EGON HILGENSTIELER

ROTEAMENTO REVERSO NA INTERNET

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientador: Prof. Dr. Elias P. Duarte Jr.

CURITIBA
2004

UFPR Mestrado em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática, do aluno Egon Hilgenstieler, avaliamos o trabalho intitulado, "*ROTEAMENTO REVERSO NA INTERNET* ", cuja defesa foi realizada no dia 04 de junho de 2004, às treze horas, no Auditório do Departamento de Informática do Setor de Ciências Exatas da Universidade Federal do Paraná. Após a avaliação, decidimos pela aprovação do candidato.

Curitiba, 04 de junho de 2004.

Prof. Dr. Elias Procópio Duarte Jr.
DINF/UFPR – Orientador

Prof. Dr. Michael Anthony Stanton
UFF - Membro Externo

Prof. Dr. Roberto A. Hexsel
DINF/UFPR – Membro Interno

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ

Dedico esta dissertação de mestrado aos meus pais, Eric Hilgenstieler (*in memoriam*) e Glória Graciela Ortega de Hilgenstieler.

Agradecimentos

Agradeço a minha família, minha mãe Glória Graciela Ortega de Hilgenstieler e aos meus irmãos Eric Hilgenstieler e Johann Peter Hilgenstieler, pelo apoio incondicional e compreensão em meus momentos de ausência.

Agradeço ao meu orientador Elias Procópio Duarte Jr. pela dedição, incentivo, conselhos e pelo apoio dado em todas as fases deste trabalho.

Aos meus amigos do LARSIS, Andreas, Thiago, Patrícia e Bona, e do mestrado Araújo, Tiaguinho, João Eugênio, Evandro e Gabriel, pelos auxílios e por proporcionarem um ambiente descontraído de trabalho.

Ao Departamento de Informática da UFPR, especialmente ao professor Alexandre I. Direne por proporcionar o suporte institucional e pelo seu grande incentivo. Agradeço à CAPES, pelo suporte financeiro que proporcionou as condições para a realização deste trabalho. Um agradecimento especial a Maria, por tantas horas de trabalho.

SUMÁRIO

Resumo

A arquitetura da Internet não permite que a origem de um pacote IP (*Internet Protocol*) seja descoberta de maneira confiável. Desta maneira, a origem de um ataque a uma rede conectada pode ser facilmente ocultado. Este trabalho descreve uma abordagem para rastreamento de pacotes IP, que permite determinar todo o caminho percorrido por um pacote específico na Internet. Chamamos a este procedimento de *roteamento reverso*. Cada rede se torna responsável por manter um rastro dos pacotes mais recentes que por ali trafegaram. Desta maneira é possível realizar consultas para descobrir o caminho percorrido por um pacote no passado recente. Para permitir que uma grande quantidade de pacotes sejam armazenados e, ao mesmo tempo, garantir a privacidade é calculado um resumo digital (*hash*) de algumas partes do pacote IP, sendo então armazenado apenas o valor deste resumo em uma estrutura de dados eficiente para este propósito chamada *Bloom Filter*. Nesta arquitetura é admitida uma pequena taxa de falsos positivos que pode ser configurada. Ao contrário de arquiteturas propostas anteriormente descobrimos todo o caminho percorrido, e não só a origem do pacote. Em particular eliminamos as chamadas *falsas arestas* comuns no grafo de ataque retornado por outros sistemas. Introduzimos também um novo critério para paginação do *Bloom Filter* que é dependente da carga da rede. Este novo critério é dinâmico e adaptativo, realizando a paginação apenas quando o número de falsos positivos atinge o valor máximo permitido. Este critério permite dimensionar de maneira precisa o intervalo de tempo entre paginações consecutivas. Um protótipo da arquitetura foi implementado e resultados experimentais são apresentados.

Abstract

The architecture of the Internet does not allow an IP (Internet Protocol) packet source to be reliably determined. Thus the source of an attack to a connected network can be easily hidden. This work presents an approach to trace IP packets from the destination, that discovers the route an arbitrary packet has traversed, given only a few fields and the approximate time the packet was received. Each network becomes responsible for maintaining audit trails of the most recent packets it has seen. Through apropriate extraction techniques the complete path a recent packet has traversed can be determined. In order to allow a larger number of packets to be stored and, at the same time, guarantee the user's privacy, a hash function is employed instead of the packet itself. The result is then inserted in a space-efficient structure called Bloom Filter. In the proposed architecture a small rate of false positives is admitted, which can be controlled. In comparison with other proposed architectures, the so called *false edges* are eliminated, in other systems they have not been even defined. Thus our architecture allows the determination of the complete route from destination to source. We also introduce a new approach for paging the Bloom Filter which is dependent on the network load. This new approach is dynamic and adaptive. The Bloom Filter is paged only when the false positive rates reaches the maximum allowed value. This way, it is possible to precisely determine the time interval before the next paging is required. An architecture prototype was implemented and experimental results are presented.

Capítulo 1

Introdução

Atualmente, existem centenas de milhões de pessoas utilizando a Internet [28], muitas destas utilizam a rede para realizar operações críticas como compras e operações bancárias. A segurança é requisito fundamental para sistemas conectados, com grandes quantidades de ataques reportados diariamente [29].

Existem vários mecanismos para detectar a violação de segurança de um sistema [30]. Após detectada uma invasão, um passo importante é a identificação da origem do atacante, ou do sistema que ele utiliza, para tomar as medidas defensivas necessárias para desencorajar tais ataques no futuro. Entretanto, um atacante pode facilmente ocultar a sua origem devido à estrutura do protocolo IP (*Internet Protocol*). Mesmo não havendo a intenção do atacante de ocultar seu endereço, sua real origem pode ser alterada de maneira legítima por algumas transformações que um pacote pode sofrer na rede, por exemplo através de técnicas como NAT (*Network Address Translation*) [17] e *Mobile IP* [16]. Logo, o campo de origem de um pacote IP qualquer não pode ser considerado recurso confiável para autenticação da sua origem. Para resolver este problema, uma solução é o rastreamento de um pacote IP visando descobrir o caminho que este percorreu, até a sua origem. Este trabalho propõe uma estratégia prática de rastreamento na Internet.

Neste capítulo são apresentados conceitos básicos de segurança e os tipos de ataques mais comuns. Em seguida, é visto o funcionamento de um sistema de rastreamento de pacotes IP e uma visão geral da arquitetura proposta.

1.1 Segurança na Internet

A segurança [32] tem o objetivo de evitar que pessoas e processos acessem ou alterem informações e utilizem serviços a que não estão autorizadas ou pratiquem atos de vandalismo, como degradar o desempenho de um serviço da rede ou desabilitá-lo totalmente. É denominado de *ataque* uma tentativa de violar a segurança de um serviço da rede. O host ou rede de destino de um ataque é denominado *vítima*.

Na maioria dos ataques, o endereço real do atacante fica ocultado. Esta técnica é conhecida como *IP Spoofing* [24]. A vítima recebe pacotes contendo um endereço que na realidade é falso, possivelmente pertencente a uma máquina considerada confiável. Isto é feito simplesmente alterando o campo do endereço de origem do cabeçalho IP.

Este tipo de ataque é baseado no fato de que os roteadores, ao redirecionar um pacote, examinam apenas o endereço de destino e geralmente ignoram o endereço de origem. Este endereço é utilizado apenas pelo *host* de destino ao mandar um pacote de volta à origem. Ao ocultar a origem de um pacote, o atacante não pode receber pacotes vindos da rede. Mesmo assim, a técnica de *IP Spoofing* é utilizada como parte de muitos ataques que não necessitam receber respostas da vítima.

Esta seção apresenta os dois principais tipos de ataques que podem ser conduzidos na Internet [22]: os ataques do tipo *Man-in-the-Middle* e de negação de serviço. Em seguida, são exibidas algumas defesas e precauções que podem ser aplicadas contra estes ataques.

1.1.1 Ataques to Tipo *Man-in-the-Middle*

Uma série de variações de ataques podem ser conduzidos usando *IP Spoofing* [24]. Por exemplo, um atacante pode observar o tráfego entre dois *hosts* através de um *sniffer* e enviar pacotes fingindo ser um destes *hosts* (*Man-in-the-Middle*). De alguma maneira deve-se garantir que nenhum pacote chegue até o *host* verdadeiro que tem o endereço forjado. O objetivo deste ataque é explorar uma possível relação de confiança existente entre dois hosts.

É possível também realizar um ataque sem realmente receber nenhuma resposta da

vítima (*Blind spoofing*). Em alguns casos estas respostas não são de interesse, como nos ataques de negação de serviço. Em outros casos [20] é possível enviar requisições apenas prevendo que tipo de resposta a vítima estará produzindo e as próximas requisições esperadas. Ou seja, apesar dos pacotes da vítima não alcançarem o atacante, este pode tentar prever o seu conteúdo e a sua resposta. Isto exige também que o atacante consiga prever qual será o número de sequência deste pacote. Para isto, o atacante conecta em alguma porta disponível na vítima para descobrir o número de sequência inicial (*ISN - Initial Sequence Number*). Este processo é repetido diversas vezes para determinar o *round trip time* (RTT) do host e determinar o padrão de alteração do ISN em função do tempo e das conexões recebidas. Com isto, o atacante pode tentar prever em um dado instante qual será o número de sequência do pacote que a vítima estará esperando.

O verdadeiro *host* detentor do endereço IP a ser utilizado no ataque pode ser desabilitado através, por exemplo, de uma inundação de pacotes SYN [25]. Estes pacotes geralmente contém como endereço de origem *hosts* aleatórios distintos e inexistentes. Desta maneira, nenhum *host* recebe a resposta destes pacotes. Como o protocolo TCP aceita apenas um número limitado de requisições SYN simultâneas, as requisições SYN subsequentes serão descartadas até que todas as conexões pendentes sejam atendidas. Desta maneira, dado um número suficiente de pacotes SYN enviados pelo atacante, requisições legítimas de conexão serão negadas.

A figura 1.1 ilustra um exemplo de ataque usando *IP Spoofing*. O atacante com endereço 1.34.150.37, utiliza a sessão criada entre os hosts 60.166.4.6 e 60.166.47.47 desabilitando o host 60.166.4.6 e enviando pacotes com endereço IP deste host desabilitado.

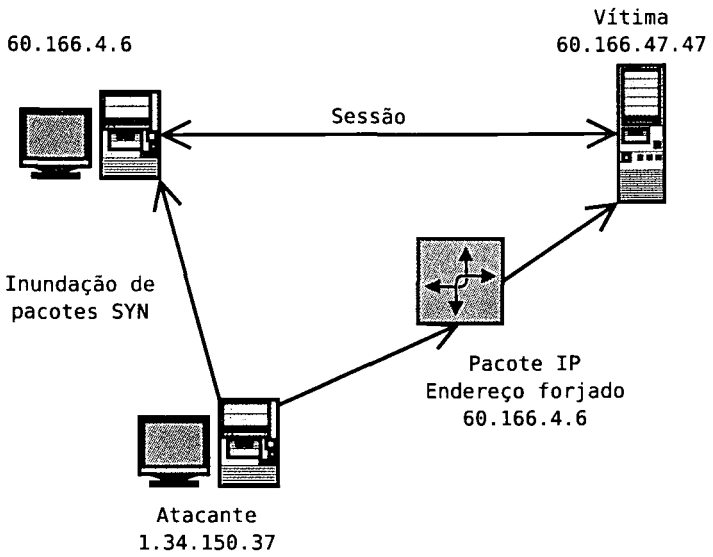

Figura 1.1: Exemplo de um ataque usando *IP Spoofing*.

1.1.2 Ataques de Negação de Serviço

Um atacante pode ter como objetivo simplesmente degradar o desempenho ou desabilitar totalmente um serviço. Este ataque é chamado de *negação de serviço* [26] e possui muitas variações. É possível desabilitar um host enviando um único pacote à vítima. Estes ataques se aproveitam geralmente de falhas nas implementações da pilha de protocolos em alguns sistemas operacionais. Exemplos incluem os ataques *Ping-of-Death*, *Teardrop*, LAND, entre outros [27]. Alguns ataques também podem ser realizados no nível de aplicação, enviando uma determinada combinação de dados não testada pelos desenvolvedores da aplicação. Ataques de negação de serviço também podem explorar a complexidade algorítmica na implementação de alguns tipos de dados utilizados em aplicações [19]. Nestes casos, poucos pacotes são suficientes para completar um ataque.

Um outro tipo de ataque de negação de serviço consiste numa inundação de pacotes direcionados à vítima. Este ataque pode ser realizado através da inundação de pacotes SYN, conforme apresentado anteriormente. Uma série de conexões são abertas através de vários pacotes SYN. Ao esgotar os recursos da vítima, o ataque desabilita o host ou degrada muito o seu desempenho. Uma outra variação deste tipo de ataque é utilizando

uma técnica chamada *Smurf* [23]. Neste caso, o atacante envia um pacote ICMP *echo request* utilizando a técnica de *IP Spoofing* para ocultar o seu real endereço. No lugar do endereço IP do atacante, o pacote ICMP contém como endereço de origem o endereço de sua vítima. O endereço de destino do pacote é um endereço de *broadcast*. Ao alcançar o último hop da rede, o roteador redireciona o pacote para todas as máquinas da rede (se estiver com o *broadcast* habilitado). Estas máquinas, por sua vez, enviam os pacotes ICMP *echo reply* para a vítima do ataque. Com isto, um atacante consegue "amplificar" seu pacote usando uma rede que aceite endereços de broadcast.

