

Um Procedimento Alternativo de Reconstrução de Rota para o Rastreamento de Pacotes IP

Rafael P. Laufer, Pedro B. Velloso, Daniel de O. Cunha e Otto Carlos M. B. Duarte

Resumo—Uma possível estratégia de defesa contra ataques de negação de serviço é rastrear a origem de cada pacote de ataque, de forma a penalizar o atacante ou isolá-lo da rede. Um sistema de rastreamento proposto previamente pelos autores sugere que os roteadores insiram informações nos pacotes roteados, notificando a vítima sobre a sua presença na rota. Com a informação recebida, a vítima inicia um procedimento de reconstrução de rota para identificar o verdadeiro atacante. Neste artigo, é proposto um procedimento alternativo de reconstrução de rota para o rastreamento de pacotes IP. É mostrado através de resultados analíticos e de simulação que o procedimento proposto é eficaz na identificação o atacante.

Palavras-Chave—Rastreamento de Pacotes, Negação de Serviço, Filtro de Bloom, Segurança de Redes.

Abstract—A defense strategy against denial-of-service attacks is to trace the source of every attack packet for the sake of penalizing the attacker or isolating him from the network. An IP traceback scheme proposed by the authors suggests that routers notify the victim of their presence in the attack path by inserting traceback information on routed packets. With the received information, the victim initiates a reconstruction procedure to identify the true source of the attack. In this paper, an alternative reconstruction procedure is proposed for IP traceback. We show through analytical and simulation results that the proposed reconstruction procedure traces the attack back to its true source, being effective in the identification of the attacker.

Keywords—IP Traceback, Denial of Service, Bloom filters, Computer network security.

I. INTRODUÇÃO

A infra-estrutura de roteamento atual ainda é vulnerável a ataques de negação de serviço (*Denial of Service* - DoS) anônimos [1]. Tais ataques são caracterizados pelo completo desconhecimento da sua verdadeira origem e visam tornar inacessíveis os serviços providos pela vítima. Este objetivo geralmente é alcançado através do envio de pacotes a uma taxa maior do que podem ser servidos, fazendo com que legítimas requisições ao serviço não sejam atendidas. Em sua versão distribuída (*Distributed DoS*), os pacotes são enviados de diferentes origens e o tráfego agregado gerado é responsável pela total inutilização dos serviços da vítima. Ultimamente, o número de ataques distribuídos a grandes sítios é cada vez mais alarmante, sendo inclusive desenvolvidas pragas digitais específicas para tal finalidade [2]. Embora menos comum, também existem ataques de negação de serviço constituídos pelo envio de um único pacote [3], [4]. Em ambos os casos,

os resultados são desastrosos [5] e a necessidade de uma solução que identifique a verdadeira origem dos pacotes se torna evidente.

Devido à técnica datagrama usada no protocolo IP, 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 consequência, o encaminhamento de pacotes não deixa “rastros”, tornando impossível deduzir a rota percorrida por um pacote.

Ataques de um único pacote já realizados no passado reforçam a necessidade de sistemas que consigam realizar o rastreamento a partir de somente um pacote [3], [4]. Ao invés de inundar a vítima com falsas requisições ao serviço, tais ataques são baseados em vulnerabilidades do sistema operacional e são ativadas por um pacote com determinadas características [6]. Desta forma, este tipo de ataque é bem mais fácil de ser realizado se comparado aos ataques de inundação. São também mais difíceis de rastrear, uma vez que a maioria dos sistemas de rastreamento propostos exige um número mínimo de pacotes de ataque recebidos. Além disso, ataques distribuídos baseados em inundação podem ser realizados através do envio de um único pacote por atacante. Se um número suficientemente grande de atacantes conspira para atacar uma vítima em comum, o resultado pode ser alcançado se cada atacante enviar um único pacote para a vítima em um determinado espaço de tempo. Assim, teoricamente, sistemas de rastreamento de pacotes devem ser sempre capazes de rastrear ataques de um único pacote.

