Uma Proposta de Marcação de Pacotes para Rastreamento Robusto a Ataques*

Marcelo D. D. Moreira[1], Rafael P. Laufer[2], Pedro Velloso[3] e Otto Carlos M. B. Duarte[1]

[1] Universidade Federal do Rio de Janeiro (UFRJ)

[2]University of California, Los Angeles (UCLA)

[3]Université Pierre et Marie Curie - Paris VI (UPMC)

Abstract. *This paper proposes a new packet marking structure robust to attacks to distributed applications such as IP traceback. The structure is based on the Bloom Filter, which allows space-efficient representation of a set. The goal of the so-called Concatenated Bloom Filter (CBF) is to provide robustness to attacker interference, allowing secure transmission of information in an insecure medium. The key idea is to concatenate a set of subfilters, each of them admitting the insertion of only one element. Analytical results show the efficacy of the CBF and that the attacker success probability decreases exponentially with the increasing of subfilter size. Any other proposal with such robustness and without the limitation of legitimate information loss was not found in the literature.*

Resumo. *Este artigo propõe uma nova estrutura de marcação de pacotes robusta a ataques a aplicações distribuídas como o rastreamento de pacotes IP. A estrutura de dados baseia-se no Filtro de Bloom, que permite representar de forma compacta um conjunto de elementos. O objetivo da proposta, denominada Filtro de Bloom Concatenado (FBC), é prover robustez à interferência do atacante e, com isto, garantir maior segurança na transferência da informação em um meio inseguro. A idéia chave da proposta é concatenar diversos subfiltros, cada qual admitindo somente um elemento. Resultados analíticos mostram que o filtro proposto é bem eficaz e que a probabilidade de sucesso do ataque proposto decresce exponencialmente com o tamanho de cada subfiltro. Não foi encontrada na literatura nenhuma proposta que provesse tal robustez sem a limitação de perda de informação legítima.*

1. Introdução

O Filtro de Bloom é uma estrutura de dados capaz de representar um conjunto de forma compacta [Bloom 1970]. Ele permite o armazenamento de apenas um pequeno vetor de bits, ao invés de todos os elementos do conjunto a ser representado. O custo dessa compacidade é a possibilidade de um elemento externo ao conjunto ser reconhecido como um elemento legítimo do conjunto representado. Tal evento é conhecido como um falso positivo.

Filtros de Bloom foram inicialmente utilizados nas áreas de verificação ortográfica e banco de dados. Recentemente, verificou-se que eles podem garantir uma troca eficiente de informações em aplicações distribuídas. Desde então surgiram novas propostas de

*Este trabalho foi financiado com recursos do CNPq, CAPES, RNP/FUNTTEL, FAPERJ e UOL

aplicações na área de redes de computadores, como colaboração em redes de sobreposição (*overlay*) e redes par-a-par (*peer-to-peer* - P2P), localização de recursos, protocolos de roteamento e infra-estruturas de medição [Broder e Mitzenmacher 2003]. Por exemplo, nas aplicações de redes P2P e localização de recursos, cada nó envia periodicamente para os seus vizinhos um Filtro de Bloom representando os objetos que podem ser alcançados através dele. Dessa maneira, todos os nós possuem conhecimento dos objetos que podem ser alcançados e, ao invés de usar, por exemplo, um identificador de 64 bits para cada objetos, são necessários apenas 8 ou 16 bits por objeto. Seguindo a mesma idéia de enviar o Filtro de Bloom no lugar de uma lista explícita de elementos, na aplicação de distribuição de *cache Web*, servidores *proxy* trocam Filtros de Bloom a fim de compartilhar os seus *caches*. No caso de um servidor não possuir uma página requisitada, ele primeiro verifica os filtros dos outros servidores para determinar se eles possuem a página em *cache* ou não. No caso afirmativo, a requisição da página é repassada àquele servidor. A pequena probabilidade de falso positivo adicional introduzida pelo uso do Filtro de Bloom é amplamente justificada pela significativa redução do tráfego de rede. Uma importante e desafiadora aplicação de Filtros de Bloom é o rastreamento de pacotes IP [Savage et al. 2001]. O objetivo é identificar a verdadeira origem de ataques anônimos de negação de serviço (*Denial of Service - DoS*). Ao invés de armazenar os endereços de todos os roteadores atravessados pelo pacote de ataque, propõe-se embutir no cabeçalho do pacote um Filtro de Bloom, que irá armazenar a rota de ataque de forma compacta [Laufer et al. 2005a]. Durante a travessia do pacote, cada roteador insere o seu endereço no filtro. O rastreamento da origem do ataque é feito de forma recursiva a partir da vítima. Esta realiza testes de pertinência sucessivos com seus roteadores vizinhos. Os roteadores cujo teste de pertinência for positivo são aqueles pelo qual o pacote passou durante o ataque. Dessa forma, toda a rota de ataque pode ser reconstruída.

Apesar de sua comprovada eficiência, o uso de Filtros de Bloom nessas aplicações distribuídas é limitado por aspectos de segurança. A questão principal gira em torno da condição inicial do filtro. Nesse artigo é apresentado um ataque no qual a condição inicial do filtro é cuidadosamente ajustada no intuito de comprometer aplicações distribuídas nas quais o filtro é enviado pela rede. É mostrado que o atacante pode maximizar a probabilidade de falso positivo dos elementos desejados ou até mesmo saturar o filtro. Um filtro saturado torna impossível distinguir elementos legítimos de falsos positivos em um teste de pertinência, apresentando taxas de falso positivo de até 100%. Os efeitos de ataques desse tipo nas aplicações mencionadas anteriormente podem ser catastróficos. Nas aplicações de compartilhamento de *cache Web* e de redes P2P, os demais nós acreditarão que o atacante possui todas as páginas *Web* ou objetos procurados. Os pedidos serão encaminhados ao atacante, que poderá substituir os objetos procurados por outros ao seu bel prazer. No caso do rastreamento de pacotes IP, o atacante pode comprometer a acurácia do sistema de rastreamento. Ao maximizar a probabilidade de falso positivo de roteadores que não pertencem à rota de ataque, o atacante faz com que sejam rastreadas outras rotas além da verdadeira. Mostra-se que, num caso extremo, o atacante pode saturar o filtro de bloom, fazendo com que todas as rotas da Internet sejam rastreadas. Conseqüentemente, a reconstrução de rota será completamente ineficaz.

