Rastreamento Eficiente de Pacotes na Internet

Egon Hilgenstieler[1], **Elias P. Duarte Jr.**[1], **Keiko Verônica Ono Fonseca**[2]

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Caixa Postal 19.018 – CEP 81.531-990 – Curitiba – PR – Brasil

[2]CPGEI – Universidade Tecnológica Federal do Paraná (UTFPR)
Av. Sete de Setembro 3165 – CEP 80.230-901 – Curitiba – PR – Brasil

{egon,elias}@inf.ufpr.br, keiko@cpgei.cefetpr.br

Abstract. *This work presents a strategy that allows the determination of the source and the route traversed by a packet received from the Internet. The Internet architecture does not present this funcionality, that can be used, for instance, after the detection of an attack to determine its origin. Separate traffic logs are kept for each router interface; logs are efficiently stored in Bloom filters. The comunication among the system components preserves the confidentiality of the packet's information. A dynamic log paging strategy is also defined. The architecture was implemented and experimental results are presented.*

Resumo. *Este trabalho apresenta uma estratégia que permite determinar a origem e a rota percorrida por um pacote recebido da Internet. A arquitetura da Internet não tem esta funcionalidade, que pode ser usada, por exemplo, após a detecção de um ataque para determinar sua rede de origem. São mantidos logs separados de tráfego para cada uma das interfaces dos roteadores participantes, armazenados de forma eficiente em filtros de Bloom. A comunicação entre os componentes do sistema é realizada de maneira a preservar a confidencialidade do pacote. Uma estratégia de paginação dinâmica dos logs também é definida. A arquitetura foi implementada e resultados experimentais obtidos são apresentados.*

1. Introdução

Infelizmente a Internet não é segura. Diversas ferramentas estão disponíveis para que um atacante desabilite um serviço disponível na rede, se aproveitando de *bugs* em implementações de software ou hardware, ou simplesmente inundando a rede com requisições legítimas. Ataques do tipo negação de serviço são reportados mais frequentemente [1], onde a rede é inundada com uma grande quantidade de tráfego. Entretanto, outros tipos de ataques podem ser conduzidos usando uma quantidade significativamente menor de pacotes. Historicamente, observamos implementações de hardware e sistemas operacionais que puderam ser desabilitados com apenas um único pacote [2]. Infelizmente, não é possível determinar a origem de um pacote IP de maneira confiável. O endereço de um pacote pode ser alterado (*IP Spoofing*) para esconder sua real origem e a arquitetura de roteamento da Internet também não mantém informações sobre os pacotes processados.

O objetivo do rastreamento de pacotes IP é, dado um pacote IP, o horário aproximado do seu recebimento e o seu destino (chamado de *vítima*) construir um *grafo de*

ataque. Consideramos que um pacote pode conter múltiplas fontes. Este grafo de ataque consiste então da união dos caminhos formados por cada roteador por onde o pacote passou em seu caminho até a vítima. São chamados de *falsos positivos* os nós do grafo de ataque que não processaram realmente o pacote. Falsos positivos podem ocorrer, por exemplo, se um roteador for subvertido por um atacante.

Vários problemas devem ser considerados na definição de uma arquitetura de rastreamento. Os pacotes, por exemplo, podem ser modificados durante o processo de roteamento. Algumas transformações possíveis são descritas no RFC 1812 [15] tais como fragmentação de pacotes, processamento de opções do pacote IP, processamento de pacotes ICMP e duplicação de pacotes. Outras formas de transformações incluem NAT, tunelamento IPsec [17] e *IP-in-IP* [16]. Muitas destas transformações resultam em perdas irrecuperáveis do estado do pacote original. Um sistema de rastreamento ideal deve ser capaz de rastrear pacotes que sofrem transformações válidas até a sua origem.

As implicações de privacidade também são importantes no projeto de um sistema de rastreamento. O conteúdo dos pacotes processados deve ser devidamente protegido. Além disso, como um sistema de rastreamento possivelmente irá requerer a cooperação entre diversos sistemas autônomos (AS - *Autonomous Systems*) é desejável que mesmo as informações do pacote sendo rastreado também sejam protegidas.

Assume-se que os *hosts* de origem e destino são pobres em recursos, em particular a vítima de um ataque. Isto vem do fato de que uma grande variedade de dispositivos como câmeras, microscópios, etc. podem estar conectados à Internet mas raramente têm recursos além daqueles necessários para sua função.

Diversas arquiteturas foram propostas mostrando que o rastreamento de pacotes IP é viável [7, 8, 12]. Estas arquiteturas baseiam-se na estratégia de armazenar *logs* dos pacotes em pontos estratégicos da rede. A partir de algoritmos apropriados de extração, é possível determinar a origem de um pacote IP recente com razoável precisão. A eficiência de cada uma destas arquiteturas pode ser analisada por diversas métricas independentes. Estas métricas, por sua vez, dependem de um conjunto de parâmetros.