Sistemas de rastreamento propostos sugerem que os roteadores notifiquem a vítima sobre a sua presença no rota de ataque. Tal notificação pode ser realizada pelos roteadores através da inserção de informação sobre si próprio nos pacotes roteados. Dessa forma, a vítima reconhece a presença daquele roteador específico na rota de ataque. 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.

Neste artigo, é introduzido um procedimento alternativo de reconstrução de rota para um sistema de rastreamento de pacotes IP. Uma discussão completa dos detalhes do sistema de rastreamento e uma avaliação analítica do seu desempenho foram apresentadas previamente [7]. Tal sistema possui a grande vantagem de ser capaz de encontrar o atacante a

Rafael P. Laufer, Daniel de O. Cunha e Otto Carlos M. B. Duarte, Grupo de Telemática e Automação (GTA), Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil, E-mails: {rlaufer,doc,otto}@gta.ufrj.br.

Pedro B. Velloso, Laboratoire d'Informatique de Paris 6 (LIP6), Université Pierre et Marie Curie - Paris VI, Paris, França, E-mail: pedro.velloso@lip6.fr.

Este trabalho foi realizado com recursos do CNPq, CAPES, FAPERJ, FINEP, RNP, FUNTEL e UOL.

partir de um único pacote de ataque recebido, sem armazenar estado na infra-estrutura de rede. Em linhas gerais, este sistema de rastreamento propõe que cada pacote armazene os roteadores por ele atravessados em uma estrutura de dados compacta, denominada filtro de Bloom [8]. Em posse do filtro de um pacote recebido, a vítima realiza a reconstrução da rota do pacote recebido. De forma a limitar a ação do atacante de prejudicar o rastreamento, também é proposta uma generalização do filtro de Bloom e o seu emprego para armazenar os roteadores atravessados. Entretanto, falsos negativos surgem com a generalização proposta. Um falso negativo na reconstrução da rota significa deixar de identificar alguns roteadores por onde o pacote passou. Neste artigo, o procedimento alternativo proposto lida justamente com a redução dos falsos negativos na reconstrução de rota. Com a adoção deste novo procedimento, é mostrado por simulação que os falsos negativos são reduzidos a 0%. Ainda por meio de simulações, também é mostrado que o procedimento aqui proposto identifica o atacante com acurácia.

O artigo está organizado da seguinte forma. Na Seção II, são apresentados alguns trabalhos relacionados ao rastreamento de pacotes IP. Na Seção III, o sistema de rastreamento é brevemente revisado de forma a esclarecer as contribuições deste trabalho. Na Seção IV, o procedimento alternativo de reconstrução de rota é explicado, incluindo resultados analíticos do mesmo. Resultados de simulação são apresentados na Seção V mostrando a eficácia do método proposto. Por fim, na Seção VI são relatadas as conclusões retiradas deste trabalho.

II. TRABALHOS RELACIONADOS

Savage *et al.* [9] introduzem um sistema baseado em auditoria, onde a informação necessária para o rastreamento é encontrada na vítima. Os roteadores a informam sobre sua presença na rota, sobrecarregando o campo de identificação de fragmentação do cabeçalho IP. Visando reduzir o processamento no roteador e o espaço necessário em cada pacote, técnicas de codificação e amostragem são empregadas. Posteriormente, a vítima consegue reconstruir a rota tomada por um fluxo de ataque depois de recebido um número suficiente de pacotes deste fluxo. 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.

Outro método baseado em auditoria foi proposto por Belvin *et al.* [10]. Ao rotear um pacote, cada roteador probabilisticamente envia para a vítima um pacote ICMP com informações sobre si mesmo e sobre seus roteadores adjacentes. Para um fluxo suficientemente longo, a vítima usa estes dados recebidos para reconstruir a rota de ataque. Entretanto, como as informações de auditoria são enviadas em pacotes separados, a autenticação das mensagens é necessária de forma a evitar mensagens forjadas pelo atacante. Logo, a adoção de uma infra-estrutura de chave pública se torna inevitável.

Sob a perspectiva de armazenamento na própria infra-estrutura de rede, a maneira mais simples de recolher rastros de auditoria é cada roteador registrar todos os pacotes que o atravessam, como o proposto por Stone [11]. Porém, recursos

excessivos são requisitados tanto para armazenagem quanto para a mineração dos dados. Além disso, a invasão de um roteador acarretaria ainda em problemas de privacidade, uma vez que ele contém informações sobre todos os pacotes roteados.