Apesar das falhas de segurança, pouco se tem estudado sobre mecanismos eficientes de segurança para Filtros de Bloom. Nesse trabalho, é proposto um Filtro de Bloom robusto à interferência da condição inicial do filtro. A estrutura de dados pro-

posta é denominada Filtro de Bloom Concatenado (FBC). A idéia chave é sobrescrever as informações iniciais, a fim de reduzir a interferência da condição inicial do filtro. Uma proposta existente na literatura [Laufer et al. 2005a] que visa tal objetivo possui a limitação de perda de informação, devido à possibilidade de informações inseridas pelos primeiros elementos serem sobrescritas por inserções posteriores. Para contornar esse problema, propõe-se dividir o filtro original em diversos subfiltros com somente um elemento. O Filtro de Bloom Concatenado (FBC) é o resultado da concatenação desses subfiltros. Comparado às outras propostas encontradas na literatura, o FBC possui como principais vantagens a ausência de falsos negativos, a maior capacidade de armazenamento e a maior robustez à interferência da condição inicial.

O restante do artigo está organizado da seguinte forma. O Filtro de Bloom convencional é descrito na Seção 2. A Seção 3 apresenta os principais trabalhos relacionados ao tema encontrados na literatura. Na Seção 4 é proposto um ataque capaz de comprometer aplicações distribuídas nas quais o Filtro de Bloom é enviado pela rede. A seguir, a Seção 5 descreve em detalhes a proposta desse trabalho. Na Seção 6 são apresentados os resultados analíticos obtidos. Por fim, a Seção 7 conclui este trabalho e apresenta as suas futuras direções.

2. Filtro de Bloom

O Filtro de Bloom é uma estrutura de dados capaz de representar eficientemente um conjunto $S = \{s_1, s_2, \ldots, s_n\}$ de n elementos. Ele é constituído por um vetor de m bits, inicialmente todos zerados. São admitidos dois algoritmos: inserção de elemento e teste de pertinência. O Filtro de Bloom associado a S é o vetor resultante da aplicação do algoritmo de inserção aos n elementos de S. A aplicação do teste de pertinência a um elemento x informa se x pertence ou não a S, com uma certa probabilidade de falso positivo associada.

Para inserir um elemento no filtro, são utilizadas k funções *hash* independentes $h_1, h_2, \ldots, h_k$, cujas saídas variam uniformemente no espaço discreto $\{0, 1, \ldots, m-1\}$. O Algoritmo 1 recebe como argumentos de entrada um elemento $s_i \in S$ ($1 \leq i \leq n$) e o Filtro de Bloom B. O passo seguinte é o preenchimento com 1 dos bits do vetor B correspondentes às posições $h_1(s_i), h_2(s_i), \ldots, h_k(s_i)$. Ao final desse processo, o elemento s_i está inserido no Filtro de Bloom.

Algoritmo 1: Inserção de Elemento no Filtro de Bloom

Entrada: $s_i \in S$ e o vetor B de m bits.
Resultado: Inserção do elemenho s_i no filtro B
início
  para $j = 1$ **até** k **faça**
    $B[h_j(s_i)] \leftarrow 1$
fim

Para determinar se um elemento x pertence ou não ao conjunto S, é necessário aplicar o algoritmo de teste de pertinência. O Algoritmo 2 recebe como argumentos de entrada o elemento x e o Filtro de Bloom B. Verificar a pertinência de x significa checar se x já foi inserido no Filtro de Bloom ou não. Assim, o objetivo é verificar se os bits do vetor B 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 certamente $x \notin S$, pois se x já tivesse sido

inserido no Filtro de Bloom anteriormente, então todos os bits indicados pelas funções *hash* deveriam estar preenchidos com 1. Nesse caso, o algoritmo retorna 0, indicando que o teste de pertinência foi negativo. Por outro lado, se todos os bits estão preenchidos com 1, então é assumido que $x \in S$ e o algoritmo retorna 1, indicando que o teste de pertinência foi positivo.

Algoritmo 2: Teste de Pertinência no Filtro de Bloom

Entrada: Elemento x e o vetor B de m bits.
Saída: 1, se $x \in S$; 0, caso contrário.
início
  para $j = 1$ **até** k **faça**
    se $B[h_j(x)] = 0$ **então retorna** *0*
  retorna *1*
fim

É importante notar que um elemento externo $x \notin S$ pode ser reconhecido como um elemento legítimo do conjunto, evento denominado falso positivo. Tal evento ocorre quando todos os bits $h_1(x), h_2(x), \ldots, h_k(x)$ foram preenchidos com 1 por elementos de S previamente inseridos no filtro.

