LUCAS BUENO DOS REIS

COMPARANDO AS FERRAMENTAS DE IMPLEMENTAÇÃO DE FIREWALL IPCHAINS E IPTABLES

Monografia de graduação apresentada ao Departamento de Ciência da Computação Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para a obtenção do título de Bacharel em Ciência da Computação

Orientador
Prof. Bruno de Oliveira Schneider

LAVRAS
MINAS GERAIS - BRASIL
2001

LUCAS BUENO DOS REIS

COMPARANDO AS FERRAMENTAS DE IMPLEMENTAÇÃO DE FIREWALL IPCHAINS E IPTABLES

Monografia de graduação apresentada ao Departamento de Ciência da Computação Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para a obtenção do título de Bacharel em Ciência da Computação

APROVADA em 29 de junho de 2001

Prof. _____
Luiz Henrique Andrade Correia

Prof. _____
Joaquim Quinteiro Uchôa

Prof. _____
Bruno de Oliveira Schneider
(Orientador)

LAVRAS
MINAS GERAIS - BRASIL
2001

Dedico a meus pais Matilde e Messias, a meus irmãos Luciano e Lúcio e a meus amigos Adalberto e Leonardo.

AGRADECIMENTOS

Agradeço a Deus e a todas as pessoas que de alguma forma me ajudaram a concluir este estudo. Agradeço a meus pais por deixarem eu ficar tanto tempo com o carro e a meus amigos pela força nas horas desanimadoras.

RESUMO

IPChains e IPTables são filtros de pacotes do *kernel* do Linux utilizados na implementação de um *firewall*. Estas ferramentas bem utilizadas garantem alguma segurança em sua rede privada já que ela provavelmente será alvo de invasões por estar conectada a internet que é um ambiente bastante hostil. O IPTables é uma evolução do IPChains porém não possue muitas diferenças.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1. Introdução

O TCP/IP é o protocolo de comunicação da grande rede, a internet. Existe uma tendência muito grande de que todos os aplicativos e sistemas operacionais de alguma forma façam interação com a internet. Porém o aplicativo apenas gera a informação, quem faz a transmissão dos dados num alto nível é o TCP/IP. Este protocolo implantou um novo conceito no modo de transmissão entre redes, mesmo aquelas com sistemas operacionais diferentes.

Com o crescimento da internet, sua rede privada fica exposta a acessos não autorizados e uma tentativa de restringir esses acessos é a implementação de um *firewall* na sua rede privada. Em redes de computadores, *firewall* são barreiras interpostas entre a rede privada e a rede externa com a finalidade de evitar intrusos. Estes mecanismos de segurança são baseados em *hardware* e *software* e seguem a política de segurança estabelecida pela empresa.

Existem várias ferramentas para se implementar um *firewall*. Entre elas destacam os filtros de pacotes IPChains e IPTables do Sistema Operacional Linux.

Todo o tráfego da rede é enviado em forma de pacotes. O início de cada pacote diz para onde ele está indo, de onde vem, o tipo do pacote e outros detalhes administrativos. Estas ferramentas analisam os pacotes seguindo as regras definidas pelo administrador do sistema e decide o que fazer com eles. Ou deixa o pacote entrar na rede ou não deixa.

Num primeiro momento será explicado como é o protocolo de comunicação TPC/IP e o que é um *firewall*. Depois será explicado as ferramentas IPChains e IPTables e depois suas semelhanças e diferenças junto com as conclusões obtidas.

2. TCP/IP

O início se deu quando o departamento de defesa americano, no final da década de 60, decidiu criar um *software* de comunicação entre computadores que permitisse que estes pudessem trocar informações entre si independente de que estes computadores estivessem locais ou remotos, utilizassem sistemas operacionais e aplicativos diferentes e utilizassem equipamentos diferentes.

Naquele momento, o Departamento de Defesa Americano (DOD) patrocinou o desenvolvimento de um programa de comunicação que inicialmente seria utilizado apenas para uso de instituições militares.