Um ataque de negação de serviço também pode ser distribuído, ou seja, várias máquinas podem inundar a vítima com pacotes. Neste caso, um atacante pode invadir várias máquinas existentes na rede, se aproveitando de algumas deficiências de segurança da máquina. O atacante, após subverter estas máquinas com sucesso, instala os programas necessários para que elas recebam requisições pela rede para iniciar um ataque. Após ter um número suficiente de máquinas subvertidas, o atacante envia um comando a todas estas máquinas para iniciar o ataque em uma única vítima. O resultado deste ataque é a sobrecarga nos recursos da vítima e da rede em geral.

1.1.3 Precauções e Defesas

Algumas precauções podem ser tomadas para evitar alguns tipos de ataques. Primeiramente, um host nunca deve basear sua autenticação no endereço IP, pois este pode ser forjado. Deve ser também utilizado um número aleatório para o número de sequência inicial para dificultar a sua previsão.

Um ataque de negação de serviço do tipo *smurf* pode ser evitado filtrando nos roteadores os pacotes cujo destino sejam endereços de broadcast. Isto não impede, entretanto, que um atacante envie uma série de pacotes SYN com o objetivo de desabilitar um serviço.

Como estes ataques utilizam endereços IP forjados, alguns roteadores implementam um técnica chamada de *Ingress Filtering* [18] para tentar solucionar ou amenizar este problema. Estes roteadores redirecionam apenas os pacotes cujos endereços de origem sejam validados. Entretanto, apesar desta técnica dificultar a construção de um ataque,

uma rede iria depender de outras para realizar a filtragem apropriada. Esta técnica também não elimina a possibilidade da utilização de *IP Spoofing* pois um atacante ainda pode utilizar qualquer um das centenas ou milhares de hosts pertencentes às faixas de endereços que não são filtrados pelos roteadores.

Esta descrição de ataques na Internet não é exaustiva e outros tipos de ataques têm surgido na rede. Muitos destes explorando falhas em software e hardware com objetivo de invadir uma máquina, degradar o serviço de uma rede ou desabilitá-lo totalmente [22].

1.2 Rastreamento de Pacotes IP

A infra-estrutura da Internet não permite que a origem de um pacote IP possa ser determinada com precisão. Mesmo que o endereço de origem não seja forjado pelo emissor, algumas técnicas como NAT (*Network Address Translation*) [17] e *Mobile IP* [16] alteram legitimamente o endereço de origem. Uma vez determinado o endereço de origem, determinar o caminho percorrido por um pacote também é um problema de difícil solução. Uma técnica é utilizar a ferramenta *traceroute* [21] que utiliza vários pacotes ICMP (*Internet Control Message Protocol*) variando o valor do campo TTL (*Time to Live*). Entretanto, não há garantia que a rota obtida é a rota percorrida.

O objetivo de um sistema de rastreamento de pacotes IP é, dado um pacote IP qualquer e seu horário de recebimento, determinar não apenas a origem real deste pacote, mas também todo o caminho percorrido entre a sua origem até o seu destino. Como podem haver pacotes duplicados na rede vindos de fontes diferentes, o sistema de rastreamento deve retornar um grafo identificando todas as fontes.

Para atingir este objetivo deve-se armazenar informações suficientes sobre cada pacote IP observado em pontos estratégicos da rede. Como a quantidade de memória disponível em cada ponto é limitada, seriam armazenados apenas os pacotes IP mais recentes. Esta técnica utilizaria quantidades proibitivas de memória em cada ponto de observação mesmo para pequenos períodos de tempo, logo são armazenados resumos digitais [2] de cada pacote. O uso de resumos digitais permite armazenar mais pacotes por um período de tempo maior. Além disso, isto resolve o problema de privacidade que surge com

o armazenamento de pacotes na rede. Dependendo do tipo de roteador em questão, é possível recuperar os pacotes processados através de um *sniffer* presente no mesmo segmento de rede do roteador ou através de um agente conectado através de alguma interface auxiliar do próprio roteador.

Para realizar um rastreamento de um dado pacote deve-se tentar descobrir o caminho inverso percorrido pelo pacote começando pelo último roteador pelo qual o pacote passou e consultando os roteadores precedentes para descobrir se o dado pacote foi ou não processado naquele roteador. Deve-se observar que, em geral, não é o próprio roteador que implementa o rastreamento. Este procedimento é repetido para os pontos seguintes até encontrar a origem ou até chegar ao fim da rede capaz de realizar rastreamentos.

A arquitetura SPIE [9] permite o rastreamento de pacotes IP individuais na Internet. São armazenados apenas os resumos digitais da parte invariante de cada pacote IP, de maneira que ele possa ser identificado em qualquer nodo da rede pelo qual tenha passado. Esta arquitetura propõe o uso de *Bloom Filters* para armazenamentos dos resumos digitais que representam os logs de tráfego. A partir dos registros dos pacotes mantidos por cada nodo de uma determinada região da rede, é executado o algoritmo *Reverse Path Flooding*. Para um determinado nodo que tenha visto o pacote, os registros de seus vizinhos também são consultados e assim por diante até que nenhum outro nodo tenha visto o pacote. Uma lista dos nodos já visitados é mantida para garantir o término do algoritmo.

1.3 Uma Arquitetura para Roteamento Reverso na Internet

Este trabalho apresenta uma arquitetura para rastreamento de pacotes IP que, ao contrário de arquiteturas já propostas, permite determinar todo o caminho percorrido por um pacote específico. Chamamos a este procedimento de *roteamento reverso*. Nesta arquitetura cada nodo da rede é responsável por manter um log do tráfego da rede. Para tanto, é calculado um resumo digital de alguns campos do pacote IP que são então armazenados em um *Bloom Filter*. Uma estratégia para paginação do *Bloom Filter* de memória primária para secundária também é contribuição deste trabalho. Através de técnicas de extração apropriadas é possível recuperar o grafo de ataque de um pacote específico recebido por

uma vítima. É permitida uma pequena taxa de falsos positivos, ou seja, podem aparecer nodos no grafo de ataque que não observaram realmente o pacote. Esta taxa pode ser configurada através de parâmetros do sistema.

A proposta elimina as chamadas *falsas arestas*, que aparecem, por exemplo, na arquitetura SPIE. Se existir uma ou mais arestas que conectem dois nodos do grafo de ataque podem ser incluídas arestas que não pertencem ao grafo de ataque original, como ilustra a figura 1.2. As flechas em (a) representam o caminho do ataque. Linhas contínuas representam o enlace entre os nodos. Em (b) é mostrado o grafo de ataque que seria retornado. A explicação segue abaixo.

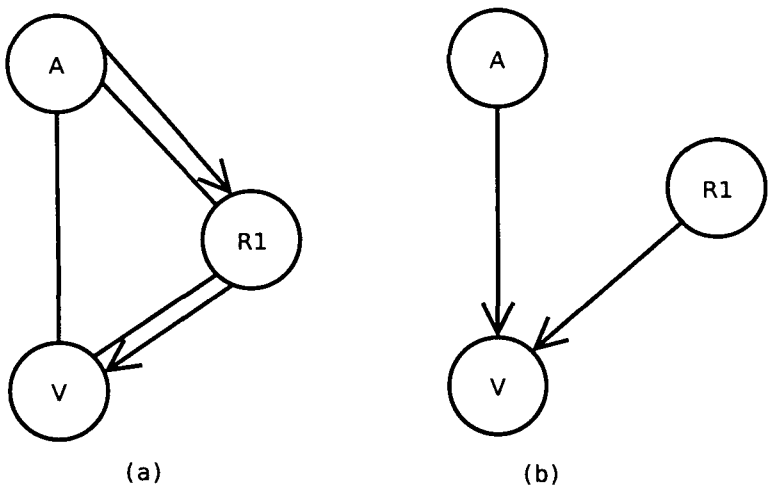

Figura 1.2: Exemplo de um grafo de ataque real e o obtido pelo sistema SPIE.

Apesar do link entre o nodo V e o nodo A não ter sido utilizado pelo pacote, ele está contido no grafo de ataque por que ambos os nodos observaram o mesmo pacote, entretanto, o caminho real percorrido, na verdade, passa por R1. Por outro lado, a aresta que conecta R1 e A não pertence ao grafo de ataque porque R1 já foi visitado após a visita a A, logo, A foi acrescentado na lista de nós já visitados, não sendo mais consultado depois de R1. Este grafo além de impreciso é mais difícil de ser interpretado. Pode ser entendido que houveram duas fontes distintas para o mesmo pacote, ou que talvez um dos nodos seja na verdade um falso positivo. Nossa arquitetura resolve este problema descobrindo o caminho exato até a origem.

Um sistema de rastreamento se torna mais útil se largamente difundido na Internet. Entretanto, podem surgir problemas de cooperação técnicas ou políticas entre diferentes entidades administrativas. Deve-se contudo destacar que um sistema de rastreamento pode ser útil também se implantado em um único sistema autônomo, possibilitando o rastreamento de pacotes internos.

1.4 Organização deste Trabalho

O restante deste trabalho está organizado da seguinte forma. O capítulo 2 traz a descrição de trabalhos relacionados. O capítulo 3 apresenta os detalhes da arquitetura proposta. A implementação e resultados experimentais são apresentados no capítulo 4. As conclusões seguem no capítulo 5.

Capítulo 2

Arquiteturas para Rastreamento de Pacotes IP

A Internet é atualmente vulnerável a atacantes suficientemente equipados [29]. Diversas ferramentas encontram-se disponíveis para atacar serviços vitais de uma rede [39]. O objetivo de um ataque pode ser simplesmente degradar o desempenho dos sistemas atacados ou desabilitar totalmente o serviço, explorando falhas em software ou hardware. Enquanto os ataques de negação de serviço conduzidos através de uma grande quantidade de tráfego são os mais comuns, existem também ataques que podem desabilitar um serviço utilizando apenas um único pacote [27].

Após identificado um ataque, um procedimento que pode ser necessário é o rastreamento dos pacotes IP que fizeram parte do ataque para que se possa determinar a sua origem. Entretanto, devido à arquitetura do protocolo IP, o rastreamento de pacotes é um problema de difícil solução. A estrutura de roteamento não mantém informações de estado, é baseada no endereço de destino e nenhuma entidade é oficialmente responsável por verificar a autenticidade do endereço de origem de um pacote IP. Portanto, um atacante pode ocultar a real origem de seus pacotes fazendo inclusive com que eles aparentem ter sido originados de *hosts* aleatórios. Além disso, a alteração do endereço de origem pode ser inclusive realizada de maneira legítima por técnicas como NAT (*Network Address Translation*) [17].

O restante deste capítulo está organizado como segue. A seção 2.1 resume a funcionalidade e os objetivos de um sistema de rastreamento de pacotes IP. A seção 2.2 detalha os requisitos deste sistema. A seção 2.3 apresenta um breve histórico de trabalhos relacionados. As seções 2.4 e 2.5 apresentam duas arquiteturas para rastreamento.

2.1 Funcionalidade

O objetivo de um sistema de rastreamento é, dado um pacote IP, o horário aproximado do recebimento deste pacote e o seu destino (chamado de *vítima*) construir um *caminho de ataque*. Este caminho consiste de cada roteador por onde o pacote passou no seu caminho até a vítima. Como podem haver múltiplas fontes para um mesmo pacote, é construído e retornado um *grafo de ataque* [9].

Um sistema de rastreamento deve possuir monitores em pontos estratégicos da rede para poder identificar os pacotes que por ali passaram, utilizando algum método de armazenamento. Ao receber uma requisição de rastreamento, o sistema deve verificar em cada monitor se o pacote passou por sua região. Em caso de resposta positiva, deve-se consultar o conteúdo dos monitores "próximos" ou "vizinhos" na rede. Este processo continua até que o sistema consiga identificar uma fonte para o pacote ou até esgotar a capacidade de realizar rastreamentos.

2.2 Requisitos de um Sistema de Rastreamento

É importante definir o conceito de rastreamento de pacotes IP, esclarecendo os requisitos necessários e as asserções que devem ser feitas sobre a rede e o seu tráfego. É importante ter uma definição clara do contexto em que um sistema de rastreamento deve ser projetado, para poder avaliar diferentes técnicas de rastreamento.

Primeiramente, são definidas asserções sobre o ambiente no qual um sistema de rastreamento opera. A primeira asserção diz respeito ao fato de os pacotes IP poderem ser endereçados a mais de uma máquina, pois podem conter como seu endereço de destino um endereço de *broadcast* ou *multicast* causando a duplicação do pacote internamente

na rede. Pacotes duplicados também podem existir na rede [31]. O próprio atacante pode injetar pacotes idênticos, possivelmente a partir de diferentes máquinas. Um sistema de rastreamento deve estar preparado para a situação onde houver várias fontes de um mesmo pacote ou uma única fonte de múltiplos (também possivelmente idênticos) pacotes.