A probabilidade de se encontrar um falso positivo para um elemento $x \notin S$ é calculada da seguinte maneira. Como 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

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

Como o mesmo cálculo se aplica a todos os bits do vetor B, na média, uma fração p dos bits permanece zerada após as inserções [Mitzenmacher 2002]. Dessa forma, a fração média de bits preenchidos com 1 depois de n inserções é $1 - p$. A probabilidade de falso positivo f_p é a probabilidade de se encontrar um bit em 1 para todas as k posições indicadas, ou seja,

$$f_p = \left(1 - p\right)^{k}. \tag{2}$$

3. Trabalhos Relacionados

A literatura acerca dos Filtros de Bloom é bastante vasta. Encontram-se também diversas variações do Filtro de Bloom convencional. Dentre elas, destacam-se o Filtro de Bloom Contador (*Counting Bloom Filter*), que admite não somente a inserção, mas também a retirada de elementos [Fan et al. 2000]; o Filtro de Bloom Paralelo, usado para inspeção de cadeias de bytes dentro de pacotes [Dharmapurikar et al. 2004]; o Filtro de Bloom Comprimido (*Compressed Bloom Filter*), que permite a transmissão eficiente de um filtro entre duas estações [Mitzenmacher 2002]; o Filtro de Bloom Hierárquico (*Hierarchical Bloom Filter*) que possibilita a busca de subcadeias de caracteres [Shanmugasundaram et al. 2004]; o Filtro de Bloom Atenuado (*Attenuated Bloom Filter*), que melhora a localização e o roteamento de documentos em redes *peer-to-peer* (P2P); e o Filtro de Bloom Espectral (*Spectral Bloom Filter*), que retorna uma estimativa do número de ocorrências de um dado elemento no filtro [Cohen e Matias 2003]. Outra variação, proposta por Estan e Varghese, é um Filtro de Bloom com suporte a multiconjuntos usado para identificar fluxos intensos de dados

em um roteador [Estan e Varghese 2003]. Seguindo a abordagem de representação de multiconjuntos, foi proposto também o *Space-Code Bloom Filter* [Kumar et al. 2004].

Apesar de serem eficientes para as aplicações a que se destinam, as propostas aqui mencionadas não levam em consideração aspectos de segurança. Dessa maneira, essas aplicações podem estar sujeitas a ataques que explorem as vulnerabilidades dos respectivos Filtros de Bloom. Uma proposta pioneira de um filtro seguro é o chamado Filtro de Bloom Generalizado (FBG), que visa deixar o Filtro de Bloom menos dependente da sua condição inicial [Laufer et al. 2005a]. Como o FBG tem o mesmo objetivo do Filtro de Bloom Concatenado proposto nesse trabalho, ele será brevemente descrito a seguir.

Assim como o Filtro de Bloom original, o Filtro de Bloom Generalizado (FBG) é uma estrutura de dados capaz de representar um dado conjunto de forma compacta. A grande novidade é a introdução de funções *hash* que zeram bits, além daquelas que preenchem bits com 1. Os autores mostram que isso permite ao filtro ser menos dependente da sua condição inicial, já que agora a probabilidade de falso positivo é limitada, independentemente do estado inicial do filtro. O custo dessa vantagem é a introdução de falsos negativos, isto é, agora um elemento pertencente ao conjunto representado pelo filtro pode não ser reconhecido como tal. O FBG também admite os algoritmos de inserção de elemento e de teste de pertinência. Entretanto, não há mais a restrição de que os bits do filtro estejam inicialmente zerados. No FBG, os bits do filtro podem assumir qualquer valor inicial. O algoritmo de inserção de elemento é semelhante ao do caso convencional. São utilizadas $k_0 + k_1$ 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 $\{0, 1, \ldots, m-1\}$. Os bits correspondentes às posições $h_1(s_i), h_2(s_i), \ldots, h_{k_1}(s_i)$ são preenchidos com 1 e os bits correspondentes às posições $g_1(s_i), g_2(s_i), \ldots, g_{k_0}(s_i)$ são zerados.

Para determinar se um elemento x pertence ou não ao conjunto S, é necessário aplicar o algoritmo de teste de pertinência. O objetivo é verificar se os bits correspondentes às posições $g_1(x), g_2(x), \ldots, g_{k_0}(x)$ 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 é assumido que $x \notin S$ e o teste de pertinência é negativo. Diferentemente do Filtro de Bloom convencional, agora há uma possibilidade de um elemento $x \in S$ não ser reconhecido pelo filtro, evento denominado falso negativo. Por outro lado, se nenhum bit está invertido, então é assumido que $x \in S$ e o teste de pertinência é positivo.

Uma análise detalhada das probabilidades de falso positivo e falso negativo do FBG pode ser obtida em [Laufer et al. 2005b]. Um resultado interessante é que, quando $m \gg k_0$ e $m \gg k_1$, a probabilidade de falso positivo do FBG é limitada e seu máximo vale

$$\left(\frac{k_0}{k_0 + k_1}\right)^{k_0} \left(\frac{k_1}{k_0 + k_1}\right)^{k_1}. \tag{3}$$

No entanto, os cálculos realizados consideram uma escolha aleatória da condição inicial do filtro, na qual todos os bits têm a mesma probabilidade p_0 de estar em 0. Portanto, escolhendo cuidadosamente os bits iniciais do filtro, um atacante fazer com que a probabilidade de ser positivo o teste de pertinência de determinados elementos que não pertencem ao filtro seja maior do o limite superior aqui mostrado. Esse ataque e sua respectiva probabilidade de sucesso são apresentados em detalhe mais adiante, nas Seções 4 e 6.2, respectivamente. É mostrado ainda que a probabilidade de sucesso do atacante é bem

menor quando o Filtro de Bloom Concatenado é utilizado.