Desenvolvido em conjunto com pesquisadores civis e militares, o DOD criou o *Transmission Control Protocol - Internet Protocol* (TCP/IP) também conhecido como Protocolo de Controle de Transmissão - Protocolo de Internet, um conjunto de programas que padroniza um sistema de comunicação.[2]

Este protocolo permitiu, no início dos anos 70, que computadores do sistema de defesa americano pudessem se comunicar, independentemente se os computadores estivessem locais ou não.

Para que isto fosse possível, foi necessário criar um sistema de comunicação onde a rede como um todo fosse dividia em pequenas redes independentes, isto é, se um segmento da grande rede parasse, somente este segmento seria afetado pela falha. Entraram em cena então os roteadores, que interligavam uma rede a outra isolando o tráfego, porém permitindo a troca de dados entre computadores nos diversos segmentos da rede.

O DOD sempre foi um dos principais compradores de sistema de informática e outras tecnologias que inicialmente poderiam ser utilizadas em momentos de guerra. E como grande comprador, em dado momento da história, emitiu um comunicado para os fabricantes de computadores e sistemas operacionais, informando que os novos equipamentos e sistemas que o departamento de defesa

americano viesse a comprar deveriam ter o sistema comunicação TCP/IP portado e funcionando para os seus sistemas operacionais.

Este tipo de atitude obrigou que todos os fabricantes de sistemas operacionais e aplicativos, como *IBM*, *HP*, *Digital*, e outros portassem este sistema de comunicação para os seus produtos.

Embora o TCP/IP tenha sido criado inicialmente por um órgão público, este se tomou, depois de um certo tempo, de domínio público, sendo possível conseguir a documentação das normas e, as vezes, o código fonte dos programas e aplicativos para TCP/IP e portar o mesmo para um determinado sistema operacional.

Por este motivo e principalmente por causa da internet, atualmente todos os sistemas operacionais possuem uma versão do TCP/IP, o que permite, por exemplo, que um usuário, utilizando um *PC* com *Windows 95*, possa trocar informações com outro usuário usando um computador *MAC* no qual rode o sistema operacional *System 7.5*.

2.1. A Internet e o TCP/IP

O surgimento da internet se deu no início da década de 70 quando o departamento de defesa americano interligou alguns de seus computadores a outras redes de computadores dos campus de pesquisas de universidades e outros centros de tecnologia, para, em conjunto com estes órgãos, trocar informações com os mestres e cientistas destes, criando assim uma inter-rede.

Os próximos computadores a se interligarem a esta inter-rede pertenciam a outras entidades de pesquisas, como laboratórios farmacêuticos, bibliotecas, o que permitia que milhares de pessoas destes órgãos trocassem informações entre si.

Na década de 80, quem entrou na grande rede foram as empresas que passaram a vender e comprar pela internet. Em 1986 foi criado um sistema de interfaciamento padrão de saída de dados chamado *HTML*, o que conhecemos hoje como *web* ou *www*, que facilitou o uso da internet pelos usuários.

Voltando agora para a existência do TCP/IP, fica fácil neste momento explicar o que o TCP/IP tem a ver com a existência da internet já que o departamento de defesa americano queria interligar os seus computadores com os centros de pesquisas e o sistema de comunicação que ele determinou foi o TCP/IP.

A partir deste momento, todos os computadores, para se ligarem na grande rede, também conhecida como internet e que hoje atravessa continentes, devem possuir uma cópia deste programa de comunicação instalada.

2.2. A importância do TCP/IP

Como este é o protocolo de comunicação da grande rede, existe uma tendência muito grande de que todos os aplicativos e sistemas operacionais de alguma forma façam interação com a internet. Porém o aplicativo apenas gera a informação, quem faz a transmissão dos dados num alto nível é o TCP/IP, que será obrigatório para todos os administradores de redes, programadores ou outras categorias de profissionais que vivam a partir da existência da informática.