Apesar de pouco provável, um atacante pode ter obtido o controle de roteadores ao longo do caminho até a vítima, desta forma o sistema de rastreamento não pode confiar em todos os roteadores *a priori*. Além disso, um atacante geralmente está ciente das características da rede, inclusive de sua capacidade de rastrear ataques. Um sistema de rastreamento não pode ser ludibriado por roteadores subvertidos por um atacante.

As implicações da instabilidade da arquitetura de roteamento da Internet são importantes no projeto de um sistema de rastreamento. Dois pacotes enviados pelo mesmo host com o mesmo destino podem percorrer caminhos completamente diferentes. Logo, qualquer sistema que tem por objetivo analisar a origem de um fluxo baseado em múltiplos pacotes deve estar preparado para lidar com informações divergentes do caminho percorrido por estes.

Um sistema eficiente de rastreamento não deve aumentar o tamanho dos pacotes IP. Vários protocolos utilizados atualmente causam o aumento do tamanho dos pacotes, como por exemplo tecnologias baseadas em tunelamento IP, como IPsec (*IP security*) e *mobile IP* [16]. Entretanto, aumentar o tamanho do pacote pode causar problemas tendo em vista o MTU (*Maximum Transfer Unit*) das redes pelas quais o pacote passa, além de aumentar a sobrecarga do protocolo.

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 muitos dispositivos com funções específicas como microscópios, câmeras e impressoras estão conectados à Internet mas raramente têm recursos além daqueles necessários para sua função. Logo, um sistema de rastreamento deve assumir que os *hosts* finais são pobres em recursos.

Os pacotes podem ser modificados durante o processo de roteamento. As transformações de pacotes podem ser resultantes de um processamento válido, de erro do

roteador ou por intenções maliciosas do atacante. Estes dois últimos casos não precisam ser considerados pelo sistema de rastreamento, pois nestes casos os pacotes deverão ser rastreados apenas até o seu ponto de transformação, uma vez que o roteador está falho ou pode ser considerado como um atacante. Entretanto, um sistema de rastreamento ideal deve ser capaz de rastrear pacotes que sofrem transformações *válidas* até a sua origem.

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 [14] e *IP-in-IP* [13]. Muitas destas transformações resultam em perdas irrecuperáveis do estado do pacote original.

Um estudo descrito em [37] indica que menos de 3% do tráfego existente na Internet passa por transformações ou por tunelamento. Entretanto, atacantes podem transmitir pacotes de maneira a tentar garantir que estes sofram transformações. Logo, um sistema de rastreamento deve ser capaz de lidar com estas transformações.

A asserção final é que a operação de rastreamento não é frequente. Isto tem implicações no projeto de um sistema de rastreamento: um roteador não precisa processar requisições de rastreamento em tempo real, na velocidade do enlace.

Idealmente, um sistema de rastreamento deve ser capaz de determinar a origem de qualquer fragmento de dados enviado através da rede. No protocolo IP, um pacote é a menor unidade de dados. Qualquer subdivisão estará contida em um único pacote. Qualquer unidade de dados maior (por exemplo um fluxo) pode ser rastreado através de qualquer um dos seus pacotes. Logo, um sistema de rastreamento perfeito deve ser capaz de identificar a origem de qualquer pacote IP arbitrário.

Como qualquer sistema auditor, o sistema de rastreamento só será efetivo nas redes em que ele está instalado. Logo, a fonte de um pacote será sempre um nodo pertencente à porção da rede capaz de realizar rastreamento de pacotes. Este nodo pode ser o próprio host de origem, um roteador subvertido dentro da rede em que o sistema de rastreamento atua ou o ponto mais próximo da origem capaz de realizar rastreamentos.

Se for considerada a possibilidade de roteadores poderem ser controlados por um

atacante, será necessário não apenas conhecer a real origem de um pacote mas sim todo o caminho percorrido por este. Se um caminho pode ser rastreado através de roteadores não subvertidos, ele deve terminar na origem do pacote. Caso contrário, se o pacote passar por um roteador subvertido ele deverá ser tratado apropriadamente. Logo, estamos interessados em construir um *caminho de ataque*, onde este caminho consiste de cada roteador por onde o pacote passou em seu caminho até a vítima. Múltiplas fontes podem ser identificadas se algum roteador for subvertido. Além disso, como múltiplos pacotes idênticos podem ser gerados de diferentes fontes, um sistema de rastreamento deve ser capaz de gerar um *grafo de ataque* composto dos caminhos de ataque de cada instância do pacote que chegou até a vítima. A figura 2.1 mostra um exemplo de um grafo de ataque contendo caminhos de ataque para dois pacotes idênticos gerados por A1 e A2 e recebidos pela vítima V. As flechas indicam os enlaces percorridos pelo pacote. Nodos pertencentes a um caminho de ataque estão destacados.

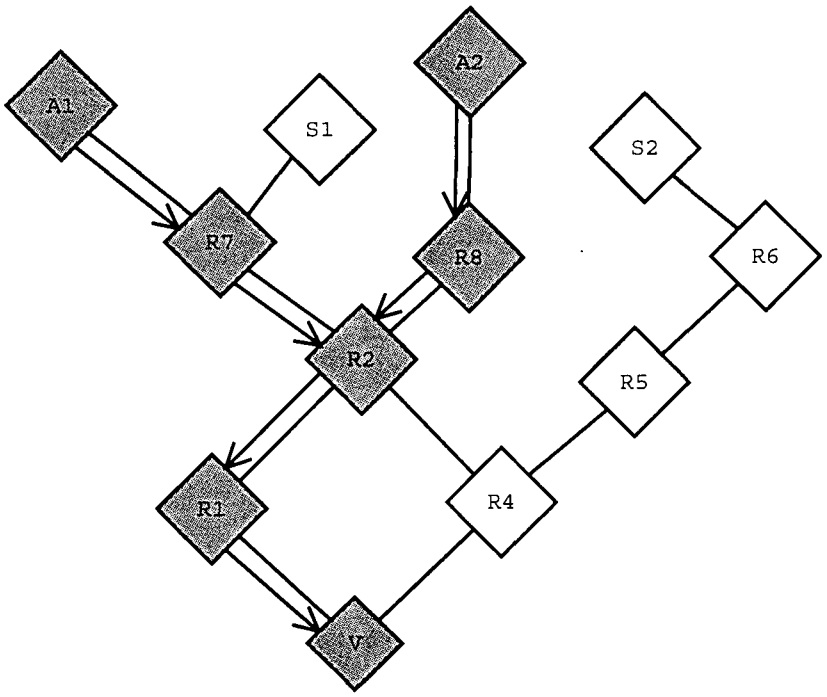

Figura 2.1: Exemplo de grafo de ataque.

É inevitável que um grafo de ataque possa conter falsos positivos, que ocorrem no caso de haverem roteadores subvertidos. Ou seja, um grafo de ataque pode conter fontes que

não emitiram realmente o pacote. Entretanto, um sistema de rastreamento ideal não deve gerar falsos negativos ao tentar reduzir o número de falsos positivos.

Além disso, quando um sistema de rastreamento é instalado em uma rede, ele não deve reduzir a privacidade das comunicações IP. Em particular, entidades envolvidas na geração, retransmissão ou recebimento do pacote original não devem ser capazes de ganhar acesso ao conteúdo do pacote nem utilizando, e nem participando, do sistema de rastreamento.

2.3 Histórico

Várias soluções foram propostas para determinar a rota de um *fluxo* de pacotes, em contraposição ao rastreamento de pacotes individuais. Isto pode ser feito basicamente de duas maneiras: observando o fluxo enquanto ele atravessa a rede ou tentando inferir a rota baseada no impacto do fluxo no estado da rede [3].

O primeiro trabalho proposto para inferir uma rota [3] considerou o problema de grandes fluxos de pacotes. Algumas redes candidatas eram sistematicamente inundadas com pacotes. Desta maneira é possível observar variações no fluxo de pacotes recebidos e é possível inferir a rota percorrida pelo fluxo. Para tanto é necessário ter certo conhecimento da topologia da rede e ter a capacidade de gerar grandes fluxos de dados para um enlace de uma rede arbitrária.

As técnicas utilizadas para observar o fluxo de pacotes podem ser classificadas de acordo com a maneira com que os recursos necessários ao rastreamento são distribuídos entre os componentes da rede. Recursos adicionais podem ser necessários nas máquinas finais, na própria rede ou em ambos [8].

Quando as informações referentes ao fluxo são armazenadas nas máquinas finais, os roteadores notificam o destino do pacote de sua presença na rota. Vários esquemas são utilizados para reduzir o tamanho necessário para enviar tais informações. Entretanto, não existe nenhuma técnica que possibilite a auditoria de cada pacote, devido à grande sobrecarga envolvida. Apenas métodos probabilísticos foram apresentados que permitem que um número suficientemente grande de pacotes seja rastreado. Apesar de não man-

terem informações, os roteadores fornecem informações parciais em um subconjunto de pacotes de um fluxo, de forma a permitir que a máquina final reconstrua todo o caminho percorrido pelos pacotes depois de receber uma determinada quantidade de pacotes pertencente àquele fluxo.

Dois esquemas foram propostos para comunicar as informações referentes ao caminho percorrido pelo fluxo para as máquinas finais. Em [8] as informações são codificadas em campos raramente utilizados no cabeçalho do pacote IP. Já em [10] as informações de auditoria são enviadas através de mensagens ICMP (*Internet Control Message Protocol*).

Uma outra alternativa é deixar a própria rede com a responsabilidade de manter os estados necessários para informações de auditoria. Uma maneira de cumprir esta tarefa é simplesmente armazenando um *log* de cada pacote em vários pontos da rede para então utilizar técnicas de extração apropriadas para descobrir o caminho percorrido pelo pacote na rede. Esta técnica foi inicialmente proposta em [35]. Entretanto, a eficiência da utilização deste *log* depende do espaço disponível para armazená-lo. Tendo em vista os enlaces usados atualmente, os *logs* de pacotes iriam rapidamente crescer até atingir tamanhos impraticáveis mesmo mantendo apenas as informações referentes a um pequeno espaço de tempo.

Este problema pode ser reduzido utilizando técnicas de amostragem, mas isto diminui a probabilidade de detectar fluxos pequenos, além disto, surge um problema adicional: a segurança da armazenagem de pacotes completos nos roteadores.

Estas técnicas propostas apresentam soluções apenas para grandes fluxos de pacotes, sendo úteis apenas para avaliar ataques do tipo negação de serviço. Entretanto, como foi apresentado anteriormente, em alguns casos um ataque pode consistir de um único pacote.

2.4 A Arquitetura SPIE

Em [9] foi proposta uma arquitetura mais completa com o objetivo de permitir o rastreamento de pacotes IP individuais na Internet. Esta técnica utiliza a própria rede para armazenar os estados necessários para o rastreamento, também utilizando *logging* para

determinar quais pacotes passaram por cada ponto da rede. No sistema descrito, chamado de SPIE (*Source Path Isolation Engine*), o problema resultante do tamanho do *log* de pacotes é resolvido utilizando-se do armazenamento de *resumos digitais* de alguns campos de um pacote IP, que o identificam univocamente. Esta estratégia também resolve o problema de privacidade que seria decorrente do armazenamento de pacotes completos em componentes da rede. As informações referentes a um pacote são mantidas por um determinado período de tempo e depois sobrescritas para guardar informações sobre pacotes mais recentes. O tamanho deste intervalo de tempo depende dos recursos dedicados ao rastreamento. Dado este sistema de armazenamento, técnicas apropriadas de extração são utilizadas para descobrir o caminho percorrido por um determinado pacote.

2.4.1 Uso de Resumos Digitais na Arquitetura SPIE

Uma função de resumo digital ou *hash* é uma função $H(M)$ que recebe como parâmetro uma mensagem M de tamanho variável e produz uma saída de tamanho fixo R, chamado de *resumo* [2]. Ao contrário dos algoritmos de criptografia de chave pública e de chave secreta esta função não é reversível e não é utilizada uma chave como parâmetro. O resumo é utilizado para representar uma determinada sequência de bits. Entre seus usos estão o teste de integridade de dados e o armazenamento destes resumos para efetuar testes de comparação. As senhas dos usuários de um sistema, por exemplo, podem ser armazenadas como resumos. A validação de uma senha seria então feita calculando o seu resumo digital e realizando um teste de comparação com o resumo armazenado.

Uma função de resumo digital H deve ter as seguintes propriedades:

- H pode ser aplicado em um bloco de dados de qualquer tamanho;
- H deve produzir um resumo de tamanho fixo;
- Deve ser computacionalmente fácil calcular $H(M)$ para uma dada mensagem M, tornando práticas a sua implementação em hardware e software;
- Deve ser computacionalmente impraticável calcular M dado $H(M)$;

- Deve ser impraticável calcular uma mensagem M' tal que $H(M') = H(M)$ para qualquer M;

- Deve ser impraticável calcular um par (M, M') onde $M \neq M'$ tal que $H(M) = H(M')$;

Os algoritmos MD5 (*Message Digest 5*) [11] e SHA (*Secure Hash Algorithm*) [12] são exemplos de algoritmos para cálculo de resumo digital.