Uma alternativa para reduzir a armazenagem de um grande volume de informações é utilizar um filtro de Bloom [8]. Ultimamente, estes filtros têm sido amplamente usados em redes de computadores. Snoeren *et al.* [12] 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.

III. O SISTEMA DE RASTREAMENTO

Esta seção apresenta brevemente o sistema de rastreamento de pacotes IP proposto previamente [7]. Este sistema objetiva rastrear a origem de cada pacote individualmente. Ela se baseia na abordagem de inserção de informações nos pacotes para evitar o armazenamento na infra-estrutura de rede. Ao invés de usar uma marcação tal qual a proposta por Savage *et al.* [9], cada roteador insere no pacote uma “assinatura” que indica sua presença na rota. A técnica de filtro de Bloom é usada para que a quantidade de informação inserida seja reduzida e permaneça constante a fim de evitar a fragmentação dos pacotes. Além disso, uma generalização do filtro de Bloom (Apêndice I) foi proposta para evitar que o atacante possa forjar as “assinaturas” e prejudicar o rastreamento.

De forma a diminuir tanto o espaço necessário em cada pacote como o custo de processamento, a rota de ataque é armazenada em um filtro de Bloom embutido em cada pacote. Para isso, um campo fixo é reservado no cabeçalho do pacote para o filtro. O algoritmo de marcação para este caso é bem simples. Pouco antes de reencaminhar um pacote, todo roteador insere no filtro daquele pacote o endereço IP da sua interface de saída. Desta forma, ao receber um pacote de ataque, a vítima dispõe de um filtro cujos elementos são todos os roteadores componentes da rota de ataque.

Para reconstruir esta rota, o seguinte algoritmo é utilizado. Inicialmente, a vítima verifica a presença de todos os seus roteadores vizinhos 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, este roteador verifica qual dos seus roteadores vizinhos também é reconhecido como elemento do filtro, identificando assim o próximo componente da rota de ataque. Um processo recursivo é então realizado sucessivamente por cada roteador visando reconstruir o caminho do pacote até a sua verdadeira origem. A Fig. 1 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_4, R_5, R_2, R_1) . Ao receber o pacote de ataque, a vítima envia o filtro para R_1 que, por sua vez, verifica a presença de R_2 e R_3 no filtro (1). Como somente R_2 é reconhecido, o filtro é então repassado somente para R_2 que faz o mesmo procedimento com seu vizinho R_5 , que por sua vez também é reconhecido (2). O roteador R_5 verifica qual dos seus dois vizinhos R_3 e R_4 é reconhecido pelo filtro; somente R_4 é reconhecido (3). Finalmente, R_4 testa a presença de R_7 no filtro sem sucesso e a reconstrução termina (4).

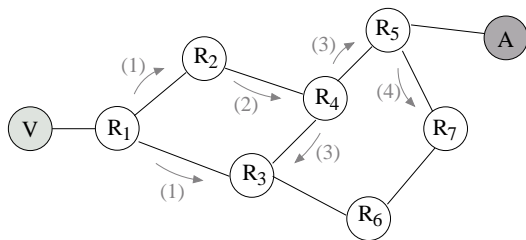


Fig. 1. Reconstrução de rota da vítima V em direção ao atacante A , passando por (R_1, R_2, R_5, R_4) .

Algumas vantagens surgem como resultado da adoção dessa abordagem. Primeiramente, a rota completa de cada pacote pode ser determinada individualmente. Tal comportamento é idealizado por todo sistema de rastreamento de pacotes uma vez que possibilita a identificação de qualquer fonte em um ataque distribuído. Além disso, nenhum estado necessita ser armazenado 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 a capacidade de se realizar o rastreamento após o término do ataque e sem ajuda de operadores de rede.