Outra consideração que deve ser feita é que os falsos negativos introduzidos com o FBG são devidos, essencialmente, à possibilidade de inversão de um bit de um elemento durante a inserção de elementos posteriores no filtro. Apesar de serem responsáveis pela independência da condição inicial do filtro, os demais efeitos dos falsos negativos são bastante indesejáveis. Se o Filtro de Bloom for encarado como uma espécie de *buffer* de armazenamento dos elementos inseridos, um efeito bastante maléfico dos falsos negativos é a perda de informação, interpretada como um fenômeno de "esquecimento" dos elementos antigos. Em outras palavras, a introdução de falsos negativos causa perda de informação, ou seja, a limitação de memória desse *buffer*. O Filtro de Bloom original possui memória ilimitada, isto é, não há perda de informação e, portanto, o filtro não "esquece" nenhum dos elementos inseridos por mais que se insiram novos elementos. Já o FBG não pode aceitar a inserção de tantos elementos quanto se queira, pois invariavelmente existirá um elemento que irá inverter um dos bits de um elemento inserido anteriormente, ocasionando o "esquecimento" do elemento mais antigo. Por exemplo, esse fenômeno de "esquecimento" dos elementos antigos foi observado nas simulações do sistema de rastreamento proposto por Laufer *et al.*, nas quais somente cerca de 7 roteadores foram rastreados, para um filtro de 256 bits e uma rota de 15 saltos [Laufer et al. 2006]. Isso significa que a probabilidade de inversão de bits foi grande o suficiente para que os 8 primeiros roteadores fossem "esquecidos" pelo filtro. Conclui-se que a inversão de bits tem como conseqüência direta a perda das informações mais antigas, a fim de que novas informações possam ser inseridas. Por outro lado, quanto mais bits forem invertidos, maior será a probabilidade das informações iniciais do filtro serem apagadas. Conseqüentemente, maior será a robustez à interferência da condição inicial do filtro. Assim, uma medida da robustez do filtro é a probabilidade de falso negativo, após n inserções subseqüentes, de um elemento que era reconhecido pelo filtro devido à condição inicial. A Figura 1 ilustra, para um FBG de 128 bits, o compromisso entre a robustez e a perda de informação legítima, medida pela probabilidade de "esquecimento" do primeiro elemento legítimo inserido.

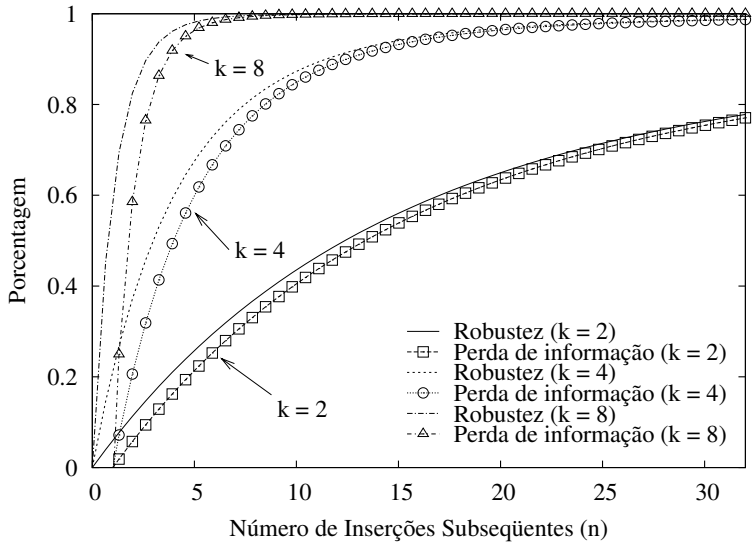


Figura 1. Compromisso entre robustez e perda de informação para um FBG.

Da figura, pode-se observar que a robustez do FBG cresce com o aumento do número

de elementos inseridos no filtro. Nota-se que, mesmo com poucos elementos inseridos no filtro, a robustez já é bastante elevada. Porém, pode-se notar que a probabilidade de perda de informação acompanha a curva da robustez. Por exemplo, na curva onde $k = k_0 = k_1 = 8$, com apenas 5 inserções já se tem uma robustez de $\approx$ 98,2%, mas isso significa uma probabilidade de "esquecimento" do primeiro elemento legítimo de $\approx$ 96,5%, ou seja, a chance de que haja perda de informação é muito grande. Logo, não se pode usar muitas funções *hash*, pois isso causa um indesejável fenômeno de "esquecimento" dos elementos mais antigos. Assim, deve-se escolher uma outra curva, na qual $k = k_0 = k_1$ seja menor. Por exemplo, na curva onde $k = 2$, a perda de informação cai para $\approx$ 22%, porém agora a robustez agora é de apenas 27%. Nesse caso, como $k = k_0 = k_1 = 2$, da Equação 3, a probabilidade máxima de falso positivo do FBG é 6,25%. Será mostrado que o FBC tem uma probabilidade de falso positivo equivalente quando são usados 4 bits por elemento, o que corresponde a uma robustez de $93,75\%$, que é bem maior do que os 27% do FBG. Nota-se que no FBG foram usados $128/5 = 25,6$ bits por elemento, enquanto que no FBC foram necessários apenas 4 bits por elemento, para uma mesma probabilidade de falso positivo. Portanto, conclui-se que o filtro concatenado alcança maior robustez usando menos bits. Além disso, no filtro concatenado são utilizados subfiltros que admitem somente um elemento cada. Dessa maneira, inserções posteriores nunca poderão inverter bits dos elementos mais antigos e, portanto, a probabilidade de falso negativo é sempre nula. Logo, não mais ocorre o efeito de "esquecimento" dos elementos mais antigos. Resumindo, para uma mesma probabilidade de falso positivo, a proposta desse trabalho supera o FBG em robustez e compacidade. Essas e outras vantagens do FBC ficarão mais claras na Seção 5.

4. Ataque do Ajuste da Condição Inicial

O ataque descrito nessa seção é bem simples. Ele se inspira na aplicação do rastreamento de pacotes. Nessa aplicação, cada roteador insere seu endereço no filtro de bloom contido no cabeçalho do pacote IP. Desta forma, ao receber um pacote de ataque, a vítima dispõe de um filtro cujos elementos são todos os roteadores componentes da rota atravessada. Com isso, a vítima pode reconstruir a rota de ataque. Inicialmente, a vítima verifica a pertinência dos endereços de seus roteadores vizinhos no filtro do pacote recebido. Aquele roteador cujo teste de pertinência for positivo é assumido como o roteador pelo qual o pacote chegou, sendo 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 visando reconstruir o caminho do pacote até a sua verdadeira origem. É importante notar que é o atacante quem inicializa o filtro de cada pacote enviado e esse filtro é depois modificado pelos roteadores atravessados. Assim, como o atacante detém o controle sobre a condição inicial do filtro, se a marcação dos pacotes não for robusta à interferência da condição inicial, então o rastreamento poderá ser burlado.