Neste artigo apresentamos uma arquitetura que procura aumentar a eficiência de rastreamento de pacotes na Internet. Em comparação com o trabalho apresentado em [12] a arquitetura aqui proposta tem diferenças significativas. Neste trabalho a operação de rastreamento mantém o sigilo das informações trafegadas. Propõe-se o uso de *logs* por interface do roteador ao invés de um *log* para o roteador como um todo, o que implica em um forte impacto no custo da operação de rastreamento pois o número de requisições é reduzido ao mínimo. Novos algoritmos de rastreamento são propostos neste trabalho. Três métricas são analisadas: a precisão do grafo de ataque obtido, permitindo determinar corretamente não apenas a origem mas também a rota percorrida pelo pacote; o custo da operação de rastreamento, medido pelo número de requisições na rede e o período de tempo pelo qual um pacote recebido pode ser rastreado.

O restante deste trabalho esta organizado como segue. A seção 2 apresenta o histórico de trabalhos relacionados. A seção 3 apresenta a arquitetura proposta. A implementação, bem como resultados experimentais são apresentados na seção 4. Conclusões seguem na seção 5.

2. Trabalhos Relacionados

Os primeiros esforços para rastreamento na Internet visavam determinar a rota de um fluxo de pacotes, em contraposição ao rastreamento de pacotes individuais. É possível determinar a rota deste fluxo basicamente de duas maneiras: observando-o enquanto ele atravessa a rede ou tentando inferir a rota baseada no impacto deste no estado da rede. Em [5] algumas redes candidatas eram sistematicamente inundadas com pacotes. Desta maneira é possível observar variações no fluxo de pacotes recebidos e inferir a rota percorrida pelo fluxo. Para tanto é necessário ter conhecimento da topologia da rede e ter a capacidade de gerar grandes fluxos de dados para um enlace de uma rede arbitrária.

Além destas, outras técnicas foram propostas para observar o fluxo de pacotes. Ao fornecer informações parciais através de um subconjunto de pacotes em um fluxo, os roteadores podem permitir que a máquina final reconstrua todo o caminho percorrido pelos pacotes depois de receber uma determinada quantidade de pacotes. Dois esquemas foram propostos para comunicar as informações referentes ao caminho percorrido pelo fluxo às máquinas finais. Em [3] as informações são codificadas em campos raramente utilizados no cabeçalho do pacote IP. Já em [4] as informações de auditoria são enviadas através de mensagens ICMP (Internet Control Message Protocol).

Em todas estas abordagens, uma grande quantidade de pacotes é necessária para conseguir inferir a rota. Estas propostas são ineficazes para ataques conduzidos com uma quantidade pequena de pacotes. Outra abordagem é rastrear os pacotes IP individualmente. Desta maneira, uma vez identificado um ataque poderia ser possível identificar a sua origem utilizando apenas um dos pacotes envolvidos neste ataque. A arquitetura SPIE (*Source Path Isolation Engine*) [8] foi a primeira arquitetura proposta para este fim. Esta arquitetura mantém em cada roteador um *log* dos pacotes processados mais recentemente. Como a quantidade de armazenamento é limitada, registros mais novos ocupam os registros mais antigos quando necessário. Se um pacote for julgado interessante por alguma métrica, por exemplo por um sistema de detecção de intrusão, uma consulta é realizada em cada roteador pelo registro do pacote.

O armazenamento de pacotes no roteador requer quantidades muito grandes de espaço, mesmo para enlaces de baixa velocidade e por curtos períodos de tempo. Este armazenamento possui também sérias implicações de privacidade. Para resolver este problema a arquitetura SPIE utiliza os chamados filtros de Bloom [9] para armazenar apenas os resumos digitais (*hash*) de cada pacote. Um filtro de Bloom é uma estrutura de dados utilizada para armazenar um determinado conjunto de elementos, possibilitando uma operação de consulta eficiente. Para ser possível identificar univocamente um pacote em qualquer hop da rede e diminuir o tamanho dos dados de entrada para o filtro de Bloom são utilizados apenas alguns campos do cabeçalho do pacote IP totalizando 20 bytes mais 8 bytes do seu conteúdo. Os componentes da arquitetura SPIE e suas relações estão ilustradas na figura 1.

As tarefas necessárias para realizar o rastreamento são divididas em três componentes. O DGA (*Data Generation Agent*) é o componente que fica próximo ao roteador e que armazena a tabela com o registro dos pacotes processados, pode ser implementado em software ou hardware [11]. O SCAR (*SPIE Collection and Reduction Agent*) é um componente responsável por vários DGAs. Ao ser consultado, ele recupera os registros de cada DGA para montar um grafo de ataque parcial de sua região. O STM (*SPIE Trace-*

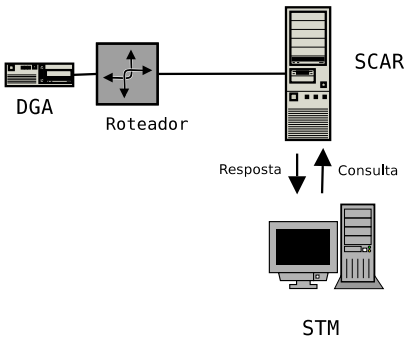

Figura 1. A arquitetura SPIE.

back Manager) é o componente que é a interface com a entidade interessada em realizar o rastreamento e o que gerencia todo o sistema.