Na arquitetura SPIE, a auditoria de tráfego é realizada através do armazenamento de resumos dos pacotes ao invés de armazenar os próprios pacotes. Isto reduz os requisitos de armazenamento ao mesmo tempo que preserva a confidencialidade.

O conteúdo do pacote utilizado como entrada para a função hash deve representar de maneira única um pacote IP, permitindo sua identificação em qualquer ponto do caminho percorrido por este pacote. Ao mesmo tempo, uma entrada deve ter o menor tamanho possível, por questões de desempenho. Vale a pena ressaltar que alguns bytes referentes ao cabeçalho do pacote IP podem sofrer alterações, e deve ser possível reconstruir o cabeçalho original, como será visto mais a frente.

A figura 2.2 mostra um pacote IP e os campos incluídos para o cálculo da função hash. O sistema SPIE calcula o hash apenas sobre a porção invariante do cabeçalho IP mais os 8 primeiros bytes do conteúdo. Campos que são frequentemente alterados são mascarados antes do resumo do pacote ser calculado.

Os resultados apresentados em [9] 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.

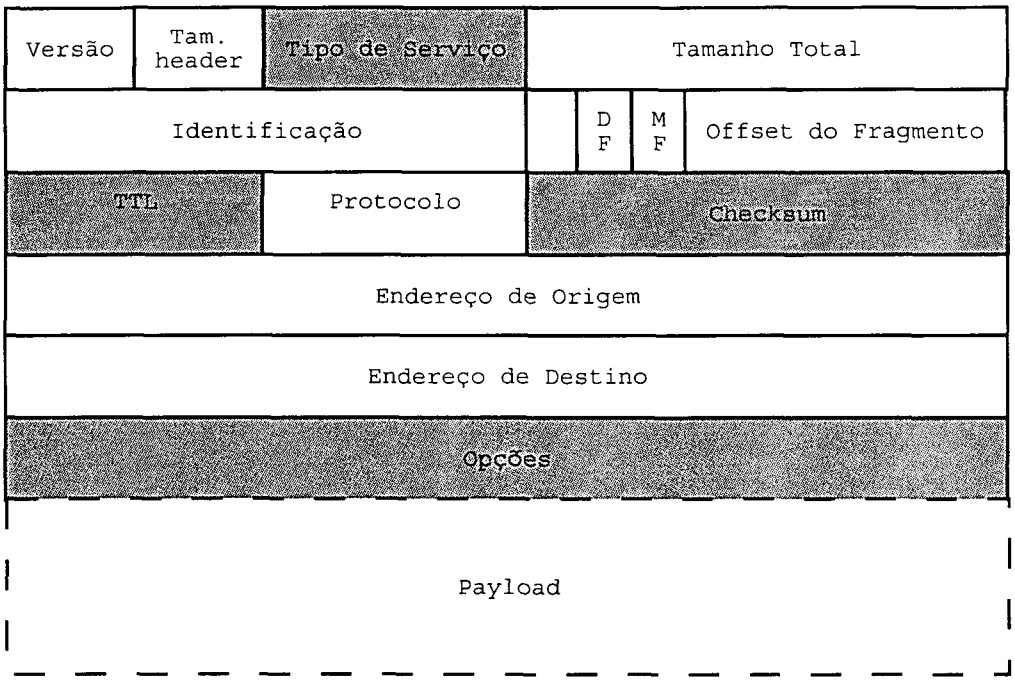

Figura 2.2: Os campos de um pacote IP. Campos em cinza são mascarados antes de ser aplicada a função hash.

O armazenamento de todos os resumos gerados pelo tráfego que passa pelo roteador requer quantidades muito grandes de armazenamento. Para resolver este problema, o sistema SPIE utiliza um estrutura de dados chamada *Bloom Filter* [1] para armazenar os resumos de cada pacote. Um *Bloom Filter* é utilizado para representar um determinado conjunto de elementos, possibilitando a operação de consulta da existência de qualquer um destes elementos.

Um *Bloom Filter* 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 2.3 ilustra a utilização de um *Bloom Filter* usando k funções hash.

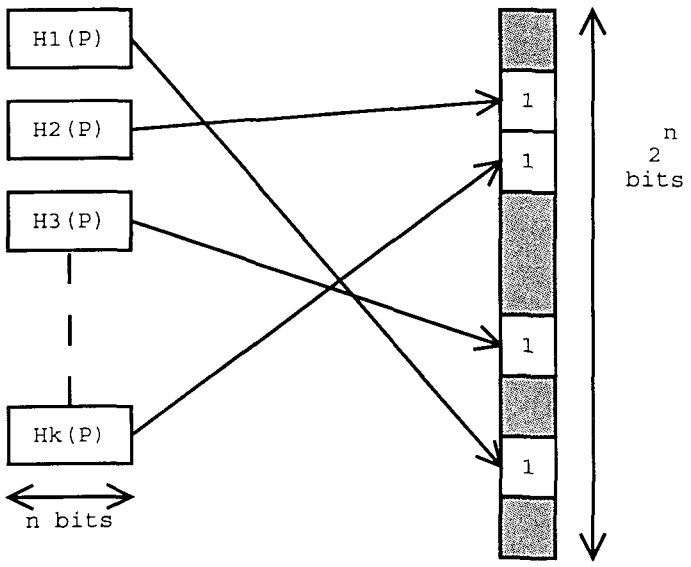

Figura 2.3: Utilização de um *Bloom Filter*.

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.

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. Pode ser que outras inserções causaram estes bits a serem setados, criando assim um *falso positivo*. A taxa de ocorrência destes falsos positivos pode ser controlada [33], ajustando os parâmetros n e k.

2.4.2 Componentes Principais da Arquitetura

As várias tarefas necessárias para o rastreamento de um pacote IP são separadas em diferentes componentes no sistema SPIE. Os três principais componentes estão ilustrados na figura 2.4 e são descritos a seguir:

- DGA (*Data Generation Agent*) - Este componente calcula o valor do hash dos pacotes que deixam o roteador e o armazena em tabelas. Estas tabelas representam o

tráfego que passou pelo roteador em um determinado intervalo de tempo e são armazenadas localmente no DGA por determinado período dependendo das limitações de recursos da máquina. Dependendo da máquina, este componente pode ser implementado por software ou como um hardware separado conectado ao roteador por alguma interface auxiliar [7].

- SCAR (*SPIE Collection and Reduction Agent*) - Um SCAR é responsável por vários DGAs. Caso seja expresso interesse em um pacote específico, o SCAR receberá uma requisição do STM, descrito abaixo. O SCAR requisita então as tabelas de todos os DGAs de sua região. Estas tabelas uma vez transferidas para o SCAR, são analisadas. Caso o pacote seja encontrado nestas tabelas é retornado ao STM um grafo de ataque correspondente à sua região. Um roteador nunca recebe uma requisição diretamente pois as tabelas do DGA são todas transferidas ao SCAR para só então ser verificado se o pacote foi visto ou não.

- STM (*SPIE Traceback Manager*) - Este componente gerencia todo o sistema e é a interface para uma entidade que pode requisitar o rastreamento de um pacote. Quando uma requisição é recebida, sua autenticidade é verificada e a requisição é despachada para os SCARs apropriados. O STM então recebe os grafos de ataques resultantes e com base nestes monta um único grafo de ataque completo. Ao término deste processo, o grafo de ataque final é retornado ao cliente.

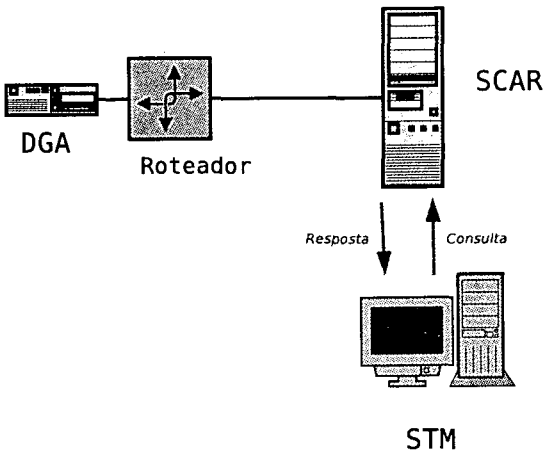

Figura 2.4: A arquitetura SPIE.

2.4.3 Processamento do Rastreamento

Antes que o processo de rastreamento possa começar, um pacote de um ataque deve ser identificado. O usuário do rastreamento pode ser, por exemplo, um sistema de detecção de intrusão que determina a ocorrência de um evento excepcional, isto é, de um ataque e fornece ao STM as seguintes informações: um pacote do ataque, o identificador da máquina alvo deste ataque (denominada *vítima*) e o horário do ataque. Dois requisitos são impostos: o primeiro é que a vítima deve ser expressa como o último *hop* e não como a máquina final em si; o segundo é que o pacote de ataque deve ser recente. O primeiro requisito fornece ao sistema um ponto de início e o segundo vem do fato que o rastreamento deve ser realizado antes que as respectivas tabelas sejam sobrescritas pelo DGA. Esta restrição referente ao tempo está diretamente relacionada aos recursos dedicados ao armazenamento de dados referentes ao tráfego.

Ao receber uma requisição, o STM verifica sua autenticidade e integridade. Após uma verificação positiva, o STM imediatamente envia uma requisição a todos os SCARS em seu domínio. Estes, por sua vez, recuperaram a partir dos seus respectivos DGAs as tabelas de tráfego relevantes para análise.

O tempo é crítico nesta etapa pois esta consulta deve acontecer enquanto os dados ainda estão residentes nos DGAs. Uma vez que as tabelas são transferidas com sucesso

para os SCARs, o processo de rastreamento não estará mais sob as mesmas restrições de tempo real.

A consulta realizada pelo STM começa no SCAR responsável pela região da rede da vítima. O SCAR, por sua vez, responde com um grafo de ataque parcial, além de informações sobre como se encontrava o pacote ao entrar na região (ele pode ter sido transformado múltiplas vezes dentro desta região). O grafo de ataque pode terminar nesta região gerenciada pelo SCAR caso uma fonte tenha sido identificada dentro da região, ou pode conter nós na borda da região da rede. Neste caso, o STM manda uma consulta para o SCAR que gerencia a região que contém aquele nodo. Este processo continua até que não haja possibilidade de expandir o grafo de ataque, com uma fonte dentro da rede ou com o fim da região gerenciada pelo sistema SPIE. O STM por fim constrói um grafo de ataque final com base nos grafos parciais recebidos e o retorna ao sistema cliente.

2.4.4 Processamento de Transformações

Pacotes IP podem passar por transformações válidas durante sua travessia pela rede. O sistema SPIE deve ser capaz de rastrear um pacote mesmo se este sofrer transformações. Em particular, deve ser capaz de reconstruir o pacote original a partir do pacote transformado. Entretanto, alguns tipos de transformações ocasionam perda de informações do pacote. Sem estas informações uma transformação não pode ser revertida. Desta maneira, informações suficientes devem ser armazenadas pelo sistema SPIE para poder reconstruir os pacotes originais.

Antes de computar o valor da função hash, o sistema SPIE mascara campos do cabeçalho IP que são frequentemente alterados. Esta operação esconde a maioria das transformações ocorridas em cada *hop*, mas obriga o sistema SPIE a lidar explicitamente com as seguintes transformações: fragmentação, NAT (*Network Address Translation*), mensagens ICMP (*Internet Control Message Protocol*), tunelamento IP em IP [13] e IPsec (*IP Security*) [14].

Para gravar as informações necessárias para reconstruir o pacote original serão necessários

recursos adicionais. Em conjunto com as tabelas armazenadas pelo DGA, o sistema 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, descritos a seguir. 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 descritas acima). O último campo contém informações de tamanho variável do pacote. O tamanho utilizado depende da transformação utilizada. A figura 2.5 ilustra uma entrada da TLT.

Para que esta tabela possa ser armazenada de maneira eficiente, este último campo é limitado a 32 bits. Algumas transformações, como NAT, podem requerer mais espaço. Estas informações são guardadas de maneira indireta. Um bit da tabela é reservado como bit de indireção (indicado como I na figura). Se este bit estiver setado, o último campo da TLT será tratado como um ponteiro para uma estrutura de dados externa que contém as informações necessárias para a reconstrução do pacote.

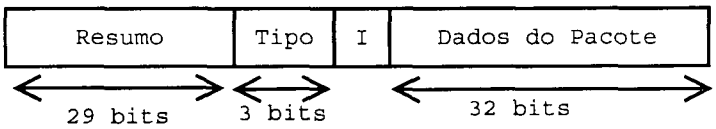

Figura 2.5: Uma entrada da TLT. A TLT armazena informações suficientes para recuperar um pacote transformado.

2.4.5 Construção do Grafo

Cada SCAR constrói um subgrafo usando as informações da topologia sobre uma região particular da rede. Após coletar as tabelas de todos os roteadores pertencentes à sua região, o SCAR simula uma inundação do caminho reverso (*Reverse Path Flooding*) examinando as tabelas já armazenadas localmente. É importante notar que não é enviada nenhuma requisição aos roteadores pois todas as tabelas contendo as informações de tráfego foram previamente armazenadas no SCAR.