Sejam U o conjunto de todos os roteadores da Internet, $S = \{s_1, s_2, \ldots, s_n\}$ o conjunto dos roteadores integrantes da rota de ataque e $A \subset (U \setminus S)$ um conjunto de roteadores que não pertencem à rota de ataque. O objetivo do ataque proposto é maximizar a probabilidade de falso positivo dos elementos de A, somente ajustando a condição inicial do filtro. Uma maneira simples de implementar tal ataque é simplesmente inserir um conjunto $B \subset U$ de roteadores no filtro antes de enviá-lo aos outros nós da rede. Caso a marcação dos pacotes não seja robusta à interferência da condição inicial e, portanto, não

haja uma maneira de retirar esses elementos do filtro, invariavelmente eles terão teste de pertinência positivo. E ainda, se $B \subset A$, então esses elementos serão falsos positivos, pois certamente eles não pertencem a S. Um caso particular interessante desse ataque ocorre quando $B = U$, ou seja, quando inicialização satura o filtro com a inserção de todos os elementos do conjunto universo. Isso pode ser facilmente implementado preenchendo-se todos os bits do Filtro de Bloom original com 1. Uma conseqüência direta desse ataque é o comprometimento da acurácia do sistema de rastreamento, já que são rastreadas outras rotas além da verdadeira rota de ataque. No caso extremo em que $B = U$, o grafo de reconstrução é formado pelo conjunto de todas as rotas da Internet, quando o ideal seria que ele fosse composto apenas pela rota de ataque.

O Filtro de Bloom original é um alvo fácil para ataques desse tipo. Por outro lado, o Filtro de Bloom Generalizado (FBG) foi projetado para ser robusto a ataques que visem a saturação do filtro. Com o uso do FBG, a probabilidade máxima de falso positivo, expressa pela Equação 3, é independente do estado do filtro e só depende do número de funções *hash* usadas. No entanto, os cálculos realizados supunham uma escolha aleatória dos bits iniciais. O problema principal é que o FBG não apaga os bits inicializados pelo atacante. Dessa forma, os bits iniciais podem influenciar os falsos positivos. Portanto, apesar de ser bem mais robusto do que o filtro original, o FBG está sujeito a ataques nos quais haja um ajuste cuidadoso da condição inicial do filtro. O ataque descrito nessa seção se encaixa perfeitamente nesse perfil e é bem simples de ser realizado. Basta que o atacante conheça o algoritmo de inserção do filtro.

Para o FBG, uma medida da robustez do filtro é a probabilidade de falso negativo dos elementos mais antigos. Quanto maior essa probabilidade, maior será a probabilidade de um elemento inserido não ser reconhecido em um teste de pertinência. No entanto, foi mostrado na Seção 3 que um aumento da probabilidade de falso negativo implica redução da capacidade de armazenamento do filtro. Há, portanto, um compromisso entre robustez e capacidade de armazenamento. A idéia fundamental do Filtro de Bloom Concatenado (FBC) é justamente combinar a robustez com uma alta capacidade de armazenamento. A construção do FBC é feita de tal maneira a mitigar os efeitos desse compromisso, permitindo que o FBC supere o FBG nesses dois quesitos. Na verdade, a construção do FBC é feita de forma a desacoplar a probabilidade de falso negativo da robustez do filtro: os falsos negativos foram eliminados e a robustez passou a ser garantida pela sobrescrição das informações iniciais. Esses resultados ficarão mais evidentes na próxima seção. Na Seção 6.2 será analisada a probabilidade de sucesso do atacante para cada um dos filtros apresentados nesse artigo.

5. O Filtro de Bloom Concatenado

Nessa seção é apresentada a proposta desse trabalho. Assim, como o Filtro de Bloom convencional, o Filtro de Bloom Concatenado (FBC) é uma estrutura de dados capaz de representar um conjunto de forma compacta. O FBC, porém, possui características adicionais que o tornam mais adequado para aplicações distribuídas nas quais a questão da segurança é crucial. A idéia central do filtro proposto é concatenar diversos filtros pequenos, denominados subfiltros. Somente um elemento é inserido em cada subfiltro. Duas principais vantagens advêm dessa construção. A primeira delas, intrinsecamente relacionada à questão da segurança, é que se podem sobrescrever as informações iniciais contidas no filtro, garantindo a robustez à interferência da condição inicial. A segunda

vantagem é a garantia de uma probabilidade de falso negativo nula, já que um bit de um elemento inserido no filtro jamais será invertido por uma futura inserção. A seguir, são detalhados os esquemas de inserção e o algoritmo de teste de pertinência.

Sejam m e N inteiros positivos tais que m é múltiplo de N. O FBC é definido como a concatenação de N vetores, denominados subfiltros, de m/N bits cada. O FBC é, portanto, constituído por um vetor de $N * (m/N) = m$ bits, que podem assumir qualquer valor inicial. Seja um conjunto $S = \{s_1, s_2, \ldots, s_n\}$ de n elementos ($n \leq N$). A inserção dos elementos de S é feita seqüencialmente de acordo com o índice de cada elemento. Assim, o elemento s_i ($1 \leq i \leq n$) deve ser inserido no i-ésimo subfiltro. O número máximo de elementos admitidos pelo FBC é N, que é arbitrado pelo projetista do filtro. A escolha do valor de N é uma decisão de projeto e varia de acordo com a aplicação. Em geral, N deve ser a cota superior para a estimativa do número de elementos que serão inseridos, de forma que todos os elementos sejam admitidos pelo filtro, mesmo no pior caso. Para algumas aplicações, como é o caso do rastreamento de pacotes IP [Laufer et al. 2006], é vantajoso que N seja fixo. Porém, se a aplicação permitir, o valor de N pode ser dinâmico, bastando alocar os subfiltros à medida que novos elementos são inseridos. Nesse último caso, a vantagem é o não desperdício de espaço com subfiltros inutilizados. Em todo caso, nesse trabalho N será suposto sempre fixo e suficientemente grande para que todas as inserções necessárias possam ser efetuadas.