Uma requisição ao STM é composta por um pacote de ataque e o horário aproximado de seu recebimento. Ao receber esta requisição, o STM imediatamente envia uma requisição a todos os SCARs de seu domínio. Estes, por sua vez, recuperam os registros de todos os seus respectivos DGAs para análise. Em seguida, cada SCAR simula uma inundação do caminho reverso (*Reverse Path Flooding*) examinando os registros armazenados localmente. Neste algoritmo, uma lista de todos os nós já visitados é mantido para garantir o seu término. Por fim, cada SCAR retorna um subgrafo parcial de sua região para o STM, que monta o grafo de ataque final retornando-o ao solicitante.

Um sistema de rastreamento deve ser capaz de reconstruir o pacote original a partir do pacote transformado. Entretanto, alguns tipos de transformações ocasionam perda de informação do pacote. Desta maneira, informações suficientes devem ser armazenadas pela arquitetura SPIE para poder reconstruir os pacotes originais. Em conjunto com as informações armazenadas pelo DGA, a arquitetura SPIE mantém uma tabela de transformações para o mesmo intervalo de tempo, chamada de *Transform Lookup Table* ou TLT. Cada entrada na TLT contém três campos: O primeiro campo armazena o resultado da função hash para o referido pacote. O segundo campo especifica o tipo de transformação (três bits são suficientes para identificar os tipos de transformações). O último campo contém informações de tamanho variado do pacote. O tamanho utilizado depende da transformação utilizada.

3. Uma Arquitetura para Rastreamento de Pacotes IP

A arquitetura proposta neste trabalho, assim como na arquitetura SPIE, também armazena informações sobre os pacotes processados na própria rede. O rastreamento de pacotes é feito de maneira mais eficiente, obtendo um grafo de ataque mais preciso a um custo mínimo de requisições na rede. Uma nova estratégia de paginação dos registros em memória secundária também é utilizada.

A arquitetura é composta basicamente por três componentes. O *Packet Monitor* é responsável por manter o *log* dos pacotes que trafegam pela rede, armazenando-os em uma base de registros chamada de *Packet Record Base* (PRB) para posterior consulta. O *Packet Record Agent* (PRA) é o componente responsável por receber requisições para verificar se um dado pacote foi visto pelo *Packet Monitor.* Recebe como entrada um

pacote e o horário aproximado de sua chegada e responde se ele foi visto ou não. Caso positivo, responde também com os PRAs vizinhos que podem também ter visto o pacote. O *Packet Tracer Application* (PTA) é a aplicação que efetua o rastreamento de um pacote, recebendo como entrada o pacote, o horário de sua chegada, e o PRA mais próximo da vítima. O PTA consulta os PRAs necessários para determinar o caminho percorrido pelo pacote e responde com um grafo de ataque. A figura 2 ilustra os componentes desta arquitetura.

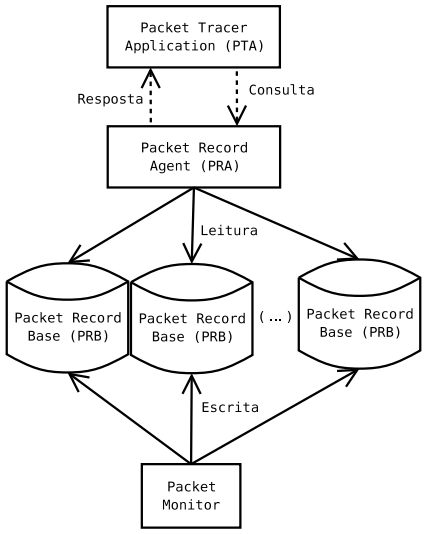

Figura 2. Componentes da arquitetura proposta.

Conforme descrito em [8], o armazenamento de pacotes inteiros em uma base de dados se torna inviável devido tanto a questões de privacidade como à grande quantidade de espaço necessário. Logo, é utilizada a mesma estratégia do componente DGA visto da arquitetura SPIE, e calculado e armazenado apenas o resumo digital de cada pacote observado.

A seguir é detalhada a estratégia de armazenamento das informações que permitem o rastreamento. Também são apresentados os algoritmos que podem ser utilizados para extração do grafo de ataque. São avaliados o número de requisições transmitidas entre os componentes e os requisitos de armazenamento.

3.1. Armazenamento de Informações para o Rastreamento

Apenas uma parte do pacote é usado como entrada para a função hash, de maneira que seja possível representá-lo de maneira única e mantendo o menor tamanho possível. A figura 3 mostra um pacote IP e os campos incluídos para o cálculo da função hash. O Packet Monitor calcula o hash apenas sobre a porção invariante do cabeçalho IP mais o TTL (*Time-to-Live*) e os 8 primeiros bytes do conteúdo totalizando 30 bytes. Outros campos que são frequentemente alterados são mascarados antes do resumo do pacote ser calculado. Os resultados apresentados em [8] mostram que 28 bytes (20 bytes de campos do cabeçalho e 8 bytes do conteúdo) são suficientes para identificar quase todos os pacotes não idênticos. O componente DGA da arquitetura SPIE, por outro lado, também mascara o campo TTL do pacote antes de calcular o seu resumo digital.