Por outro lado, a adoção do filtro de Bloom para representar a rota de ataque introduz uma certa probabilidade de falso positivo. Durante o algoritmo de reconstrução, um falso positivo implicaria a incorreta integração de um roteador à rota de ataque. Porém, se esta probabilidade for pequena, a ocorrência de falsos positivos não tem nenhum 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 todos os bits do filtro com 1. Ao saturar o filtro, todo roteador é integrado à rota de ataque durante a reconstrução, tornando impraticável a descoberta da verdadeira rota.

Para minimizar a possibilidade do atacante burlar o sistema e torná-lo menos dependente da condição inicial do vetor, uma generalização do filtro de Bloom também foi proposta junto com o sistema (Apêndice I). Um filtro generalizado estaria integrado em cada pacote, de forma a armazenar os roteadores atravessados. A idéia básica do filtro generalizado é utilizar tanto funções *hash* que preenchem bits como funções que zeram bits. Desta forma, é possível mostrar que a probabilidade de falso positivo é limitada e depende pouco da condição inicial. Por outro lado, falsos negativos que eram

inexistentes no filtro de Bloom convencional são introduzidos nesta proposta de filtro de Bloom generalizado.

IV. PROCEDIMENTO DE RECONSTRUÇÃO PROPOSTO

Se por um lado o uso de um filtro de Bloom generalizado 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 Fig. 1 há um falso negativo no roteador R_5 . Ou seja, o roteador R_5 não é reconhecido pelo filtro generalizado durante a verificação realizada por R_2 . Neste caso, os roteadores R_3 e R_4 não são checados, uma vez que R_5 não foi integrado à rota de ataque. Em consequência, o roteador R_4 também não é integrado à rota de ataque. Desta forma, somente um falso negativo pode ser suficiente para interromper o procedimento de reconstrução e evitar a determinação da verdadeira rota de ataque.

Para resolver este problema, é aqui proposto um procedimento alternativo de reconstrução que elimina os falsos negativos ao custo de uma maior probabilidade de falso positivo. Este novo procedimento de reconstrução é baseado no fato que, durante a travessia do pacote pela rede, roteadores subsequentes podem inverter os bits marcados por roteadores anteriores e causar falsos negativos durante a reconstrução. No exemplo anterior, durante o percurso do pacote de ataque por (R_4, R_5, R_2, R_1) , ou R_1 ou R_2 inverteu algum bit marcado previamente por R_5 . Em consequência, o filtro do pacote recebido não reconhece R_5 durante o procedimento de reconstrução de rota. De forma a evitar este problema, durante a reconstrução, cada roteador envia informações adicionais sobre os bits marcados por ele mesmo e pelos roteadores anteriores para os próximos roteadores. O próprio filtro generalizado também é repassado, como é feito no procedimento padrão. A Fig. 2 ilustra brevemente este conceito. Na figura, o roteador R_1 primeiramente reconhece os roteadores R_2 e R_3 no filtro recebido pela vítima e, portanto, R_1 lhes repassa o filtro de forma a continuar a reconstrução. Entretanto, R_1 também envia para os próximos roteadores um outro vetor de bits indicando quais bits foram marcados por ele. Tais bits estão marcados com X na figura. Por sua vez, os roteadores R_2 e R_3 , ao reconhecerem algum roteador vizinho como integrante da rota de ataque, repassam-lhe o próprio filtro e o vetor adicional de bits indicando as marcações feitas por eles e por R_1 . Desta forma, um roteador R_i sempre repassa para os próximos roteadores o próprio filtro junto com o vetor de bits X, indicando os bits marcados por R_i e pelos roteadores anteriores.

O processo de verificação dos vizinhos se altera um pouco em virtude da informação adicional recebida por cada roteador. Agora, quando um vizinho não é reconhecido pelo filtro, o roteador verifica se os bits marcados por este vizinho foram reescritos por algum roteador seguinte durante a travessia do pacote de ataque. Para isso, ele verifica o vetor de bits X recebido pelo roteador anterior na reconstrução. Caso todos os bits do vizinho que estão invertidos no filtro estiverem