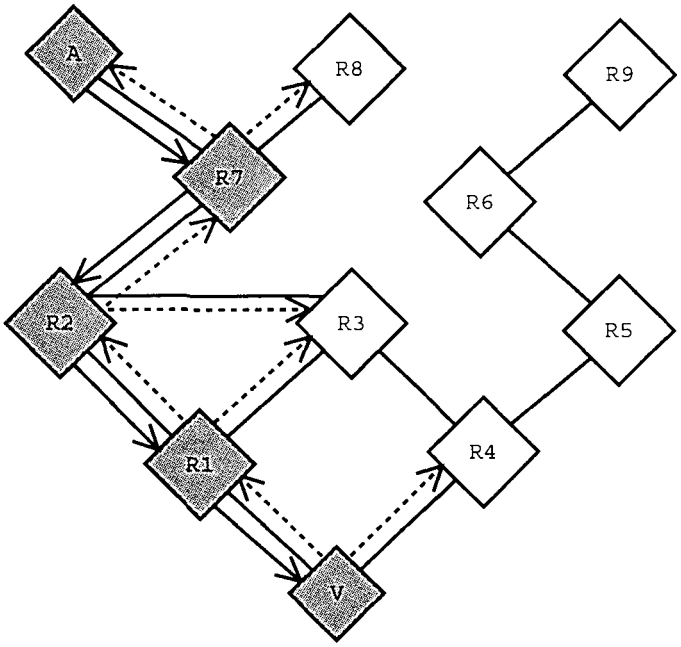

Figura 2.6: Exemplo de um Grafo de Ataque. As flechas contínuas representam o caminho do ataque. Flechas tracejadas representam requisições SPIE.

Para consultar um roteador, o SCAR calcula o valor da função hash para o determinado pacote e, em seguida, consulta a tabela para verificar se existe uma entrada para este pacote. Caso positivo, é considerado que o pacote passou por este roteador. Este nodo é adicionado então ao grafo de ataque e continua para cada um de seus vizinhos, com exceção daqueles já visitados. Entretanto, se não existe nenhuma entrada para este pacote na tabela, pode ser necessário realizar uma consulta em um período de tempo posterior. Dependendo da latência da comunicação entre os roteadores, os SCARs podem precisar requisitar múltiplas tabelas de cada roteador. Uma vez que o valor do hash é encontrado, o tempo de chegada do pacote é sempre considerado o mais tarde possível, dado o intervalo. Isto garante que o pacote deve ter sido visto anteriormente por roteadores adjacentes.

A figura 2.6 mostra um exemplo de execução do algoritmo para uma vítima V e um atacante A. O caminho percorrido pelo pacote foi A, R7, R2, R1, V.

Se o pacote não é encontrado em nenhuma das tabelas dentro do período de tempo relevante, a busca naquele determinado ramo da árvore de busca é terminada e a busca continua nos roteadores remanescentes. Uma lista de todos os nodos visitados é mantida

para evitar ciclos na busca e garantir sua finalização.

O resultado deste procedimento é um grafo conexo contendo o conjunto de nodos que acredita-se terem redirecionado o pacote até a vítima. Assumindo a correta operação dos roteadores, garante-se que este grafo contém os nodos do grafo de ataque real. Devido às colisões na função hash, entretanto, podem haver nodos no grafo de ataque construído que não estão no grafo de ataque real. Estes nodos são chamados de *falsos positivos*.

2.5 Uma Arquitetura Alternativa para Rastreamento de Pacotes IP

Em [34] foi proposta uma arquitetura simples para o rastreamento de pacotes IP. Esta arquitetura também utiliza a própria rede para armazenar os estados necessários para o rastreamento. O seu funcionamento baseia-se na existência de monitores em vários pontos de observação da rede. Cada monitor mantém um registro sobre cada pacote IP observado. Um agente é então capaz de determinar se um determinado pacote foi ou não visto em um ponto sendo monitorado. O caminho descoberto pode ter granularidade variável (em termos de *hops* da rede) dependendo do número de monitores existentes entre a origem e o destino do pacote IP.

A arquitetura para rastreamento de pacotes IP é baseada em 4 componentes: *Packet Monitors*, *Packet Record Base* (PRB), *Packet Record Agent* (PRA), e o *Packet Tracer Application* (PTA).

Um *Packet Monitor* cria um registro para cada pacote IP observado em um determinado ponto da rede. Um monitor pode fazer parte de um roteador ou pode ser um dispositivo passivo que observa todo o tráfego de um determinado ponto da rede.

Os registros gerados pelo *Packet Monitor* são armazenados no *Packet Record Base*. O conteúdo e o formato do registro armazenado no PRB devem ser suficientes para identificar se um pacote foi visto ou não. O registro armazenado pode ser uma transformação do pacote, como por exemplo um resumo digital do pacote inteiro ou de alguns dos seus campos.

Um *Packet Record Agent* consulta a PRB para verificar se um determinado pacote foi ou não visto pelo monitor que alimenta esta PRB. O PRA recebe uma chave de consulta do PTA e aplica as transformações necessárias para gerar o registro correspondente. Este componente então verifica se este registro existe ou não no PRB. Se o registro for encontrado, o PRA envia uma resposta positiva ao PTA com informações suplementares se estas existirem.

O *Packet Tracer Application* é a entidade interessada em rastrear um caminho percorrido por um pacote IP. Este componente aplica as transformações no pacote a ser rastreado para gerar uma chave de consulta. O PTA consulta então um ou mais PRAs para verificar se a chave de consulta existe em seus respectivos PRBs. Se um agente retornar uma resposta negativa o PTA assume que o pacote não foi visto no ponto de observação correspondente ao PRB.

A figura 2.7 ilustra os componentes desta arquitetura.

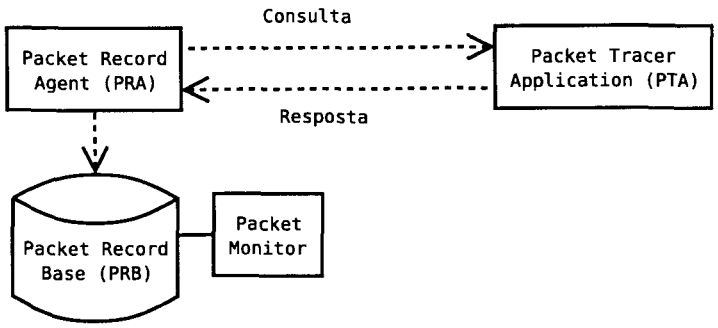

Figura 2.7: Uma arquitetura para rastreamento de pacotes IP.

2.5.1 Limitações de Escopo

Muitos detalhes da arquitetura proposta não são tratados no trabalho apresentado. O PTA de alguma maneira deve saber quais PRAs devem ser consultados. O PTA também precisa aplicar algum algoritmo para poder reconstruir o caminho percorrido por um pacote baseado nas respostas dos PRAs. Não são apresentados também que tipos de transformações um pacote deveria sofrer para poder armazenar um registro que o identificasse nem de como estas informações seriam armazenadas. Estes detalhes não são

discutidos em [34] por estarem fora do escopo da proposta.

Capítulo 3

Uma Arquitetura para Roteamento Reverso na Internet

Este capítulo apresenta a arquitetura proposta para rastreamento de pacotes IP na Internet, Esta arquitetura é baseada nas arquiteturas apresentadas no capítulo anterior. Da mesma forma que em [9], é utilizado um *Bloom Filter* para representar o *log* do tráfego da rede. Entretanto, os componentes estão distribuídos como em [34]. A arquitetura possui duas contribuições principais, a de retornar um grafo de ataque mais preciso obtendo exatamente o caminho inverso daquele percorrido pelo pacote rastreado e de oferecer um mecanismo dinâmico para determinar precisamente o período de tempo no qual os registros de pacotes podem ser mantidos, assumindo redes que possuem variações no seu nível de utilização.

A seção 3.1 descreve a arquitetura como um todo, apresentando seus componentes e descrevendo as relações entre eles. As seções 3.2, 3.3 e 3.4 descrevem o funcionamento de cada componente em separado. A seção 3.5 apresenta uma comparação com as arquiteturas apresentadas no capítulo anterior.

3.1 Arquitetura

Na arquitetura proposta, a rede é responsável por manter as informações necessárias para realizar um rastreamento. Cada nodo da rede deve manter um *log* do tráfego que ali

passou de maneira que seja possível realizar uma consulta para verificar se um dado pacote passou pelo nodo. A partir deste *log* basta utilizar técnicas apropriadas de extração para descobrir o caminho que um pacote percorreu, do seu destino até a sua origem.

A arquitetura é composta por três componentes:

- *Packet Monitor*: Este componente 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.

- *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.

- *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 3.1 ilustra os componentes e suas relações. Nas seções seguintes estes componentes são descritos em detalhes.

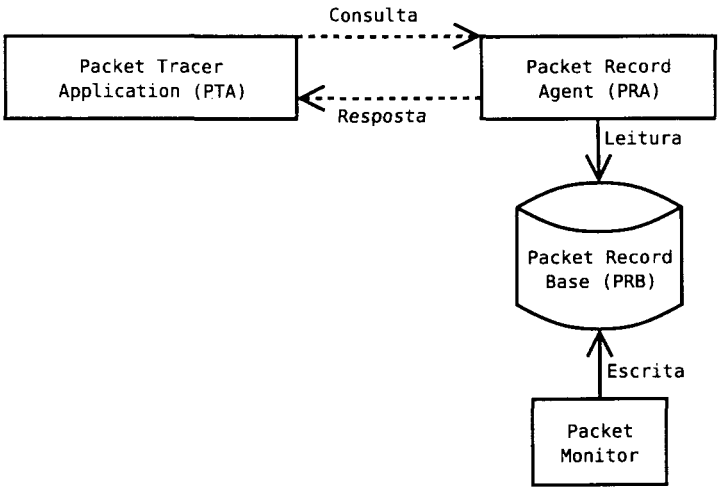

Figura 3.1: Componentes da arquitetura proposta.

3.2 *Packet Monitor*

O *Packet Monitor* deve manter um registro de todos os pacotes da rede de modo que seja possível verificar se um pacote qualquer pertence ao conjunto dos pacotes observados em um dado período. Conforme descrito em [9], o armazenamento de pacotes inteiros em uma base de dados se torna inviável devido à grande quantidade de espaço necessária. Logo, é utilizada a mesma estratégia do componente DGA da arquitetura SPIE [9]. São utilizados 30 bytes do pacote IP para identificar um pacote único, sendo 22 bytes do cabeçalho IP e 8 bytes do seu conteúdo. Como alguns campos do cabeçalho são alterados durante o processo de roteamento, alguns campos do pacote IP são mascarados. Ao contrário do DGA, o *Packet Monitor* não mascara o campo TTL para evitar o aparecimento de arestas no grafo de ataque que não existem no grafo real. Chamamos estas arestas de falsas arestas e elas são descritas detalhadamente na seção 3.5. A figura 3.2 mostra os campos utilizados pelo *Packet Monitor* para identificar um pacote único.

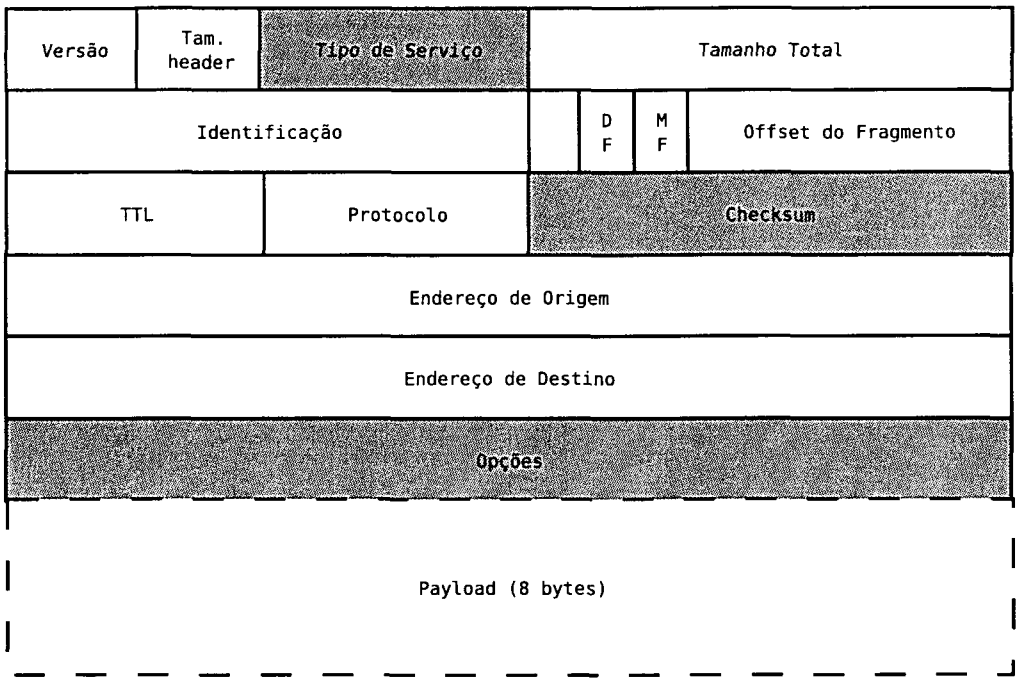

Figura 3.2: Os campos de um pacote IP. Campos em cinza são mascarados antes de ser aplicada a função para cálculo do resumo digital.