O protocolo TCP/IP implantou um novo conceito arrojado no modo de transmissão mundial entre redes, mesmo as heterogêneas, aquelas com sistemas operacionais diferentes, e qualquer sistema que tenha por pretensão conectar o usuário na grande rede deve ter tais protocolos instalados.

A estrutura do *header* (cabeçalho) TCP é a seguinte:[2]

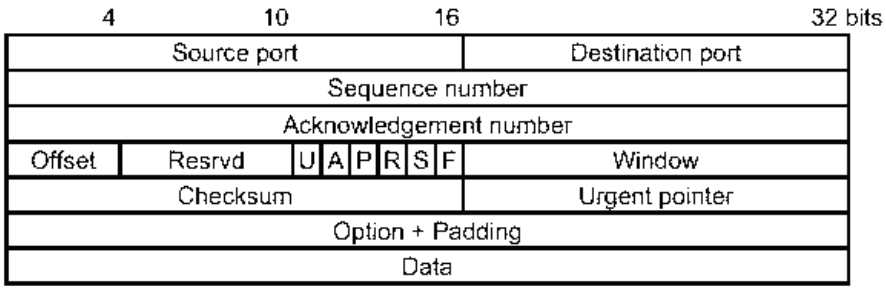


Figura 1 - Cabeçalho do TCP

Onde:

- *Source port* é o número da porta de origem;
- *Destination port* é o número da porta de destino;
- *Sequence number* é o número de seqüência do primeiro octeto de dados do segmento quando SYN não está presente. Se o SYN estiver presente, o número de seqüência é o número de seqüência inicial (ISN) e o primeiro octeto de dados é ISN + 1;
- *Acknowledgment number*, se o bit de controle ACK está acionado, o campo contém o valor do próximo número da seqüência que o destinatário do segmento está esperando receber;
- *Data offset* indica onde inicia o campo de dados dentro do *header* TCP;
- *Reserved*, seis *bits* reservados para uso futuro;
- Control *bits* são os *bits* de controle e podem assumir:
  - U (URG): Informa à aplicação a chegada de dados urgentes, que devem ser processado antes no *buffer*;
  - A (ACK): Indica que o campo *acknowledgment* é significante;
  - P (PSH): Função *push*;
  - R (RST): Reinicializa a conexão.;
  - S (SYN): Sincroniza os números da seqüência;
  - F (FIN): Indica o fim da transmissão de dados.

- *Window* são 16 *bits* do número de octetos de dados que o destinatário está aguardando receber, iniciando com o octeto que indica o campo *acknowledgment*;
- *Checksum* são 16 *bits* de controle de erros.

Esse protocolo tem como principal objetivo realizar a comunicação entre aplicações de dois *hosts* diferentes. O protocolo TCP é um protocolo de nível de transporte muito utilizado que trabalha com mensagens de reconhecimento, especificação do formato da informação e mecanismos de segurança. Ele garante que todos os pacotes serão enviados com sucesso pois realiza transmissões orientadas à conexão. Além disso, ele possibilita o uso de várias aplicações voltadas a conversação.

Quando executado, utiliza o protocolo IP, não orientado à conexão. O TCP então fica responsável pelo controle dos procedimentos da transferência segura de dados. Cabe salientar que o IP não é o único protocolo não orientado à conexão que pode ser utilizado pelo TCP, existe os protocolos ICMP, ARP.

Para maior eficiência nas comunicações, o TCP engloba várias funções que poderiam estar nas próprias aplicações, como processador de texto, base de dados e correio eletrônico. Ele foi criado com o intuito de ser um *software* universal contendo essas funções.

Existem ainda outros serviços do TCP:

- Gerenciamento da informação nas transmissões orientadas a conexão - O protocolo TCP pode controlar todos os aspectos da informação que esta sendo transmitida, pois é um protocolo de transmissão orientada à conexão. A ação do TCP se estende a toda a trajetória da informação onde ele procura garantir o sucesso da transmissão.