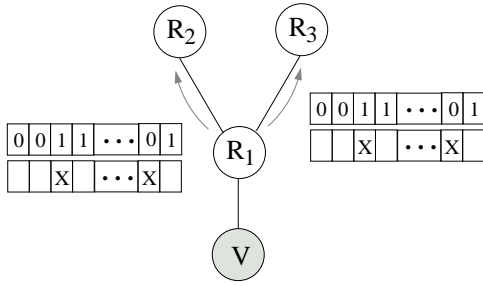


Fig. 2. Procedimento de reconstrução de rota proposto.

marcados como reescritos no vetor de bits X, então o vizinho é reconhecido como um elemento do filtro e, portanto, integrado à rota de ataque. Por outro lado, se algum dos bits do vizinho que estão invertidos no filtro não estiver no vetor de bits X, é porque nenhum roteador inverteu aquele bit e, portanto, aquele vizinho realmente não foi inserido no filtro durante a travessia do pacote de ataque. No exemplo dado anteriormente, ao não reconhecer o roteador R_5 no filtro, o roteador R_2 verificaria se as marcações invertidas de R_5 estão no vetor de bits X. Como certamente elas estão, o roteador R_5 é integrado à rota de ataque e continua o procedimento de reconstrução.

Desta forma, falsos positivos não podem mais ocorrer, dado que os bits reescritos de cada roteador agora podem ser checados a cada salto. Entretanto, 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 X aumenta para cada roteador integrado à rota de ataque. Assim, à medida que a fração de bits X aumenta, roteadores que não eram antes reconhecidos porque alguns de seus bits estavam invertidos, agora podem ser reconhecidos com maior probabilidade.

A probabilidade de falso positivo do procedimento de reconstrução proposto pode ser calculada de uma maneira simples. Primeiramente, é calculada a fração $q_X(n-i)$ de bits X presentes durante a verificação feita por um roteador $(n-i)$, onde n é o tamanho da rota de ataque e $0 \leq i \leq (n-1)$. Por definição, é determinado que o roteador n é aquele mais próximo da vítima e o roteador 1 é aquele mais próximo do atacante. Ou seja, para o roteador mais próximo da vítima $i = 0$ e para o roteador mais próximo do atacante $i = n-1$. Seja q_0 e q_1 a fração de bits colocados em 0 e em 1, na média, a cada inserção. A fração de bits X para um roteador $(n-i)$ pode ser expressa por

$$q_X(n-i) = 1 - (1 - q_0 - q_1)^i, \quad (1)$$

para $0 \leq i \leq (n-1)$. A fração $(1 - q_0 - q_1)$ representa a fração de bits que permanece intocada a cada inserção. A fração $(1 - q_0 - q_1)^i$ representa justamente a fração de bits que não foi marcada por nenhum dos i roteadores anteriores na reconstrução. Naturalmente, a expressão $1 - (1 - q_0 - q_1)^i$ representa a fração de bits alterados. Seja $p_0(n-i)$ e $p_1(n-i)$ a fração de bits em 0 e em 1, na média, do filtro generalizado depois de inseridos $(n-i)$ roteadores durante a travessia do pacote de ataque. A fração $p_{0X}(n-i)$ dos bits que podem ser interpretados como 0 no filtro do $(n-i)$ -ésimo roteador

durante a reconstrução é

$$p_{0X}(n-i) = p_0(n-i) [1 - q_X(i)] + q_X(i). \quad (2)$$

Equivalentemente, a fração $p_{1X}(n-i)$ de bits que podem ser interpretados como 1 no filtro do $(n-i)$ -ésimo roteador durante a reconstrução é

$$p_{1X}(n-i) = p_1(n-i) [1 - q_X(i)] + q_X(i). \quad (3)$$

A partir de (2) e (3), a probabilidade $f_p(n-i)$ de um falso positivo para o $(n-i)$ -ésimo roteador no procedimento alternativo de reconstrução de rota pode ser calculada e expressa por

$$f_p(n-i) = p_{0X}(n-i)^{b_0} p_{1X}(n-i)^{b_1}, \quad (4)$$

onde b_0 e b_1 são respectivamente o número de bits colocados em 0 e em 1, por inserção. Essa expressão representa que um falso positivo ocorre quando são achados b_0 bits em 0 e b_1 bits em 1, ou no próprio filtro ou no vetor de bits X.