Após este processo o pacote mascarado é inserido em um *Bloom Filter*. Esta estrutura de dados permite a ocorrência de falsos positivos, entretanto, a taxa de ocorrência pode ser controlada através do número de funções de resumo digital utilizadas e o seu fator de capacidade, que é determinado pela divisão do número de elementos inseridos na estrutura pelo número de bits do vetor utilizado pelo *Bloom Filter*. O *Packet Monitor* recebe estes dois valores como parâmetro, podendo controlar assim a taxa máxima de falsos positivos.

O DGA da arquitetura SPIE também utiliza um *Bloom Filter*, paginando o vetor de bits gerado em uma memória secundária para consultas posteriores. Esta paginação ocorre através de um período de tempo fixo pré-determinado naquela arquitetura. O *Packet Monitor* da arquitetura proposta neste trabalho utiliza uma outra abordagem. Um *Bloom Filter* é paginado apenas quando atingir um fator de capacidade máximo conforme foi definido em sua configuração. Desta maneira, o período de tempo em que um *Bloom Filter* 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 *Bloom Filter* 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). A figura 3.3 ilustra o formato dos registros do PRB. O tamanho do campo vetor é determinado através do número de funções de resumo digital e do número de bits que elas retornam, ou seja, o seu tamanho é dependente da implementação deste componente. O tamanho máximo que pode ser ocupado por este conjunto de registros também é determinado na configuração do *Packet Monitor*. Ao atingir o tamanho máximo, os registros mais antigos são substituídos.

(32 bits)	(32 bits)	(32 bits)	(variável)
Timestamp	Número de Elementos	Núm. Funções	vetor

Figura 3.3: Formato dos registros do PRB.

3.3 *Packet Record Agent*

O objetivo do *Packet Record Agent* é de receber requisições do PTA para verificar se um dado pacote foi visto pelo *Packet Monitor*. O PRA recebe como entrada um pacote e o horário aproximado de sua chegada. O PRA não se comunica diretamente com o Packet Monitor, ele apenas processa o pacote realizando as transformações necessárias, e consulta o *Packet Record Base* que é alimentado pelo *Packet Monitor*. Logo, estes dois componentes devem compartilhar a mesma região de memória. Para realizar uma consulta no *Packet Record Base* basta fazer uma pesquisa binária utilizando o horário da paginação como índice.

O PRA também é responsável por manter informações sobre a topologia da rede. Cada PRA mantém uma lista estática de PRAs vizinhos que é definida na sua configuração. Ao receber uma requisição, o PRA responde se o pacote foi visto ou não, e caso afirmativo, informa a sua lista de vizinhos ao PTA que o requisitou.

3.4 *Packet Tracer Application*

A aplicação responsável pelo rastreamento precisa receber três parâmetros de entrada. O pacote a ser rastreado, o horário aproximado de sua chegada, e a vítima do pacote. A vítima deve ser expressa em termos do PRA mais próximo do host que recebeu o pacote.

Com estes parâmetros, o PTA consulta o PRA recebido, que deve retornar a sua lista de PRAs vizinhos. Como um pacote pode ter origens distintas ou pode ter sido duplicado em qualquer momento da rede, o PTA deve consultar todos os caminhos possíveis que um pacote pode ter percorrido. O PTA realiza uma busca em profundidade no grafo de ataque formado pela resposta de cada PRA, incrementando o TTL a cada nó visitado. O procedimento acaba quando nenhum PRA restante retorna uma resposta afirmativa. Não é necessário manter uma lista dos nós já visitados pois o incremento do TTL garante o término do algoritmo.

O algoritmo para efetuar o rastreamento é mostrado na figura 3.4.

```
Traceback (praAnterior, praAtual, pacote, ttl)
    Se praAtual.viu_ pacote(pacote,ttl)
        imprime praAnterior, praAtual
        Para todo vizinho de praAtual
            traceback(praAtual,vizinho,ttl+1)
        Fim Para
    Fim Se
Fim
```

Figura 3.4: Algoritmo para rastreamento de pacotes IP.

Como é necessária uma requisição para o primeiro nodo mais próximo da vítima e como para cada nodo consultado, deve ser feita uma requisição a todos os seus vizinhos, o número de requisições necessárias para realizar o rastreamento de um dado pacote é igual ao somatório do grau de cada nodo pertencente ao grafo de ataque mais um.

Logo, quanto maior o grau médio dos nodos da rede, maior será o número de requisições necessárias para completar um rastreamento. É essencial que este processo de rastreamento seja o mais rápido possível, para que as requisições sejam feitas enquanto

os registros referentes ao período de tempo solicitado ainda se encontram em memória. Para esconder a latência do tempo de resposta de cada PRA, as requisições ao conjunto de vizinhos podem ser realizadas em paralelo.

3.5 Contribuições

O problema do tamanho do *log* de pacotes e das implicações com relação à privacidade são resolvidos utilizando um *Bloom Filter*, assim como na arquitetura SPIE. Uma das implicações desta abordagem é a possibilidade de ocorrerem falsos positivos no grafo de ataque. Ou seja, podem ocorrer nodos no grafo que não foram utilizados realmente pelo pacote. Em [9], 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 *falsas arestas*. Ou seja, mesmo que os nodos do grafo sejam corretos, o grafo pode conter arestas que não foram realmente percorridas pelo pacote ou, por outro lado, pode não conter arestas que foram utilizadas.

O grafo gerado neste caso pode ser considerado incompleto e mais difícil de ser interpretado. Portanto, a precisão do grafo de ataque nesta arquitetura é definida em termos do número de falsos positivos e do número de falsas arestas do grafo de ataque gerado.

No processo de rastreamento da arquitetura SPIE, cada SCAR simula o algoritmo *Reverse Path Flooding* após coletar os *Bloom Filters* de cada DGA pertencente à sua região. A partir de um DGA que tenha observado um pacote, os registros de seus vizinhos são consultados para verificar se este pacote também foi visto. Este processo se repete até que nenhum DGA vizinho restante tenha visto o pacote. Uma lista dos nós já visitados é mantida para garantir o término do algoritmo.

Alguns octetos do cabeçalho de um pacote IP são alterados durante o processo de roteamento. Por exemplo, o campo TTL é decrementado e o campo checksum recalculado em cada hop. Um sistema de rastreamento deve poder identificar o mesmo pacote através de diversos nós da rede mesmo com estas alterações. Devido a isto, na arquitetura SPIE alguns campos do cabeçalho IP são mascarados, entre eles está o campo TTL.

Entretanto, se houver uma ou mais arestas não pertencentes ao grafo de ataque mas

que conectem dois nodos deste grafo, (links redundantes entre nodos da rede) pode ocorrer o aparecimento de falsas arestas. Ou seja, arestas pertencentes ao grafo de ataque podem ser omitidas e podem ser incluídas arestas não pertencentes ao grafo real, conforme ilustra o exemplo na figura 3.5. As flechas contínuas representam o caminho do ataque. Linhas tracejadas representam o grafo de ataque que seria retornado. Existem duas arestas pertencentes ao grafo de ataque que não pertencem ao caminho original percorrido pelo pacote. O exemplo é descrito a seguir.

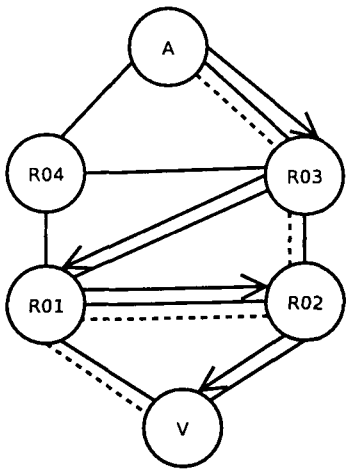

Figura 3.5: Exemplo de um grafo de ataque.

Apesar do link 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 V 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 problema é resolvido considerando também o campo TTL do pacote IP na realização de uma consulta de rastreamento. Sabe-se que um vizinho de um dado nodo X só vai preceder um outro nodo vizinho Y no grafo de ataque de um pacote, se o TTL do pacote observado em X for igual ao TTL observado em Y mais um. Desta maneira, basta fazer algumas alterações no algoritmo para a busca do caminho e não mascarar o campo TTL ao inserí-lo no *log* de pacotes. Desta maneira é possível obter exatamente o caminho

inverso daquele percorrido pelo pacote através do processo de roteamento; chamamos a este processo de *roteamento reverso*.

É contribuição deste trabalho também um novo método de paginação do *Bloom Filter*. Enquanto que na arquitetura SPIE a paginação ocorre em períodos fixos de tempo, a abordagem desta arquitetura permite que este período seja adaptado de acordo com a carga da rede. É definido um fator de capacidade máximo de tal forma que em períodos nos quais a carga é baixa, isto é, poucos pacotes são transmitidos, o *Bloom Filter* permanece por um período maior em memória primária. Por outro lado, nos períodos em que a carga é muito alta, evita que a taxa de falsos positivos aumente a um nível indesejável.

Capítulo 4

Resultados Experimentais

Este capítulo apresenta resultados experimentais obtidos da implementação da arquitetura proposta. Os resultados de dois experimentos são apresentados. O primeiro experimento consiste em executar o processo de rastreamento em uma rede simulada com 50 instâncias do *Packet Monitor* e do *Packet Record Agent*. São executados dois algoritmos, o primeiro utilizando a mesma abordagem vista na arquitetura SPIE e o segundo sendo o algoritmo da arquitetura proposta. São comparados os grafos de ataque obtidos nos dois casos. O segundo experimento compara a utilização de duas funções de resumo digital distintas utilizadas no *Packet Monitor*.

O restante deste capítulo está organizado como segue. A seção 4.1 descreve a implementação da arquitetura. As seções 4.2 e 4.3 descrevem o primeiro e o segundo experimento, respectivamente.

4.1 Implementação

O sistema foi implementado na linguagem C++ por favorecer a modularização e reaproveitamento de código e pela existência de uma grande biblioteca de classes. A figura 4.1 mostra o diagrama de classes utilizado para modelar o problema.

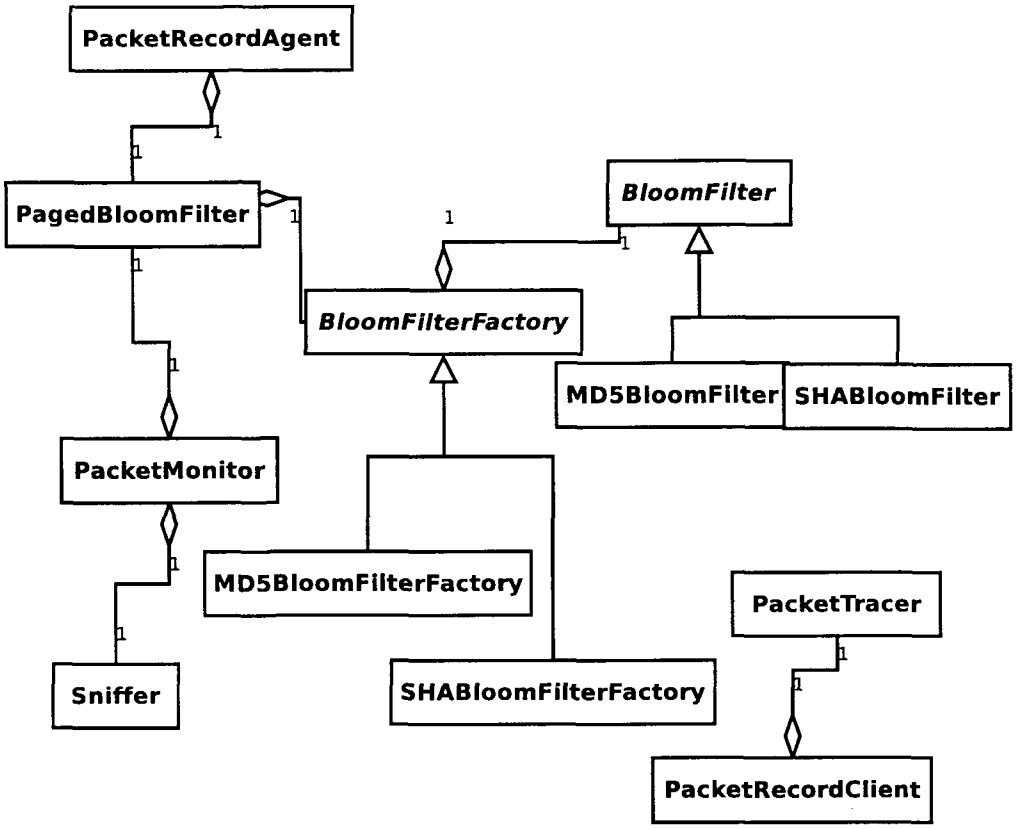

Figura 4.1: Diagrama de classes utilizado na implementação.