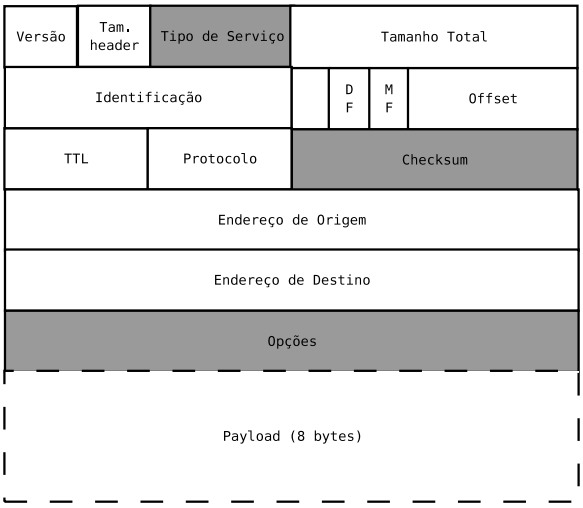

Figura 3. Um pacote IP; campos em cinza não são usados na função hash.

Mesmo o armazenamento de todos os resumos digitais gerados pelo tráfego que passa pelo roteador requer quantidades muito grandes de armazenamento. Para resolver este problema, o Packet Monitor também utiliza um filtro de Bloom para armazenar os resumos de cada pacote.

Um filtro de Bloom calcula *k* resumos distintos para cada pacote usando funções hash independentes. O tamanho dos resumos gerados são todos de *n* bits. Um vetor de 2^n bits é inicializado com zeros e as posições endereçadas pelo resultado dos resumos gerados são setadas para um. Em outras palavras, se hash(*pacote*)=y, então Bloom[y]=1. À medida em que outros pacotes vão chegando, os bits do vetor vão sendo setados. A figura 4 ilustra a utilização de um filtro de Bloom usando *k* funções hash. Um mesmo vetor é utilizado apenas por um determinado período de tempo. Depois deste período outro vetor será utilizado. O conjunto destes vetores forma uma tabela que representa o tráfego que passou pelo roteador em um determinado período de tempo.

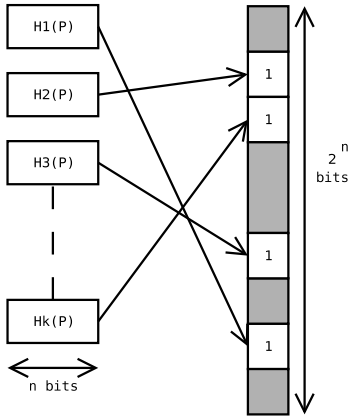

Figura 4. Utilização de um filtro de Bloom.

A operação de consulta é realizada computando o resultado das *k* funções hash

e verificando os bits correspondentes do vetor. Se os bits forem zero existe a certeza de que o pacote não foi armazenado. Entretanto, se os bits forem um, existe uma grande possibilidade de que o pacote foi armazenado. Entretanto, há a possibilidade de outras inserções causarem estes bits a serem setados, criando assim um *falso positivo.* A taxa de ocorrência destes falsos positivos pode ser controlada [10], dependendo de k e do fator de capacidade, definido como $c = m/n$, sendo m o tamanho do vetor de bits e n o número de elementos inseridos.

Um sistema de rastreamento pode ser avaliado através de três métricas. A primeira métrica é a precisão do grafo de ataque retornado, comparado com o grafo de ataque real. A segunda métrica é o período de tempo pelo qual um pacote pode ser visto. Quanto maior este período, maior o tempo em que requisições de rastreamento podem ser processadas. Por último, é importante que a operação de rastreamento seja eficiente, utilizando um mínimo de requisições pela rede, sendo esta a terceira métrica. Este trabalho possui três contribuições principais: a de produzir um grafo de ataque mais preciso buscando obter exatamente o caminho inverso daquele percorrido pelo pacote através do processo de roteamento; reduzir o número de requisições da operação de rastreamento distribuindo o tráfego de acordo com o número de interfaces do roteador e de oferecer um mecanismo para aumentar o período de tempo pelo qual os registros de pacotes podem ser mantidos, assumindo redes que possuem variações no seu nível de utilização.

Em [8], a precisão do grafo de ataque é definida em termos do número de falsos positivos que o grafo de ataque contém. Entretanto, na arquitetura SPIE podem ocorrer também o que chamamos de *falsas arestas*, que são eliminadas na arquitetura proposta. Se existir uma ou mais arestas que conectem dois nodos da rede que fazem parte do grafo de ataque podem ser incluídas arestas que não pertencem ao caminho de ataque real, como ilustra a figura 5. As flechas representam o caminho do ataque. Linhas contínuas representam os enlaces entre os nodos. O grafo de ataque que é retornado é mostrado pelas linhas pontilhadas.

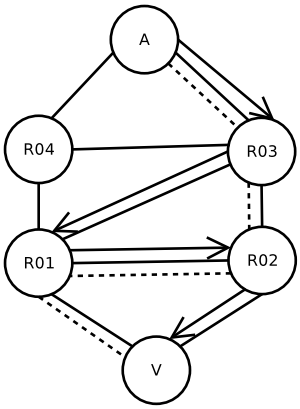

Figura 5. Exemplo de um grafo de ataque real e o obtido pelo sistema SPIE.