V. RESULTADOS DE SIMULAÇÃO

De forma a analisar o comportamento do procedimento alternativo de reconstrução proposto, foi desenvolvido um simulador em C++. Inicialmente, para criar a topologia usada na simulação, foi usado o gerador de topologia *nem*, baseado em amostragem do mapa da Internet [13]. O gerador *nem* extrai aleatoriamente um sub-grafo 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.

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 definido o número de saltos da rota de ataque. Com o atacante como origem, o algoritmo de Dijkstra é então executado até encontrar alguma rota com o tamanho escolhido. A escolha desse algoritmo, ao invés de uma escolha aleatória da rota, reflete melhor as rotas usadas nas redes de computadores. Posteriormente, é simulado o envio de um pacote de ataque através da marcação do filtro do pacote de acordo com os roteadores escolhidos para compor a rota de ataque. 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 proposto neste artigo, onde informações adicionais também são repassadas a cada salto. Para cada ponto medido, foram realizadas 100 simulações e calculado o intervalo de confiança usando uma confiabilidade de 95%.

A Fig. 3 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). De forma a facilitar o entendimento das simulações, usou-se o mesmo número de funções que colocam bits em 0 e em 1 no filtro, ou

seja, $k = k_0 = k_1$. 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, foram rastreados aproximadamente sete roteadores. Este comportamento ocorre devido a 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.

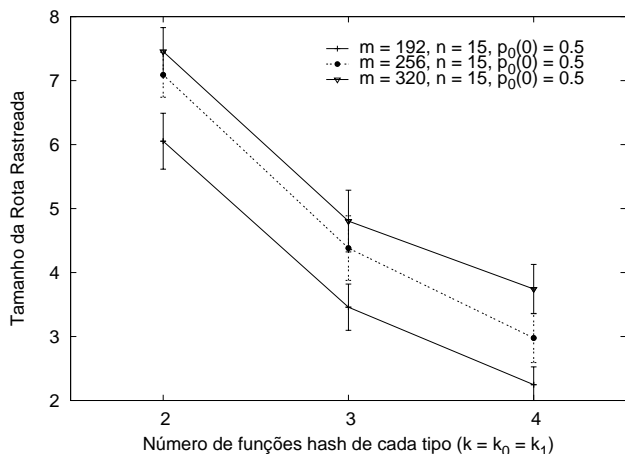


Fig. 3. Tamanho da rota reconstruída para o procedimento de reconstrução padrão em função do número de funções *hash* de cada tipo (k), para diferentes tamanhos de filtro (m), um tamanho da rota de ataque (n) de 15 roteadores e supondo o pior caso para a condição inicial do filtro ($p_0(0)$).

O procedimento de reconstrução de rota proposto neste artigo 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. Por isso, como forma de avaliar o desempenho do procedimento proposto, 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.

A Fig. 4 mostra o número de atacantes encontrados pelo procedimento proposto em função do número de funções *hash* usadas (k). Pode-se perceber que, para os mesmos tamanhos de filtro usados no procedimento padrão, o número de possíveis atacantes é reduzido a menos de quatro, podendo inclusive se aproximar do caso ideal para filtros de maior tamanho. 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 atacantes rastreados 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 X 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 atacantes rastreados.

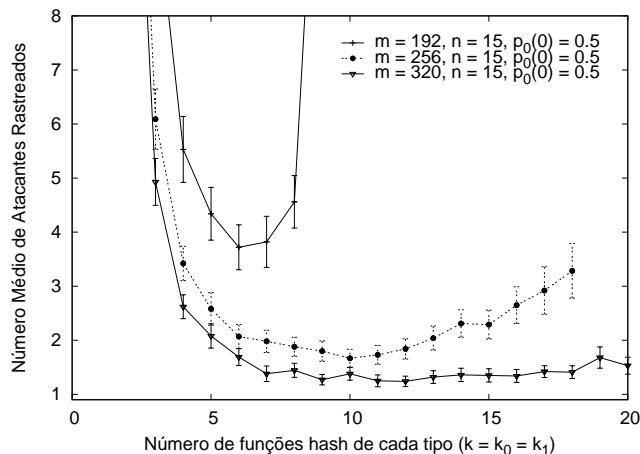


Fig. 4. Número médio de atacantes rastreados pelo procedimento de reconstrução proposto em função do número de funções *hash* de cada tipo (k), para diferentes tamanhos de filtro (m), um tamanho da rota de ataque (n) de 15 roteadores e supondo o pior caso para a condição inicial do filtro ($p_0(0)$).