O *Packet Monitor* foi implementado como um *sniffer* utilizando a biblioteca PCAP (*Packet Capture*) [36]. Cada pacote lido é mascarado e inserido em um *Bloom Filter* que é armazenado em memória primária. Ao atingir o fator de capacidade máximo especificado, o conteúdo do *Bloom Filter* é escrito em um arquivo em memória secundária junto com algumas informações de controle, como o *timestamp* em que o *Bloom Filter* foi criado.

Duas funções de resumo digital foram testadas: MD5 (*Message Digest 5*) [11] e SHA (*Secure Hash Algorithm*) [12]. O resultado da função MD5, de 128 bits, foi dividido em 8 partes de 16 bits, simulando 8 funções de resumo digital independentes. Desta maneira, cada vetor de bits do *Bloom Filter* tem 2^{16} bits ou 8 kbytes. Já o resultado da função SHA, de 160 bits, foi dividido em 8 partes de 20 bits. Com a função SHA cada vetor de bits do *Bloom Filter* tem 2^{20} bits ou 128 kbytes. Esta divisão foi utilizada pois quanto maior o tamanho do vetor de bits de um *Bloom Filter*, menor a taxa de falsos positivos. Entretanto, uma menor divisão do resultado das funçoes de resumo digital, isto é, um

maior tamanho do vetor, impossibilitaria o seu armazenamento completo em memória primária nas máquinas utilizadas para testes.

O *Bloom Filter* foi definido com um fator de capacidade máximo de 9, para só então ser paginado em memória secundária. Com este valor, no máximo 7281 elementos são inseridos em um único *Bloom Filter* usando a função MD5 e 116508 elementos usando a função SHA. Teoricamente, isto garante um índice de falsos positivos de 1.45% para ambos os casos.

Os componentes *Packet Monitor* e *Packet Record Agent* são executados como dois processos distintos em uma mesma máquina. Ao receber uma requisição, o PRA consulta o arquivo gerado pelo *Packet Monitor*. O PRA deve pesquisar qual é o registro que deve ser lido. Como os registros são escritos em ordem do horário da paginação, é possível realizar uma pesquisa binária para achar o *Bloom Filter* correspondente ao *timestamp* do pacote. Caso o pacote seja muito recente e ainda não tenha sido paginado em memória secundária, é consultado o *Bloom Filter* que está em memória principal. Isto é possível pois o *Packet Monitor* e *Packet Record Agent* compartilham a mesma região de memória. Com esta estratégia, o *Packet Monitor* não precisa interromper o processo de captura de pacotes para se comunicar com o PRA, o que poderia causar a perda de alguns pacotes.

Foi necessário definir um protocolo de comunicação para as requisições do PTA e as respostas do PRA. O formato das mensagens é simples, conforme ilustrado nas figuras 4.2 e 4.3. O campo "versão" indica a versão do protocolo que deve ser utilizada para interpretar as mensagens. O campo "tipo msg." indica o tipo de mensagem, que pode ser uma consulta ao PRA ou uma resposta do PRA ao PTA. No caso de uma consulta, o campo "timestamp inicial" indica o horário em que o pacote foi observado. O campo seguinte indica o tamanho do pacote, seguido do pacote em si. Já no caso de uma reposta, o campo "tipo resp." indica se a resposta é positiva ou negativa. Em seguida é indicado o número de vizinhos do PTA consultado, seguido do endereço IP e porta de cada vizinho. No caso de uma resposta negativa, os endereços dos vizinhos não são comunicados.

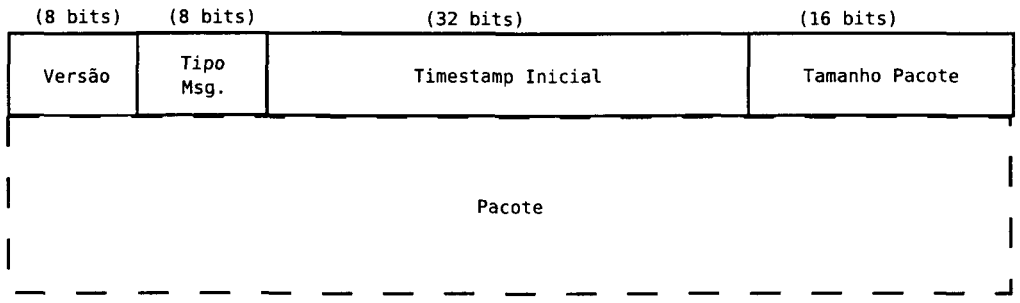

Figura 4.2: Formato de uma consulta ao PRA.

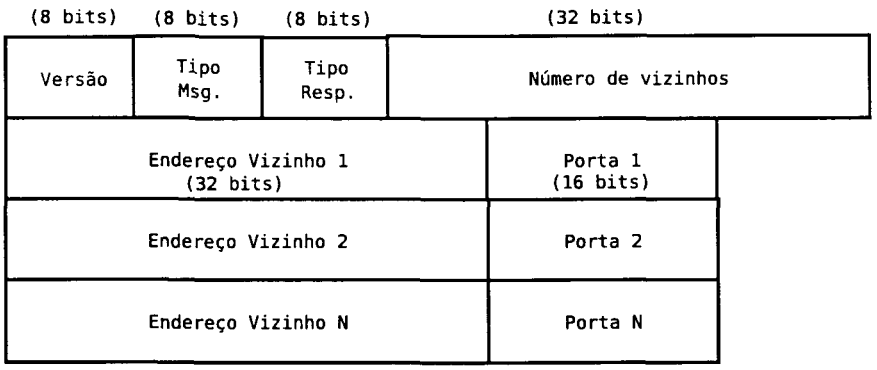

Figura 4.3: Formato de um resposta do PRA ao PTA.

O *Packet Tracer Application* é a aplicação que executa o rastreamento. O PTA 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 também ser gerado pelo utilitário *tcpdump* [36].

4.2 Resultados Experimentais

A identificação de um PRA é feita pelo seu endereço IP e a porta que serve requisições. Foi simulada uma rede de 50 nodos em uma única máquina. A biblioteca PCAP permite que os pacotes sejam lidos de um arquivo, e não necessariamente da interface de rede. Desta maneira foi possível gerar um tráfego sintético para cada *Packet Monitor*. Um

caminho de ataque foi simulado inserindo um mesmo pacote em diversos monitores da rede, variando apenas o campo TTL. Para monitores não pertencentes ao caminho de ataque, um tráfego aleatório foi utilizado.

Foram gerados 50 grafos de 50 nodos utilizando o método de Waxman [38] 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 existente 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.

Nesta simulação duas métricas foram analisadas. A primeira foi o número de requisições geradas pelo rastreamento. Esta métrica é importante pois tem influência no tempo total necessário para efetuar o rastreamento de um pacote. A segunda foi a precisão do grafo de ataque gerado. Nesta simulação, o tráfego gerado não implicava na geração de falsos positivos, apenas em falsas arestas. Foram utilizados dois 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*. Neste caso, uma lista dos nodos já visitados deve ser mantida para garantir o término do algoritmo. O segundo foi o algoritmo de rastreamento da arquitetura proposta, onde o campo TTL não é mascarado.

Os mesmos grafos e caminhos foram utilizados nos dois experimentos. Os resultados são exibidos nas tabelas 4.1, 4.2 e 4.3. A coluna "Grafo" indica o número do grafo testado. A coluna "Tam. Caminho" exibe o número de nós do caminho de ataque gerado. Em seguida são exibidos os parâmetros α e β para geração do grafo aleatório. As colunas "# req. (sem TTL)" e "# req. (com TTL)" indica o número de requisições realizadas para a execução do primeiro e do segundo algoritmo, respectivamente. A coluna "Arestas a mais" indica o número de arestas do grafo de ataque obtido que não pertencem ao caminho de ataque original. Já a coluna "Arestas a menos" indica o número de arestas que

estão contidas no caminho de ataque original mas que não aparecem no grafo de ataque retornado. Estes dois valores são obtidos a partir da execução do primeiro algoritmo, que mascara o campo TTL. A última coluna, "Total falsas arestas (com TTL)" indica o número total de falsas arestas da execução do algoritmo da arquitetura proposta.

Grafo	Tam. caminho	Parâm. α	Parâm. β	# req. (sem TTL)	# req. (com TTL)	Arestas a mais	Arestas a menos	Total falsas arestas (com TTL)
0	10	0.808669	0.568594	50	204	4	4	0
1	10	0.915408	0.671377	50	211	5	5	0
2	10	0.231484	0.308439	29	43	0	0	0
3	10	0.619036	1.15522	50	197	6	6	0
4	10	0.175216	0.941675	34	61	3	3	0
5	10	0.485991	0.686354	47	129	4	3	0
6	10	0.324351	1.35804	45	118	4	4	0
7	10	0.178027	0.578125	28	54	3	3	0
8	10	0.855678	0.963804	50	260	5	4	0
9	10	0.168569	1.14614	32	56	2	2	0
10	15	0.907496	0.522733	50	307	10	9	0
11	15	0.24293	0.8439	45	114	4	4	0
12	15	0.751609	1.24419	50	383	8	9	0
13	15	0.340186	1.39923	49	212	9	9	0
14	15	0.647868	0.288719	48	157	7	7	0
15	15	0.323883	0.812895	48	162	6	6	0

Tabela 4.1: Resultados experimentais, parte 1.

Grafo	Tam. caminho	Parâm. α	Parâm. β	# req. (sem TTL)	# req. (com TTL)	Arestas a mais	Arestas a menos	Total falsas arestas (com TTL)
16	15	0.453518	0.390714	47	113	3	3	0
17	15	0.386444	1.00826	50	206	7	7	0
18	15	0.590655	1.09217	50	277	6	6	0
19	15	0.486047	0.759299	50	214	9	8	0
20	20	0.106337	1.41285	41	91	4	4	0
21	20	0.810912	1.37801	50	568	14	15	0
22	20	0.350566	1.46795	50	262	8	9	0
23	20	0.217778	0.557753	47	113	3	3	0
24	20	0.987448	1.05393	50	609	14	11	0
25	20	0.37527	0.980862	50	269	10	10	0
26	20	0.916058	0.33154	50	288	11	12	0
27	20	0.958745	0.66182	50	489	14	14	0
28	20	0.150997	1.18203	45	106	5	6	0
29	20	0.14936	0.39847	41	112	6	6	0
30	25	0.73723	0.592109	50	469	18	17	0

Tabela 4.2: Resultados experimentais, parte 2.

Grafo	Tam. caminho	Parâm. α	Parâm. β	# req. (sem TTL)	# req. (com TTL)	Arestas a mais	Arestas a menos	Total falsas arestas (com TTL)
31	25	0.752025	0.759898	50	543	16	16	0
32	25	0.677126	1.02934	50	555	18	21	0
33	25	0.3953	0.238815	43	132	8	9	0
34	25	0.123294	0.45648	46	105	2	2	0
35	25	0.879902	1.17913	50	742	19	19	0
36	25	0.92907	0.394996	50	419	14	13	0
37	25	0.491907	0.168055	46	117	8	8	0
38	25	0.787778	1.40516	50	701	19	17	0
39	25	0.680264	0.96349	50	533	19	21	0
40	30	0.273564	0.271755	50	363	16	15	0
41	30	0.783431	1.27042	50	813	26	24	0
42	30	0.433282	0.973342	50	404	15	15	0
43	30	0.929904	0.389298	50	495	22	25	0
44	30	0.78542	1.11016	50	782	25	25	0
45	30	0.322373	1.14768	50	321	16	15	0
46	30	0.972588	0.22022	50	341	23	21	0
47	30	0.257208	1.38542	50	279	13	16	0
48	30	0.723421	1.36563	50	741	22	22	0
49	30	0.636613	1.2483	50	640	25	23	0

Tabela 4.3: Resultados experimentais, parte 3.

No pior caso da primeira simulação da arquitetura SPIE, todos os nodos recebem uma requisição. Entretanto, é possível reparar que apenas em uma simulação o grafo de ataque retornado foi idêntico ao grafo de ataque real, não contendo nenhuma falsa aresta. Todos as outras simulações continham arestas a mais, que não pertenciam ao grafo de ataque original, e também arestas a menos, que pertenciam ao grafo de ataque original mas não apareceram no grafo de ataque resultante. Observa-se também, que se no caminho gerado uma falsa aresta for retornada, geralmente a aresta verdadeira será excluída. Isto explica porque o número de arestas a mais e de arestas a menos tendem a ser muito parecidos.

Na segunda simulação foi utilizada a arquitetura proposta neste trabalho. Neste caso o campo TTL não foi mascarado. O número de requisições nesta simulação é igual ao somatório do grau dos nodos pertencentes ao grafo de ataque mais um. Logo, quanto maior o valor dos parâmetros α e β e quanto maior o tamanho do caminho maior será o número de requisições. É possível notar que o número de requisições é maior que o da simulação anterior. Por outro lado, o grafo de ataque retornado é exatamente o mesmo que o grafo original, não houve o aparecimento de nenhuma falsa aresta.