- Prioridade e segurança - O protocolo TCP permite que o administrador do *host* controle os níveis de segurança e permissão de acesso, bem como as prioridades nas conexões.
- Transferência orientada a *stream* - Os aplicativos em geral enviam dados ao TCP de forma orientada a *stream*, onde a informação é transmitida *byte* a *byte*, um após o outro. Quando a informação chega ao TCP, é então agrupada em pacotes e assim enviada aos demais níveis de transmissão.
- Controle de fluxo - O controle de fluxo atribui uma janela de transmissão ao *host* de origem. Essa janela limita o número de *bytes* transmitidos por vez. O controle de fluxo em si está na possibilidade de atribuir diferentes valores às janelas.
- Confiabilidade na transmissão - A confiabilidade nas transmissões via TCP está baseada no fato do protocolo ser orientado à conexão e trabalhar com números de reconhecimento seqüenciais e positivos.

O TCP do *host* origem transfere os dados em forma de octetos, onde a cada octeto vão sendo atribuídos números em seqüência. O TCP do *host* destino analisa então esses números para garantir a ordem e a integridade da mensagem enviada. Se a transferência for perfeita, o TCP do *host* destino envia uma mensagem de reconhecimento à origem. Caso contrário, é enviada uma seqüência numérica para o TCP do *host* origem que informará o tipo do problema, bem como ordenará uma nova transmissão.

Os números em seqüência podem ser utilizados ainda para eliminar octetos duplos, que por causa da transmissão não orientada à conexão podem ocorrer.

O TCP da origem e do destino possuem *timers*, um espaço de tempo também conhecido como *time-out*, para garantir que não se perca muito tempo

entre uma mensagem errada e sua correção. A margem de tempo do *time-out* é controlada pelos administradores dos *hosts*.

2.3. O protocolo IP

A principal função do protocolo IP é transportar os datagramas de uma rede a outra na internet. Ele é um protocolo de transmissão não orientado à conexão e por ser mais básico não apresenta muitas características do TCP. Podemos dizer que o IP :

- Não possui mecanismos de retransmissão;
- Não dá garantia de uma transmissão íntegra ou ordenada;
- Utiliza os endereços IP como base para o direcionamento dos datagramas;
- Descarta um datagrama se ele não for entregue ou se passar por muitos *hops*;
- Suas operações e padrões estão descritos em vários RFC´s (*Request for Comments*) e IEN´s (*Internet Engeneering Notes*).

Embora o protocolo IP não possua essas características, elas não deixam de ser importantes. Por isso, toda essa parte de consistências para a integridade dos dados transmitidos fica com o TCP.

Ao realizar trocas de pacotes, os aplicativos da internet se deparam com um problema que é a diferença do tamanho das mensagens nas diversas redes. Nesse caso, o protocolo IP suporta o processo de fragmentação, onde os datagramas são divididos em unidades menores.

O procedimento de fragmentação é realizado por um *gateway*, onde as mensagens são partidas em unidades menores e adequadamente identificadas. O

host destino então reagrupará as instruções baseado nas identificações do *gateway*.

Na identificação dos fragmentos o *gateway* cria um *header* para cada fragmento. O *header* contém os endereços iniciais das redes (*Source address*) e uma identificação referente à mensagem a qual faz parte.

Já na fase de reagrupamento, o *host* destino, ao receber o primeiro fragmento, independentemente de estar na ordem correta, aciona um *timer* (*Time to Live* - Tempo de Vida). Se uma margem de tempo, medida em número de hops, padrão for ultrapassada e a mensagem não estiver completamente reconstituída, o *host* destino descarta os fragmentos recebidos e retorna à origem uma mensagem de erro.