Pode ser observado na Fig. 5 o número de atacantes rastreados pelo procedimento proposto 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 proposto, a probabilidade de falso positivo aumenta à medida que os roteadores são testados mais longe da vítima. Em conseqüência, mais roteadores são reconhecidos como componentes da rota de ataque e mais atacantes são encontrados. Porém, mesmo assim, ainda é possível encontrar um número de atacantes muito próximo do ideal dependendo do tamanho do filtro utilizado.

VI. CONCLUSÕES

Neste artigo, é introduzido um procedimento alternativo de reconstrução de rota para o rastreamento de pacotes IP. 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.

O procedimento de reconstrução de rota proposto foi comparado com o procedimento padrão por meio de simulação. É

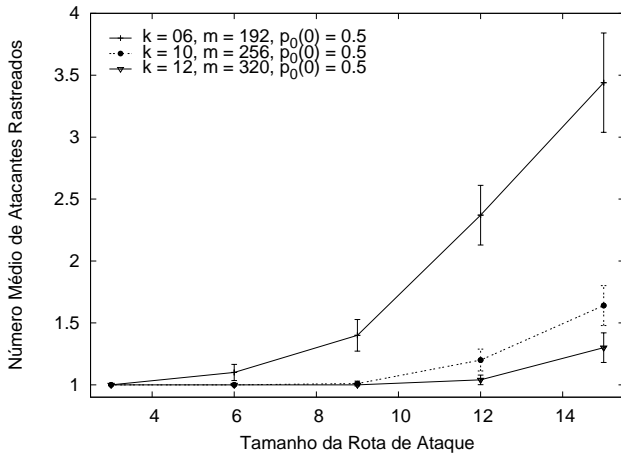


Fig. 5. Número médio de atacantes rastreados pelo procedimento de reconstrução proposto em função do tamanho da rota de ataque, para diferentes tamanhos de filtro (m), cada qual com o número de funções *hash* ótimo, de acordo com a Fig. 4, e supondo o pior caso para a condição inicial do filtro ($p_0(0)$).

mostrado por meio de simulação 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 que o procedimento proposto 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. No procedimento proposto, é mostrado ainda que o atacante é sempre identificado e que esta identificação é muito próxima da ideal, onde somente um atacante é encontrado.

REFERÊNCIAS

- [1] CERT, *CERT Advisory CA-2003-20 W32/Blaster worm*, agosto de 2003. <http://www.cert.org/advisories/CA-2003-20.html>.
- [2] CERT, *CERT Advisory CA-2000-01 Denial-of-Service Developments*, janeiro de 2000. <http://www.cert.org/advisories/CA-2000-01.html>.
- [3] CERT, *CERT Advisory CA-1996-26 Denial-of-Service Attack via ping*, dezembro de 1996. <http://www.cert.org/advisories/CA-1996-26.html>.
- [4] CERT, *CERT Advisory CA-1997-28 IP Denial-of-Service Attacks*, dezembro de 1997. <http://www.cert.org/advisories/CA-1997-28.html>.
- [5] L. Garber, "Denial-of-Service Attacks Rip the Internet", *IEEE Computer*, vol. 4, no. 33, pp. 12–17, abril de 2000.
- [6] Microsoft Corporation, *Stop OA in Tcpip.sys When Receiving Out Of Band (OOB) Data*, outubro de 2002.
- [7] R. P. Laufer, P. B. Velloso e O. C. M. B. Duarte, "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, Brasil, maio de 2005.
- [8] B. H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors", *Comm. of the ACM*, vol. 7, no. 13, pp. 442–426, julho de 1970.
- [9] S. Savage, D. Wetherall, A. Karlin e T. Anderson, "Network Support for IP Traceback", *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, pp. 226–237, junho de 2001.
- [10] S. M. Bellovin, M. D. Leech e T. Taylor, "ICMP Traceback Messages", *Internet Draft: draft-ietf-itrace-04.txt*, agosto de 2003.
- [11] R. Stone, "CenterTrack: An IP Overlay Network for Tracking DoS Floods", em *9th USENIX Security Symposium*, Denver, Colorado, EUA, pp. 199–212, agosto de 2000.
- [12] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent e W. T. Strayer, "Single-Packet IP Traceback", *IEEE/ACM Transactions on Networking*, vol. 10, no. 6, pp. 721–734, dezembro de 2002.
- [13] D. Magoni e J.-J. Pansiot, "Internet Topology Modeler Based on Map Sampling", em *IEEE International Symposium on Computer Communications*, julho de 2002.