Apesar do enlace entre o nodo V e o nodo R01 não ter sido utilizado pelo pacote, ele está contido no grafo de ataque porque ambos os nodos observaram o mesmo pacote, mas o caminho real percorrido passa por R02. Por outro lado, a aresta que conecta R01 e R02 não pertence ao grafo de ataque porque R02 já foi visitado após a visita a R01, logo,

R02 foi posto na lista de nós já visitados, não sendo mais consultado novamente. Este grafo além de impreciso é mais difícil de ser interpretado. Pode ser entendido que houve duas fontes distintas para o mesmo pacote, ou que talvez um dos nodos seja na verdade um falso positivo.

A arquitetura proposta resolve este problema considerando a variação do campo TTL (*Time To Live*) no processo de rastreamento. Desta forma, utiliza-se também o campo TTL como entrada para o cálculo do resumo digital do pacote e um algoritmo apropriado para busca do caminho, incrementando o TTL a cada requisição. O incremento do TTL garante o término do algoritmo de busca.

Esta abordagem implica necessariamente em um aumento no número de requisições geradas pela operação de rastreamento conforme podem ser observados pelos resultados obtidos em [12]. É importante ressaltar que a operação de rastreamento, apesar de não ser frequente, precisa ser efetuada rapidamente. Os PRAs devem ser consultados enquanto os registros referentes ao período de tempo em questão ainda se encontram no PRB.

3.2. Algoritmo de Rastreamento

O rastreamento de pacotes IP levanta outras questões que não são essencialmente técnicas. Um rastreamento de pacotes IP na Internet geralmente irá envolver diferentes sistemas autônomos pertencentes a diferentes entidades administrativas. Um sistema autônomo pode desejar não expor nenhuma parte do conteúdo do pacote sendo rastreado, mesmo que sejam apenas alguns campos do pacote completo. Nesta situação a arquitetura permite que seja realizado um rastreamento apenas através de consultas envolvendo o resumo digital do pacote. Esta tarefa não é trivial pois a cada consulta ao PRA o TTL do pacote deve ser incrementado, gerando assim um resumo digital completamente novo.

Duas alternativas são apresentadas. A busca pode ser realizada iterativamente por um PTA ou recursivamente cada PRA pode consultar seus vizinhos. Na primeira abordagem o PTA consulta o primeiro PRA com o resumo digital do pacote com o seu TTL decrementado em uma unidade. O PRA retorna os vizinhos que efetivamente viram o pacote e o PTA, por sua vez, consulta estes vizinhos com o novo resumo digital construído a partir do pacote original com o seu TTL decrementado em duas unidades. O procedimento é repetido até que não existam mais PRAs a serem consultados. A figura 6 ilustra o funcionamento deste algoritmo. Linhas contínuas representam o enlace entre os nodos e linhas pontilhadas representam as consultas e respostas dos componentes.

A segunda abordagem consiste em uma busca recursiva efetuada pelo próprio PRA. O PTA inicia a consulta ao primeiro PRA e este, por sua vez, consulta seu vizinho e assim sucessivamente. Eventualmente não haverá mais vizinhos a serem consultados e o último PRA irá responder a requisição recebida. Cada PRA então retorna ao seu predecessor o grafo de ataque recebido acrescentado com o seu próprio endereço. Por fim, o PTA recebe o grafo de ataque completo.

Como cada consulta deverá ser efetuada com um resumo digital diferente devido ao decremento do TTL cada consulta deve ser composta por uma lista de todos os resumos digitais que serão necessários nas consultas subsequentes. A cada passo da recursão do algoritmo, um dos resumos é retirado da lista para efetuar a consulta repassando a lista remanescente. A desvantagem desta abordagem é o aumento do tamanhos das requisições.

A figura 7 ilustra o funcionamento deste algoritmo.

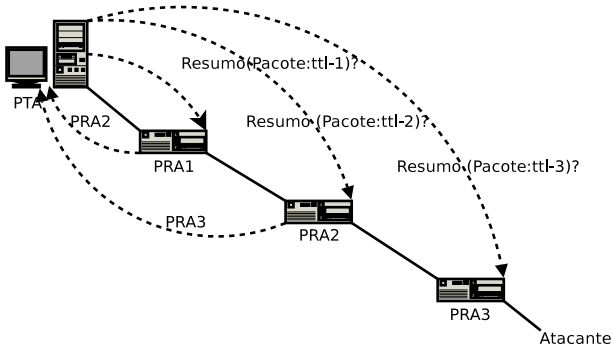

Figura 6. Algoritmo de rastreamento iterativo.

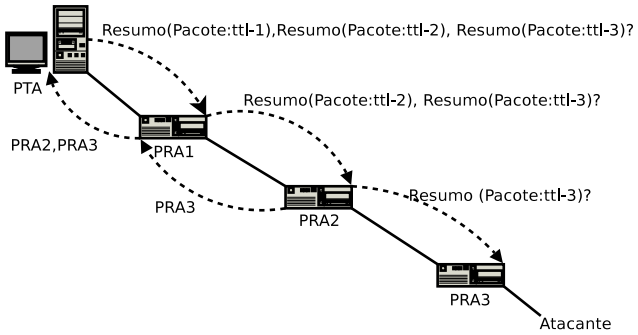

Figura 7. Algoritmo de rastreamento recursivo.