4.3 Avaliação Experimental da Taxa de Falsos Positivos

Foi realizada uma simulação para confirmar o índice de falsos positivos esperado utilizando as duas funções. Após preencher um *Bloom Filter* até seu fator de capacidade máximo, o índice de falsos positivos foi calculado realizando testes com aproximadamente 880000 pacotes distintos. A taxa de falsos positivos calculada foi de 1.441% para ambos os casos, apenas ligeiramente inferior ao resultado esperado. A figura 4.4 e 4.5 ilustra a convergência da taxa de falsos positivos em função do número de pacotes testados para a função MD5 e SHA, respectivamente.

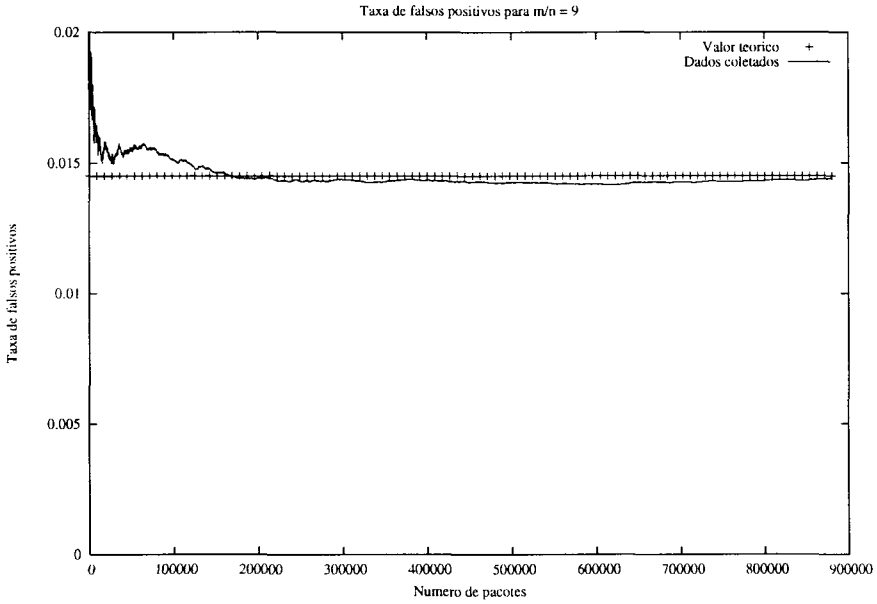

Figura 4.4: Taxa de falsos positivos utilizando MD5 e fator de capacidade 9.

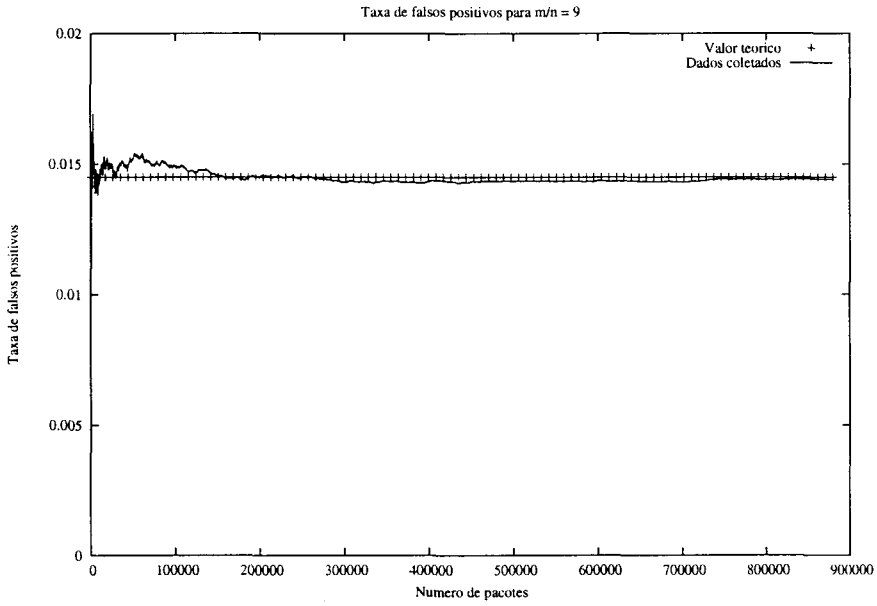

Figura 4.5: Taxa de falsos positivos utilizando SHA e fator de capacidade 9.

. Considerando o tamanho médio de um pacote IP como sendo de 1000 bytes e assumindo a total utilização do link, um *Packet Monitor* em um enlace a 100 Mbps iria processar em torno de 12500 pacotes/s. Desta maneira, um vetor de bits de um *Bloom Fil-*

ter utilizando MD5 poderia teoricamente representar no mínimo 582 ms de tráfego desta rede. Seria necessário então de aproximadamente 1158 Mbytes para representar o tráfego de um dia em uma rede de 100 Mbps. Já utilizando a função SHA seria possível representar no mínimo 9,320 segundos de tráfego no mesmo enlace sendo necessário também 1158 Mbytes para representar o tráfego de um dia inteiro.

A utilização da função SHA torna possível considerar sua utilização em um enlace de 1 Gbps. Neste caso um *Packet Monitor* processaria em torno de 125000 pacotes/s. Diminuindo o período de tempo que pode ser armazenado em um *Bloom Filter* para no mínimo 932 ms. Neste caso seria necessário no máximo 11587 Mbytes para representar o tráfego de um período de 24 horas.

Apesar das duas funções possuírem requisitos similares de armazenamento, uma vez que o fator de capacidade máximo se manteve o mesmo, é desejável que o tamanho do vetor de bits do *Bloom Filter* seja o maior possível para representar a maior quantidade de tempo. Isto porque no caso de não ser possível determinar com precisão o horário de chegada do pacote será necessário realizar a consulta em vários *Bloom Filters* armazenados na PRB, aumentando a taxa de falsos positivos.

Capítulo 5

Conclusão

Este trabalho apresentou uma arquitetura para realizar o rastreamento de pacotes IP na Internet. São duas as principais contribuições desta arquitetura. A primeira é a de retornar um caminho de ataque mais preciso. Em contraposição a trabalhos anteriores, esta arquitetura permite a determinação precisa de todo o caminho até a origem do pacote evitando o aparecimento de falsas arestas. Simulações foram executadas que demonstram que o grafo de ataque obtido pela arquitetura proposta é mais preciso que o grafo de ataque obtido pela arquitetura SPIE. A segunda contribuição é a proposta de uma nova abordagem para paginação do *Bloom Filter*, onde é definido o fator de capacidade máximo permitido. Em períodos nos quais a carga é baixa, isto é, poucos pacotes são transmitidos, o *Bloom Filter* permanece por um período maior em memória primária. Por outro lado, nos períodos em que a carga é muito alta, evita que a taxa de falsos positivos aumente a um nível indesejável.

Para recuperar os pacotes processados por um roteador, pode-se utilizar um *sniffer* presente no mesmo segmento de rede do roteador ou através de um agente conectado ao roteador através de alguma interface adequada do próprio roteador. Idealmente, um sistema de rastreamento deve ser largamente difundido na Internet. Entretanto, o rastreamento pode ser útil também se implantado apenas em um sistema autônomo, onde os problemas de cooperação entre várias redes são reduzidos.

Existem algumas limitações no sistema desenvolvido que são propostas como trabalhos

futuros. A implementação atual não considera aspectos de segurança na comunicação entre os componentes do sistema. Para que um sistema de rastreamento possa ser utilizado fora de ambientes de pesquisa é essencial que a identidade dos componentes seja validada. Este trabalho confia apenas no endereço IP para identificar um componente. Um ataque de *IP Spoofing* poderia comprometer o sistema.

O sistema precisa conhecer a topologia da rede para poder realizar o rastreamento. Atualmente a topologia é armazenada estaticamente em cada PRA do sistema, que mantém uma lista dos seus vizinhos. Qualquer mudança na topologia necessita de correção manual por parte do administrador do sistema. Trabalhos futuros incluem o estudo de uma maneira de facilitar esta administração de forma que os nodos possam reconhecer automaticamente mudanças na topologia da rede.

Um pacote pode sofrer algumas outras transformações ao ser processado em algum roteador. Estudos indicam que menos de 3% do tráfego IP passa por transformações [37]. Entretanto, um sistema de rastreamento deve ser capaz de reverter qualquer transformação realizada no pacote, pois um atacante pode tentar tirar vantagem destas transformações. A reversão destas transformações não foi considerada neste trabalho, pois cada tipo de transformação deve ser tratada de maneira distinta. O sistema pode ser alterado para que seja possível a inserção de *plugins* que permitam o tratamento de cada tipo de transformação apropriadamente.

Referências Bibliográficas

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

[2] B. Preneel "Analysis and Design of Cryptographic Hash Functions" *Thesis (Ph.D.)*, Katholieke Universiteit Leuven. Leuven, Belgium, 1993.

[3] 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.

[4] W. Richard Stevens, *TCP/IP Illustrated Vol. 1: The Protocols*, Addison-Wesley, 1994.

[5] 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.

[6] Partridge, C. "New Protocols to Support Internet Traceback," `http://www.ir.bbn.com/documents/internet_drafts/draft-partridge-ippt-discuss-00.txt`, acessado em 04/2004.

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

[8] Savage, S., Wetherall, D., Karlin, A. R. e Anderson, T. "Practical Network Support for IP Traceback," *Proceedings of the ACM Special Interest Group on Data Communications 2000 (SIGCOMM'2000)*, pp.295-306, 2000.

[9] 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.

[10] Wu, S. F., Zhang, L., Massey, D. e Mankin, A. "Intention-Driven ICMP Trace-Back," Internet Draft, IEFT, `draft-wu-itrace-intention-00.txt`, Fevereiro 2001.

[11] Rivest, R. "The MD5 Message-Digest Algorithm," *RFC 1321*, IETF, Abril 1992.

[12] Jones, P. "US Secure Hash Algorithm 1 (SHA1)," *RFC 3174*, IETF, Setembro 2001.

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

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

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

[16] Perkins, C. "IP Mobility Support," *RFC 2002*, IETF, Outubro 1995.

[17] Srisuresh, P. e Holdrege, M. "IP Network Address Translator (NAT) Terminology and Considerations" *RFC 2663*, IETF, Agosto 1999.

[18] Ferguson, P. "Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing," *RFC 2827*, IETF, May 2000.

[19] Crosby, S. A. e Wallach, D. S. "Denial of Service via Algorithmic Complexity Attacks," Rice University, *TR03-416*, Fevereiro 2003.

[20] SANS Institute, "Spoofing: An Overview of Some Current Spoofing Threats," `http://www.sans.org/rr/paper.php?id=321`, acessado em 04/2004.

[21] Trace Route, `http://www.traceroute.org/`, acessado em 04/2004.

[22] CERT Coordination Center, http://www.cert.org/, acessado em 04/2004.

[23] Huegen, C., "Smurf Attack Information," http://www.pentics.net/denial-of-service/white-papers/smurf.cgi, acessado em 04/2004.

[24] SANS Institute, "Introduction to IP Spoofing," http://www.sans.org/rr/paper.php?id=959, acessado em 04/2004.

[25] Todd, B., "Distributed Denial of Service Attacks," http://www.opensourcefirewall.com/ddos_whitepaper_copy.html, acessado em 04/2004.

[26] SANS Institute, "Spoofed IP Address Distributed Denial of Service Attacks: Defense-in-Depth," http://www.sans.org/rr/paper.php?id=469, acessado em 04/2004.

[27] 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 04/2004.

[28] NUA.com "How Many Online," http://www.nua.ie/surveys/how_many_online/world.html, acessado em 04/2004.

[29] SecurityStats.com "Security Statistics," http://www.securitystats.com/, acessado em 04/2004.

[30] Crothers, T., *An Overview of Intrusion Detection,Implementing Intrusion Detection Systems: A Hands-On Guide for Securing the Network*, Paperbock, 2002.

[31] Comer, D. E., *Internetworking with TCP/IP, Vol 1: Principles, Protocols and Architecture*, Prentice Hall, 2000.

[32] Russel, D. e Gangemi, G.T, *Computer Security Basics*, O'Reilly, 1992.

[33] 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.

[34] Keeni, G. M. "An Architecture for Ip Packet Tracing," http://www.ietf.org/internet-drafts/draft-glenn-ippt-arch-00.txt, acessado em 04/2004.

[35] Sager G. "Security Fun with OCxmon and cflowd," http://www.caida.org/projects/ngi/content/security/1198/mt0002.htm, acessado em 04/2004.

[36] tcpdump/libpcap "TCPDUMP Public Repository," http://www.tcpdump.org/, acessado em 03/2004.

[37] McCreary, S. e Claffy, K. "Trends in Wide Area IP Traffic Patterns: A View from Ames Internet Exchange," *ITC Specialist Seminar on IP Traffic Modeling, Measurement and Management*, pp 1-11, 2000.

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

[39] About.com "Hacker tools - Utilities used by hackers, crackers & phreaks.," http://netsecurity.about.com/cs/hackertools/, acessado em 04/2004.