O Algoritmo 3 realiza a inserção de elemento no FBC. O algoritmo recebe como argumentos de entrada um elemento $s_i \in S$ e o Filtro de Bloom Concatenado C. A seguir é aplicado um dos dois esquemas heurísticos de inserção diferentes propostos nesse trabalho. O esquema de inserção pode ser interpretado como a forma de "assinatura" de cada elemento. O primeiro esquema é uma adaptação do algoritmo de inserção de elemento do Filtro de Bloom original (Seção 2) na qual os zeros também representam informação. Essa adaptação tem como objetivo inverter, o máximo possível, os bits iniciais do filtro, no intuito de aumentar a robustez à interferência da condição inicial. O resultado dessa adaptação é analisado na Seção 6.2. O segundo esquema de marcação é bem simples. O vetor de marcação é simplesmente o resultado da aplicação de uma função *hash* $H : S \rightarrow \{0, 1, \ldots, 2^{m/N} - 1\}$ ao elemento s_i. Esse esquema é similar ao utilizado em [Yaar et al. 2003]. Uma vantagem desse tipo de inserção é que, ao contrário do Filtro de Bloom convencional e do FBG, se pode paralelizar o acesso ao vetor de bits.

Algoritmo 3: Inserção de Elemento no Filtro de Bloom Concatenado

Entrada: $s_i \in S$ e o vetor C de m bits.
Resultado: Inserção do elemenho s_i no filtro C
início
  selecione *esquema* **faça**
    caso *esquema 1*
      para $j = 0$ **até** $(m/N) - 1$ **faça**
        $C[i * (m/N) + j] \leftarrow 0$
      $inserirElementoFB(s_i, C[i * (m/N)])$
    caso *esquema 2*
      $C[i * (m/N)] \leftarrow H(s_i)$
fim

Propõem-se dois algoritmos de teste de pertinência para o FBC, que diferem basicamente nas suposições que são realizadas. Seja U o conjunto universo de todos os elementos. O primeiro algoritmo supõe que existe um mecanismo auxiliar, como um contador, que informa o índice do subfiltro no qual qualquer elemento $x \in U$ foi inserido ou, caso não tenha sido inserido, o mecanismo auxiliar informa o único subfiltro no qual x teria alguma chance de ser inserido. Dessa maneira, não é necessário checar todos os subfiltros para determinar a pertinência de x, bastando somente checar o subfiltro indicado pelo mecanismo auxiliar. O segundo algoritmo não faz nenhuma suposição adicional, sendo necessária a verificação de todos os subfiltros. Caso o elemento seja reconhecido por pelo menos um dos subfiltros, então o elemento é considerado membro do grupo. Pode-se questionar se a hipótese do primeiro algoritmo é viável ou não. A viabilidade dessa hipótese depende da aplicação em questão. Evidentemente, a probabilidade de ser positivo o resultado do segundo algoritmo é maior. Como o teste de pertinência será positivo para todo elemento $s_i \in S$ nos dois algoritmos, então a probabilidade de falso positivo do segundo algoritmo será maior. Logo, se na aplicação em questão a hipótese do primeiro algoritmo for viável, deve-se optar por esse algoritmo, em detrimento do segundo. No caso na aplicação do rastreamento IP, pode-se utilizar o campo TTL (*time-to-live*) do cabeçalho IP como o contador que indicará o índice do subfiltro em que o endereço de cada roteador tem chance de ter sido inserido. Por essa razão, nesse trabalho será analisado apenas o primeiro algoritmo. Uma análise do segundo algoritmo será deixada para um trabalho futuro.

O primeiro algoritmo é apresentado a seguir. O Algoritmo 4 recebe como argumentos de entrada um elemento x e o Filtro de Bloom Concatenado C. É chamado o procedimento $esquemaInsercao()$, que atualiza o vetor $mascaraInsercao$ com a "assinatura" do elemento x. Se o vetor $mascaraInsercao$ for exatamente igual ao i-ésimo subfiltro, então o algoritmo retorna 1, indicando teste de pertinência positivo. Caso contrário, o algoritmo retorna 0, indicando teste de pertinência negativo.

Algoritmo 4: Teste de Pertinência no Filtro de Bloom Concatenado

Entrada: Elemento x e o vetor C de m bits.
Saída: 1, se $x \in S$; 0, caso contrário.
início
  declare $mascaraInsercao[m/N]$ vetor de bits
  $esquemaInsercao(x, mascaraInsercao)$
  $i \leftarrow mecanismoAuxiliar(x)$
  para $j = 0$ **até** $(m/N) - 1$ **faça**
    se $C[i * (m/N) + j] \neq mascaraInsercao[j]$ **então retorna** *0*
  retorna *1*
fim

6. Resultados Analíticos

6.1. Falsos Positivos

Nessa seção é feita uma análise da probabilidade de falso positivo do FBC. Um falso positivo ocorre quando um elemento externo ao filtro possui uma "assinatura" idêntica ao do elemento que foi inserido no subfiltro em questão. Vale ressaltar que apenas um elemento é inserido em cada filtro e supõe-se que é conhecido o único subfiltro no

qual um dado elemento tem chance de ter sido inserido. Como cada esquema de inserção possui uma forma de "assinatura" diferente, cada um deles terá uma probabilidade de falso positivo diferente. Seja p a probabilidade de um bit específico ser zerado. Vale notar que p é igual à probabilidade de um bit estar zerado, após a inserção do elemento no subfiltro. Além disso, na média, $p(m/N)$ bits são zerados e $(1-p) * (m/N)$ bits são preenchidos com 1. Assim, em todos os esquemas, a probabilidade f_p de falso positivo será