3.3. Redução do Número de Requisições

Na arquitetura proposta um *Packet Record Base* é mantido para cada interface do roteador. Para cada interface, por sua vez, existe um PRA vizinho que provavelmente também processou este pacote. Desta maneira, podemos guiar a consulta de maneira mais eficiente. O PRA ao ser requisitado, verifica em todos os respectivos PRBs se o pacote foi visto. Como resposta, são fornecidos apenas os PRAs associados ao PRB que contém o pacote, e não toda a lista de vizinhos. Com esta modificação, com excessão das requisições geradas pelas ocorrências de falsos positivos, serão consultados apenas os nodos que realmente observaram o pacote.

Por exemplo, considere um roteador com 5 interfaces. Pela abordagem anterior, se durante o processo de rastreamento fosse descoberto que este roteador tivesse visto um determinado pacote, todos os seus 5 roteadores vizinhos deveriam ser consultados em seguida para verificar se eles também processaram o pacote. Pela nova abordagem, sabe-se exatamente qual dos vizinhos precede este roteador no caminho de ataque, pois o armazenamento do *log* é dependente da interface, gerando apenas uma requisição ao invés de 5, desconsiderando os falsos positivos e a eventual duplicação de pacotes.

Como para cada requisição a um único PRA serão consultados vários filtros de Bloom, a taxa de falsos positivos como um todo será maior. Analiticamente podemos

demonstrar que, para um roteador com n interfaces e um filtro de Bloom com taxa de falsos positivos p, a probabilidade de que se obtenha pelo menos um falso positivo será de $1-(1-p)^n$. Para cada falso positivo será feita pelo menos uma requisição a mais pelo PTA.

3.4. Redução dos Requisitos de Armazenamento

Para utilizar uma quantidade menor de armazenamento para o vetor de bits do filtro de Bloomi, o *Packet Monitor* desta arquitetura utiliza uma outra abordagem para paginação do vetor em memória secundária. O DGA no sistema SPIE também utiliza um filtro de Bloom, paginando o vetor de bits gerado em uma memória secundária para consultas posteriores. Entretanto, esta paginação ocorre através de um período de tempo fixo pré-determinado. O *Packet Monitor* desta arquitetura, por outro lado, recebe como parâmetro o fator de capacidade máximo, sendo paginado apenas quando atingir este valor. Desta maneira, o período de tempo em que um filtro de Bloom permanece em memória principal é variável, podendo ser mais longo em períodos de baixa atividade da rede e mais curto em períodos de maior atividade, aproveitando ao máximo o vetor de bits utilizado.

Ao atingir o fator de capacidade máximo, o filtro de Bloom é paginado em memória secundária junto com outras informações de controle. O conjunto de registros armazenados em memória secundária é chamado de *Packet Record Base* (PRB). O tamanho máximo que pode ser ocupado por este conjunto de registros é determinado na configuração do *Packet Monitor.* Ao atingir o tamanho máximo, os registros mais antigos vão sendo substituídos.

4. Implementação e Resultados Experimentais

O *Packet Monitor* foi implementado em software como um *sniffer* usando a biblioteca PCAP (*Packet Capture*) [13]. Para cada interface de rede existe uma instância do *Packet Monitor* sendo executada como uma *thread*. Cada *Packet Monitor* possui seu próprio filtro de Bloom. O *Packet Record Agent* é executado como um processo separado do *Packet Monitor*. Ao receber uma requisição, o PRA consulta os filtros de Bloom de cada *Packet Monitor* utilizando compartilhamento de memória. Um arquivo de configuração indica quais são os PRAs vizinhos e por quais interfaces eles estão conectados. O PRA fornece como resposta apenas os vizinhos que realmente viram o pacote, ou fornece uma resposta negativa caso nenhuma interface tenha recebido o pacote. O *Packet Tracer Application* recebe como parâmetro o endereço do PRA mais próximo, que consiste no seu endereço IP e porta, e um arquivo binário que contém o pacote e o seu horário de recebimento. O formato deste arquivo é o mesmo utilizado pela biblioteca PCAP, que pode ser gerado pelo utilitário *tcpdump*.

Diversas simulações foram realizadas para comparar o número de requisições com outras abordagens, desconsiderando a taxa de falsos positivos. Uma avaliação analítica também foi realizada demonstrando a relação entre o número de interfaces, a taxa de falsos positivos individual de cada filtro de Bloom e a taxa de falsos positivos total.

O que identifica um PRA é o endereço IP e porta, desta maneira, foi possível simular uma rede de 50 nodos utilizando uma única máquina. Um tráfego sintético foi inserido nos respectivos PRBs. Um caminho de ataque foi simulado inserindo um mesmo pacote em diversos monitores da rede, variando apenas o campo TTL.

Foram gerados 20 grafos com 50 nodos utilizando o método de Waxman [14] para geração de grafos aleatórios que representam topologias similares às da Internet. Neste método, a probabilidade de existir uma aresta entre dois vértices varia de acordo com a distância entre tais vértices, tentando assim capturar a característica de localidade existentes nas redes reais. Dois valores, α e β são usados como parâmetros para geração do grafo. Um aumento em α aumenta o número de arestas no grafo. Um aumento em β aumenta a proporção de arestas longas sobre arestas curtas. Para cada grafo foi gerado um caminho aleatório. Foram gerados caminhos com 10, 15, 20, 25 e 30 nodos.

