E. H., SUNDARAM, A., E ZAMBONI, D. Analysis of a Denial of Service Attack on TCP. Em *Proceedings of the 1997 IEEE Symposium on Security and Privacy* (Oakland, CA, EUA, maio de 1997), pág. 208–223.

[40] KEYES, R. *The Naptha DoS Vulnerability*. BindView Corp., novembro de 2000. http://www.bindview.com/Services/razor/Advisories/2000/adv_NAPTHA.cfm.

[41] POSTEL, J. Character Generator Protocol. *RFC 864* (maio de 1983).

[42] WAGNER, J. *Dealing With Massive Attack: DNS Protection*. Internetnews.com, outubro de 2002. http://www.internetnews.com/dev-news/article.php/1487331.

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

[44] MIRKOVIC, J., E REIHER, P. A taxonomy of DDoS Attack and DDoS Defense Mechanisms. *ACM SIGCOMM Computer Communications Review 34*, 2 (2004), 39–53.

[45] FERGUSON, P., E SENIE, D. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. *RFC 2827* (maio de 2000).

[46] PERKINS, C. E. IP Mobility Support for IPv4. *RFC 3220* (janeiro de 2002).

[47] LI, J., MIRKOVIC, J., WANG, M., REIHER, P., E ZHANG, L. SAVE: Source Address Validity Enforcement Protocol. Em *Proceedings of the IEEE INFOCOM 2002 Conference* (Nova Iorque, NY, EUA, junho de 2002), pág. 1557–1566.

[48] SNOEREN, A. C., PARTRIDGE, C., SANCHEZ, L. A., JONES, C. E., TCHAKOUNTIO, F., SCHWARTZ, B., KENT, S. T., E STRAYER, W. T. Single-Packet IP Traceback. *IEEE/ACM Transactions on Networking 10*, 6 (dezembro de 2002), 721–734.

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

[50] BURCH, H., E CHESWICK, B. Tracing Anonymous Packets to their Approximate Source. Em *USENIX LISA'00* (Nova Orleans, LA, EUA, dezembro de 2000), pág. 319–327.

[51] BELENKY, A., E ANSARI, N. On IP Traceback. *IEEE Communications Magazine 41*, 7 (julho de 2003), 142–153.

[52] POSTEL, J. Internet Protocol. *RFC 791* (setembro de 1981).

[53] SONG, D. X., E PERRIG, A. Advanced and Authenticated Marking Schemes for IP Traceback. Em *Proceedings of the IEEE INFOCOM 2001 Conference* (Anchorage, AK, EUA, abril de 2001).

[54] PARK, K., E LEE, H. 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, abril de 2001).

[55] MANKIN, A., MASSEY, D., WU, C.-L., WU, S. F., E ZHANG, L. On Design and Evaluation of "Intention-Driven" ICMP Traceback. Em *Proceedings of the IEEE ICCCN 2001 Conference* (Scottsdale, AZ, EUA, outubro de 2001).

[56] BRODER, A., E MITZENMACHER, M. Network Applications of Bloom Filters: A Survey. *Internet Mathematics 1*, 4 (2003), 485–509.

[57] LI, J., SUNG, M., XU, J., E LI, L. 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, maio de 2004).

[58] CAIDA. *Packet Fragmentation on the UCSD-CERF Link*, junho de 2002. http://www.caida.org/analysis/workload/fragments/sdscposter.xml.

[59] FAN, L., CAO, P., ALMEIDA, J., E BRODER, A. Z. Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. *IEEE/ACM Transactions on Networking 8*, 3 (junho de 2000), 281–293.

[60] MARGOLIASH, D. J. CSPELL - A Bloom Filter-Based Spelling Correction Program. Tese de Mestrado, The University of Western Ontario, London, ON, Canadá, junho de 1987. Departamento de Ciência da Computação.

[61] MITZENMACHER, M. Compressed Bloom Filters. *IEEE/ACM Transactions on Networking 10*, 5 (outubro de 2002), 604–612.

[62] SEVERANCE, D. G., E LOHMAN, G. M. Differential Files: Their Application to the Maintanance of the Large Databases. *ACM Transactions on Database Systems 1*, 3 (setembro de 1976), 256–267.

[63] SPAFFORD, E. H. OPUS: Preventing Weak Password Choices. *Computers and Security 11*, 3 (maio de 1992), 273–278.

[64] DHARMAPURIKAR, S., KRISHNAMURTHY, P., SPROULL, T. S., E LOCKWOOD, J. W. Deep Packet Inspection Using Bloom Filters. *IEEE Micro 24*, 1 (janeiro de 2004), 52–61.

[65] DHARMAPURIKAR, S., KRISHNAMURTHY, P., E TAYLOR, D. E. Longest Prefix Matching Using Bloom Filters. Em *Proceedings of the ACM SIGCOMM'03 Conference* (Karlsruhe, Alemanha, agosto de 2003), pág. 201–212.

[66] ESTAN, C., E VARGHESE, G. New Directions in Traffic Measurement and Accounting: Focusing on the Elephants, Ignoring the Mice. *ACM Transactions on Computer Systems 21*, 3 (2003), 270–313.

[67] COHEN, S., E MATIAS, Y. Spectral Bloom Filters. Em *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data* (San Diego, CA, EUA, junho de 2003), pág. 241–252.

[68] KUMAR, A., XU, J., WANG, J., SPATSCHEK, O., E LI, L. Space-Code Bloom Filter for Efficient Per-Flow Traffic Measurement. Em *Proceedings of the IEEE INFOCOM 2004 Conference* (Hong Kong, China, março de 2004), pág. 1762–1773.

[69] SHANMUGASUNDARAM, K., BRÖNNIMANN, H., E MEMON, N. Payload Attribution via Hierarchical Bloom Filters. Em *Proceedings of the 11th ACM conference on Computer and Communications Security* (Washington, DC, EUA, outubro de 2004), pág. 31–41.

[70] RHEA, S. C., E KUBIATOWICZ, J. Probabilistic Location and Routing. Em *Proceedings of the IEEE INFOCOM 2002 Conference* (Nova Iorque, NY, USA, junho de 2002), pág. 1248–1257.

[71] LAUFER, R. P. Generalized Bloom Filter (GBF) Implementation. http://www.gta.ufrj.br/~rlaufer/gbf.

[72] RAMAKRISHNA, M. V. Practical Performance of Bloom Filters and Parallel Free-Text Searching. *Communications of the ACM 32*, 10 (outubro de 1989), 1237–1239.

[73] LAUFER, R. P. GBF Traceback Simulator.
http://www.gta.ufrj.br/~rlaufer/gbf_traceback.

[74] MAGONI, D., E PANSIOT, J.-J. Internet Topology Modeler Based on Map Sampling. Em *Proceedings of the 7th IEEE International Symposium on Computer Communications* (Washington, DC, EUA, julho de 2002), pág. 1021.