APÊNDICE I
O FILTRO DE BLOOM GENERALIZADO

Da mesma forma que o filtro convencional, o filtro de Bloom generalizado [7] é uma estrutura de dados usada para representar um conjunto $S = \{s_1, s_2, \dots, s_n\}$ de n elementos de forma compacta. Ele é constituído por um vetor de m bits e por $k_0 + k_1$ funções *hash* independentes $g_1, g_2, \dots, g_{k_0}, h_1, h_2, \dots, h_{k_1}$ cujas saídas variam uniformemente no espaço discreto $\{0, 1, \dots, m - 1\}$. O vetor de bits é calculado de maneira semelhante ao do caso convencional. Entretanto, não há mais a restrição de que seus bits são inicialmente zerados. No filtro generalizado, os bits do vetor podem assumir qualquer valor inicial. Para cada elemento $s_i \in S$, os bits do vetor correspondentes às posições $g_1(s_i), g_2(s_i), \dots, g_{k_0}(s_i)$ são zerados e os bits correspondentes às posições $h_1(s_i), h_2(s_i), \dots, h_{k_1}(s_i)$ 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 Fig. 6 ilustra de maneira sucinta a inserção de um elemento no 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 $g_1(x), g_2(x), \dots, g_{k_0}(x)$ estão zerados e se os bits $h_1(x), h_2(x), \dots, h_{k_1}(x)$ estão preenchidos com 1. Se pelo menos um bit está invertido, então $x \notin S$ com alta probabilidade. 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 $g_1(x), g_2(x), \dots, g_{k_0}(x)$ é preenchido ou quando pelo menos um dos bits $h_1(x), h_2(x), \dots, h_{k_1}(x)$ é zerado por algum outro elemento inserido posteriormente. Por outro lado, se nenhum bit está invertido, então $x \in S$ também com alta probabilidade. A incerteza se deve à possibilidade do filtro reconhecer como afiliado um elemento externo $x \notin S$, criando um falso positivo. Um falso positivo ocorre quando os bits $g_1(x), g_2(x), \dots, g_{k_0}(x)$ estão zerados e os bits $h_1(x), h_2(x), \dots, h_{k_1}(x)$ estão preenchidos com 1 em virtude de um subconjunto dos elementos de S ou da própria condição inicial do vetor.

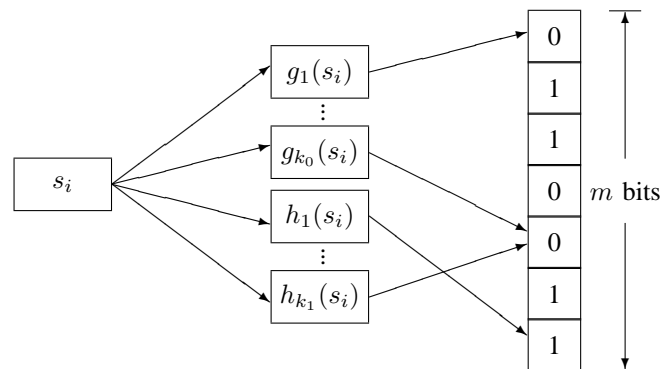


Fig. 6. Inserção de um elemento em um filtro de Bloom generalizado.