Duas métricas foram analisadas na simulação. A primeira foi a precisão do grafo de ataque em termos do número de falsas arestas. A segunda foi o número de requisições geradas pelo rastreamento. Foram utilizados três algoritmos para realizar o rastreamento. O primeiro foi uma simulação do algoritmo *Reverse Path Flooding* sendo que o campo TTL do pacote era mascarado pelo *Packet Monitor*, como é feito no sistema SPIE. Neste caso, uma lista dos nodos já visitados deve ser mantida para garantir o término do algoritmo. Na segunda simulação foi utilizado o algoritmo proposto sem utilizar o número mínimo de requisições, por último, foi utilizada esta melhoria no algoritmo. Nos dois últimos casos o campo TTL não é mascarado. Os resultados são apresentados na tabela 1.

Com relação à primeira métrica, o grafo de ataque obtido não mostra nenhuma falsa aresta das duas abordagens que não mascaram o campo TTL, ou seja o grafo de ataque obtido é idêntico ao grafo de ataque real. Na arquitetura SPIE, por outro lado, apenas em uma simulação o grafo de ataque obtido foi idêntico ao grafo de ataque real. Observa-se um aumento no número de falsas arestas no algoritmo do sistema SPIE de acordo com o aumento do número de arestas no grafo. Entretanto, isto implica também em um número muito maior de requisições na segunda abordagem, pois para cada nodo, todos os seus vizinho receberão uma requisição. Podemos definir o número total de requisições desta abordagem como o somatório do grau de todos os nodos que formam o caminho de ataque mais um.

Este problema é resolvido na terceira abordagem, que além de considerar o campo TTL no rastreamento, consegue determinar em cada nodo qual de seus vizinhos processou o pacote. Desta maneira, o número de requisições é o menor possível, que é o número de nodos que compõem o grafo de ataque.

Destaca-se o desempenho da proposta deste trabalho: de acordo com a tabela 1 observamos um ganho em termos de número de requisições de no mínimo 4 vezes e no máximo 20 vezes a menos comparado pela abordagem anterior. Comparado com a arquitetura SPIE, observamos uma redução de no mínimo 60% e no máximo 5 vezes, com a desvantagem ainda de o grafo de ataque ser preciso em apenas uma única simulação, onde o número de falsas arestas é zero.

Grafo	Tam. caminho	Parâmetro Waxman α	Parâmetro Waxman β	# Req. (sem TTL)	# Req. (com TTL)	# Req. (nova prop.)	Falsas arestas (SPIE)	Falsas arestas (nova prop.)
0	10	0.808669	0.568594	50	204	10	4	0
1	10	0.231484	0.308439	29	43	10	0	0
2	10	0.175216	0.941675	34	61	10	3	0
3	10	0.324351	1.35804	45	118	4	4	0
4	15	0.907496	0.522733	50	307	15	10	0
5	15	0.751609	1.24419	50	383	15	8	0
6	15	0.453518	0.390714	47	113	3	3	0
7	15	0.590655	1.09217	50	277	6	6	0
8	20	0.106337	1.41285	41	91	4	4	0
9	20	0.350566	1.46795	50	262	8	9	0
10	20	0.987448	1.05393	50	609	14	11	0
11	20	0.150997	1.18203	45	106	25	5	0
12	25	0.677126	1.02934	50	555	25	18	0
13	25	0.123294	0.45648	46	105	25	2	0
14	25	0.92907	0.394996	50	419	14	13	0
15	25	0.787778	1.40516	50	701	19	17	0
16	30	0.273564	0.271755	50	363	16	15	0
17	30	0.433282	0.973342	50	404	15	15	0
18	30	0.78542	1.11016	50	782	25	25	0
19	30	0.723421	1.36563	50	741	30	22	0

Tabela 1. Resultados experimentais.

Em um PRA, a ocorrência de falsos positivos depende de todos os filtros de Bloom que representam o tráfego de um determinado período de tempo. Como cada filtro de Bloom possui uma taxa de falsos positivos p que pode ser calculada a partir do seu fator de capacidade, que é definido por m/n, sendo m o tamanho do vetor de bits e n o número de chaves inseridas, e do número de funções de resumo digital k, podemos definir a probabilidade de ocorrência de pelo menos um falso positivo pelo PRA que possui n vizinhos por $1 - (1 - p)^n$.

A tabela 2 ilustra a taxa real de falsos positivos para um filtro de Bloom com fator de capacidade 13 utilizando 8 funções de resumo digital variando o número de interfaces. Consultando apenas um filtro de Bloom obtemos uma taxa de falsos positivos de 0.199%. Porém, se o nodo possuir 10 interfaces a taxa de falsos positivos sobe para 1.972%, e se possuir 20 interfaces, 3.906%. As duas últimas colunas mostram o total de requisições geradas a partir deste nodo por 100 operações de rastreamento, considerando tanto a abordagem anterior como a nova proposta, mostrando inclusive o impacto dos falsos positivos.

Pode-se concluir que apesar do aumento dos falsos positivos o número de requisições da nova abordagem ainda fica muito melhor que o da abordagem anterior. Perceba que cada falso positivo implica em uma requisição a mais. Desta forma, por exemplo, se o roteador possui 12 interfaces, de um mínimo de 100 consultas, serão geradas aproximadamente 102 requisições posteriores. Na abordagem anterior, seriam geradas 1200 requisições.