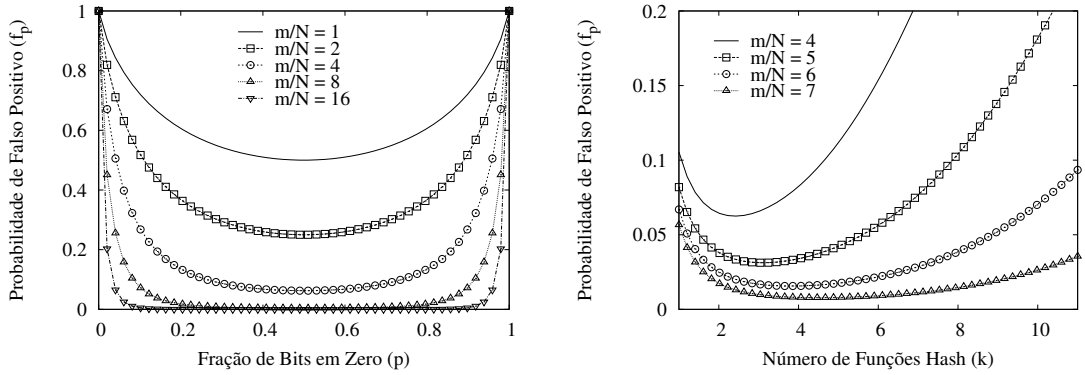
$$f_p = p^{p(m/N)}(1-p)^{(1-p)(m/N)}. \tag{4}$$

A diferença é que, para cada esquema, p, pode assumir valores distintos. No Esquema 1, p é a probabilidade de nenhuma das k funções *hash* preencher um bit com 1. Como as funções *hash* são uniformes e independentes, então

$$p = \left(1 - \frac{1}{(m/N)}\right)^k. \tag{5}$$

No Esquema 2, como a saída da função *hash* é uniformemente distribuída ao longo do seu contra-domínio, cada bit tem a mesma probabilidade de ser preenchido com 1 ou 0. Logo, p é simplesmente $1/2$.

A Figura 2(a) mostra o gráfico da probabilidade de falso positivo, dada pela Equação 4, em função da fração de bits em zero p. Em cada uma das curvas apresentadas tem-se um tamanho de subfiltro m/N diferente. Para a compreensão desse gráfico, é importante ter em mente que p representa tanto a fração de bits em zero como a probabilidade de um bit ser zerado numa inserção. Observa-se que, quando p tende a 1, f_p tende a 1. Isso ocorre porque, nessa situação, todos os bits do filtro estão em 0 e a probabilidade de um bit ser zerado é de 100%. Logo, a probabilidade de falso positivo tenderá a 1. Uma situação análoga ocorre quando p_0 tende a 0. Outra observação é que f_p diminui com o aumento da fração de bits por subfiltro m/N e, à medida que m/N aumenta, se alarga o intervalo no qual f_p é próxima de zero.



(a) Esquemas 1 e 2. (b) Esquema 1.

Figura 2. Probabilidade de falso positivo.

É importante notar que o gráfico possui eixo de simetria em $p = 1/2$. Nesse momento, será mostrado que esse é o ponto de mínimo de f_p. Se a equação 4 fosse dada em função do termo $1 - p$, ela seria expressa da mesma forma. Assim, por razões de simetria, $p^{(min)} = 1/2$. Substituindo esse valor na Equação 4, determina-se a probabilidade mínima de falso positivo $f_p^{(min)}$, dada por

$$f_p^{(min)} = \left(\frac{1}{2}\right)^{m/N}. \tag{6}$$

Pode-se observar que o decaimento de f_p com o aumento número de bits por subfiltro m/N é exponencial, de forma que f_p atinge valores bem próximos a zero mesmo quando a razão m/N ainda assume valores razoavelmente pequenos. Nota-se que, no Esquema 2, p é sempre igual a $1/2$. Logo, nesse caso, a probabilidade de falso positivo é sempre mínima, sendo expressa pela Equação 6. Já no Esquema 1, p depende também do número k de funções *hash*. Porém, o gráfico da Figura 2(b) mostra que existe um valor ótimo de k que minimiza f_p. Apenas graficamente, podemos observar que isso ocorre quando k é ligeiramente menor do que m/N. Analiticamente, o valor ótimo de k é dado por

$$k^{(min)} = -\frac{\ln(2)}{\ln\left(1 - \frac{1}{(m/N)}\right)}. \tag{7}$$

Para se chegar a essa expressão, basta igualar a Equação 5 a $1/2$. Esse resultado mostra que os dois esquemas de inserção propostos possuem a mesma probabilidade de falso positivo mínima.

6.2. Robustez

Nessa seção, é analisada a robustez à interferência do atacante do Filtro de Bloom original e do filtro proposto nesse trabalho. É usada como métrica de robustez a probabilidade p_a de sucesso do atacante ao executar o ataque descrito na Seção 4. Quanto menor a probabilidade p_a, maior será a robustez do filtro. É considerado que o atacante teve sucesso se o elemento por ele inserido tiver teste de pertinência positivo após a inserção de n elementos legítimos. Se pelo menos um bit inserido pelo atacante for invertido pelas n inserções posteriores, então o ataque não terá sucesso. Assim, p_a é a probabilidade de nenhum bit inserido pelo atacante ser invertido.