# de interfaces	Taxa real de falsos positivos	# requisições abordagem anterior	# requisições nova abordagem considerando falsos positivos
1	0.00199	100	100
2	0.00398	200	100
3	0.00596	300	100
4	0.00794	400	101
5	0.00991	500	101
6	0.01188	600	101
7	0.01385	700	101
8	0.01581	800	102
9	0.01777	900	102
10	0.01972	1000	102
11	0.02167	1100	102
12	0.02362	1200	102
13	0.02556	1300	103
14	0.02750	1400	103
15	0.02944	1500	103
16	0.03137	1600	103
17	0.03330	1700	103
18	0.03522	1800	104
19	0.03714	1900	104
20	0.03906	2000	104

Tabela 2. Análise de falsos positivos para $k = 8$ e $m/n = 13$

5. Conclusão

A eficiência de um sistema de rastreamento de pacotes IP pode ser analisada por diversas métricas. Segundo [8] as métricas seriam a precisão do grafo de ataque obtido em comparação com o grafo de ataque real e o tempo pela qual os registros dos pacotes podem ser armazenados. O trabalho apresentado aumenta a precisão do grafo de ataque pela eliminação das falsas arestas e apresenta um mecanismo mais eficiente para paginação do vetor de bits do filtro de Bloom em memória secundária. Este trabalho apresentou também uma abordagem para se obter um número mínimo de requisições na operação de rastreamento. Nesta arquitetura, é utilizado um filtro de Bloom para cada interface de nodo. Desta maneira, sabe-se exatamente qual vizinho precede o caminho de ataque. Como em uma requisição vários filtros de Bloom devem ser consultados, ocorre um aumento na taxa de falsos positivos como um todo. Como as requisições entre os componentes da arquitetura são feitas apenas usando o resumo digital do pacote, se torna impossível determinar o seu conteúdo. Desta maneira é garantido que a utilização do sistema de rastreamento não diminui a privacidade dos usuário da rede.

Trabalhos futuros incluem o tratamento de transformações de pacotes, a implementação prática desta arquitetura em uma rede real e o projeto de uma implementação em hardware do *Packet Monitor* para roteadores de alta velocidade.

Referências

[1] SecurityStats.com "Security Statistics," `http://www.securitystats.com/`, acessado em 03/2006.

[2] Microsoft Corporation, "Stop 0A in Tcpip.sys When Receiving Out Of Band (OOB) Data," `http://support.microsoft.com/support/kb/articles/Q143/4/78.asp`, acessado em 03/2006.

[3] Savage, S., Wetherall, D., *et all* "Practical Network Support for IP Traceback," *Proceedings of the ACM Special Interest Group on Data Communications 2000 (SIGCOMM'2000)*, pp.295-306, 2000.

[4] Wu, S. F., Zhang, L., *et all* "Intention-Driven ICMP Trace-Back," Internet Draft, IEFT, `draft-wu-itrace-intention-00.txt`, Fev. 2001.

[5] Burch, H. e Cheswick, B. "Tracing Anonymous Packets to Their Approximate Source," *Proceedings of the Fourth Symposium on Operating Systems Design and Implementation (OSDI 2000)*, pp.319,327, San Diego, 2000.

[6] Duffield, N. G. e Grossglauser, M. "Trajectory Sampling for Direct Traffic Observation," *Proceedings of the ACM Special Interest Group on Data Communications 2000 (SIGCOMM'2000)*, pp.271-282, Stockholm, 2000.

[7] Keeni, G. M. "An Architecture for IP Packet Tracing," `http://www.cysol.co.jp/contrib/draft-glenn-ippt-arch-01.txt`, acessado em 03/2006.

[8] Snoeren, A. C., Partridge, C., *et al* "Hash-Based IP Traceback," *Proceedings of the ACM Special Interest Group on Data Communications 2001 (SIGCOMM'2001)*, pp 3-14, 2001.

[9] Bloom, B. H. "Space/Time Trade-Offs in Hash Coding with Allowable Errors," *Communications of the ACM*, Vol.13, pp.422-426, 1970.I

[10] 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*, Vol.8, pp.281-293, 2000.

[11] Sanchez, L. A., Milliken, W. C., *et al* "Hardware Support for a Hash-Based IP Traceback". *Second DARPA Information Survivability Conference and Exposition*, 2001.

[12] Hilgenstieler, Egon e Duarte Jr., Elias P. "Uma Arquitetura para Rastreamento de Pacotes na Internet," IV Workshop em Segurança de Sistemas Computacionais (WSeg'2004), Gramado, RS, 2004.

[13] tcpdump/libpcap "TCPDUMP Public Repository," `http://www.tcpdump.org/`, acessado em 03/2006.

[14] B. M. Waxman "Routing of Multipoint Connections,"*IEEE Journal of Selected Areas in Communications*, pp 1617-1622, 1988.

[15] Baker, F. "Requirements for IP Version 4 Routers,"*RFC 1812*, IETF, Junho 1995.

[16] Simpson, W. "IP in IP Tunneling,"*RFC 1853*, IETF, Outubro 1995.

[17] Kent, S. e Atkinson, R. "Security Architecture for the Internet Protocol,"*RFC 2401*, IETF, Novembro 1998.