No FBC, para todos os esquemas de inserção, p_a será a probabilidade da "assinatura" do elemento inserido pelo atacante ser igual à "assinatura" do elemento legítimo inserido posteriormente no mesmo subfiltro. Isso é equivalente à probabilidade de falso positivo. Portanto,

$$p_a = f_p^{(min)} = \left(\frac{1}{2}\right)^{m/N}. \tag{8}$$

Uma observação importante é que, no Esquema 1, essa probabilidade seria maior, caso não fosse feita a adaptação apresentada na Seção 5, que visa inverter o máximo possível os bits iniciais do filtro. Sem essa adaptação, os bits inseridos pelo atacante que permanecessem intocados pela inserção do elemento legítimo não seriam invertidos e, portanto, a probabilidade de sucesso do atacante seria maior. Agora, basta que apenas um desses bits seja invertido pela adaptação que o ataque irá falhar.

Já o Filtro de Bloom original não possui nenhum mecanismo que garanta robustez à interferência do atacante. Assim, como nenhum bit inserido pelo atacante é invertido, então a probabilidade de sucesso do atacante $p_a = 1$. Isso significa que um nó malicioso pode comprometer completamente uma aplicação distribuída no qual o filtro original é utilizado. No caso do rastreamento de pacotes, o atacante consegue burlar o sistema de rastreamento e garantir o seu anonimato.

7. Conclusões

Nesse trabalho, é proposto um Filtro de Bloom robusto à interferência da condição inicial denominado Filtro de Bloom Concatenado (FBC). O FBC possui a propriedade de ser robusto a ataques, além da característica do Filtro de Bloom original de representar um conjunto de forma compacta. Portanto, o FBC é apropriado para aplicações distribuídas onde o filtro é transferido em um meio inseguro e a questão da segurança é crucial. A idéia chave da proposta é concatenar diversos subfiltros, cada qual representando um elemento. Esta estrutura permite uma alta robustez à interferência da condição inicial obtida com a ação de sobrescrever o conteúdo original de cada subfiltro de forma que o seu conteúdo original seja apagado. Assim, é garantido que a informação inicial originada pelo atacante não influencia a probabilidade de falso positivo. Além disso, essa técnica não introduz falsos negativos, que poderiam causar perda de informações legítimas e limitação da capacidade de armazenamento. Como somente um elemento é inserido por subfiltro, podem-se sobrescrever as informações iniciais sem perda de informação legítima.

São obtidas expressões analíticas para os falsos positivos e os resultados mostram que a probabilidade de sucesso do atacante decresce exponencialmente com o tamanho de cada subfiltro.

É mostrado que o Filtro de Bloom convencional não é robusto a ataques, pois ao saturar o filtro a probabilidade de sucesso do atacante é de 100%. Quando comparado ao Filtro de Bloom Generalizado (FBG), o FBC não apresenta falsos negativos e mostra-se bem mais robusto para o mesmo tamanho do filtro. Os falsos negativos são eliminados pela construção específica da representação de um único elemento por subfiltro. A maior robustez é obtida pela sobrescrita dos subfiltros. No Filtro de Bloom Generalizado, existe um compromisso entre a robustez e a probabilidade de "esquecimento" dos elementos mais antigos. Quanto maior a robustez, maiores são as taxas de "esquecimento" do FBG, interpretadas como uma forma de limitação de memória do filtro.

Os resultados indicam que o Filtro de Bloom Concatenado proposto se apresenta como uma excelente opção para representação de elementos em aplicações distribuídas eficientes e seguras devido a sua compacidade e robustez. Com apenas 4 bits por subfiltro, a probabilidade de sucesso do atacante é de 6,25% e, com 8 bits por subfiltro, essa probabilidade cai para aproximadamente 0,39%

Referências

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

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

Cohen, S. e Matias, Y. (2003). Spectral Bloom Filters. Em *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, páginas 241–252, San Diego, CA, EUA.

Dharmapurikar, S., Krishnamurthy, P., Sproull, T. S. e Lockwood, J. W. (2004). Deep Packet Inspection Using Bloom Filters. *IEEE Micro*, 24(1):52–61.

Estan, C. e Varghese, G. (2003). New Directions in Traffic Measurement and Accounting: Focusing on the Elephants, Ignoring the Mice. *ACM Transactions on Computer Systems*, 21(3):270–313.

Fan, L., Cao, P., Almeida, J. e Broder, A. Z. (2000). Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. *IEEE/ACM Transactions on Networking*, 8(3):281–293.

Kumar, A., Xu, J., Wang, J., Spatschek, O. e Li, L. (2004). Space-Code Bloom Filter for Efficient Per-Flow Traffic Measurement. Em *Proceedings of the IEEE INFOCOM 2004 Conference*, páginas 1762–1773, Hong Kong, China.

Laufer, R. P., Velloso, P. B., de O. Cunha, D., Moraes, I. M., Bicudo, M. D. D. e Duarte, O. C. M. B. (2005a). A New IP Traceback System against Denial-of-Service Attacks. Em *12th International Conference on Telecommunications - ICT'2005*, Cidade do Cabo, África do Sul.

Laufer, R. P., Velloso, P. B., de O. Cunha, D., Moraes, I. M., Bicudo, M. D. D., Moreira, M. D. D. e Duarte, O. C. M. B. (2006). Towards Stateless Single-Packet IP Traceback. Relatório Técnico GTA-06-38, COPPE/UFRJ.

Laufer, R. P., Velloso, P. B. e Duarte, O. C. M. B. (2005b). 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.

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

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

Shanmugasundaram, K., Brönnimann, H. e Memon, N. (2004). Payload Attribution via Hierarchical Bloom Filters. Em *Proceedings of the 11th ACM conference on Computer and Communications Security*, páginas 31–41, Washington, DC, EUA.

Yaar, A., Perrig, A. e Song, D. (2003). Pi: A path identification mechanism to defend against ddos attacks.