Estrutura do *header* IP: [2]

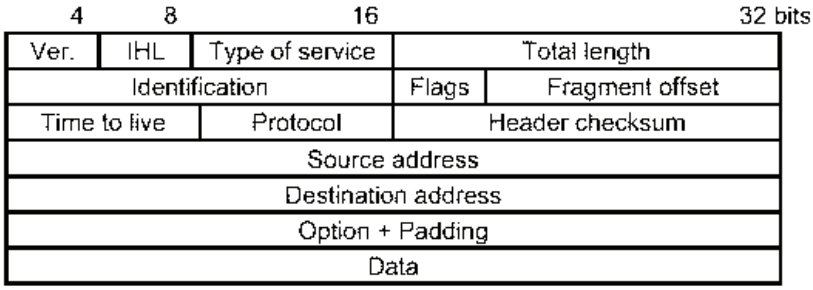


Figura 2 - Cabeçalho do IP

Onde:

- *Ver*. controla a versão do protocolo a que o datagrama pertence;
- *IHL* informa o tamanho do cabeçalho;
- *Type of service* permite que o *host* informe à sub-rede o tipo de rede que deseja;
- *Total lenght* é o tamanho total do pacote;
- *Idendification* é a identificação;

- *Fragment offset* informa a que ponto do datagrama atual o fragmento pertence;
- *Option* permite que versões posteriores do protocolo incluam informações.

2.4. Endereços IP

Na Internet, cada *host* e cada roteador tem um endereço IP que codifica seu número de rede e número de *host*. A combinação é exclusiva: duas máquinas nunca têm o mesmo endereço IP. Todos os endereços IP têm 32 bits e são usados nos campos *source address* e *destination address* dos pacotes IP. Os formatos usados para endereços IP são mostrados na Figura abaixo. Essas máquinas conectadas a diversas redes têm endereços IP distintos em cada rede.[4]

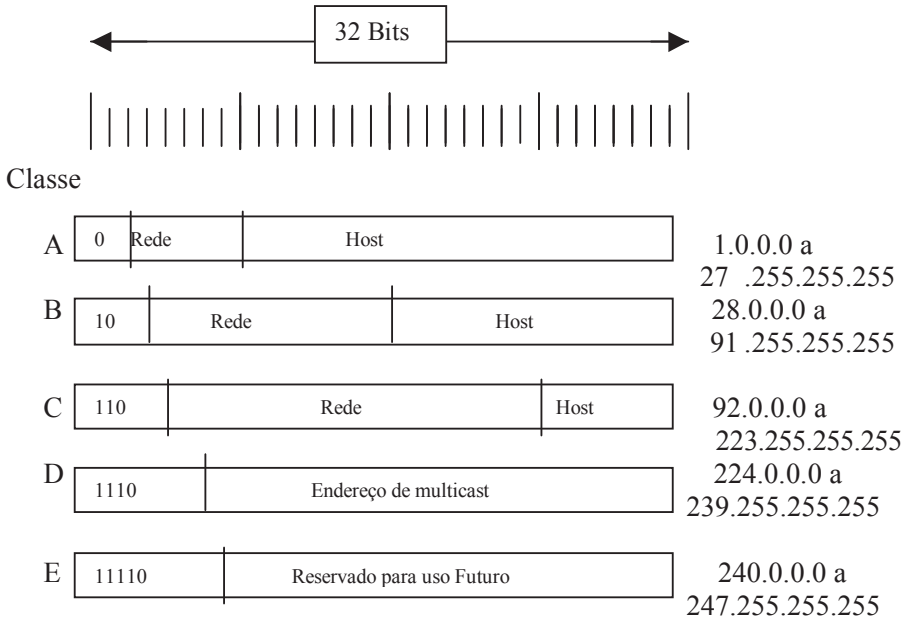


Figura 3 - Endereços IP

3. Firewall

"Antigamente, paredes de tijolos eram construídas entre construções em complexos de apartamentos de forma que se ocorresse um incêndio ele não poderia se espalhar de uma construção para a outra. De uma forma completamente natural, as paredes foram chamadas de *firewall*". [3]

Em redes de computadores, *firewalls* são barreiras interpostas entre a rede privada e a rede externa com a finalidade de evitar intrusos. Estes mecanismos de segurança são baseados em *hardware* e *software* e seguem a política de segurança estabelecida pela empresa.

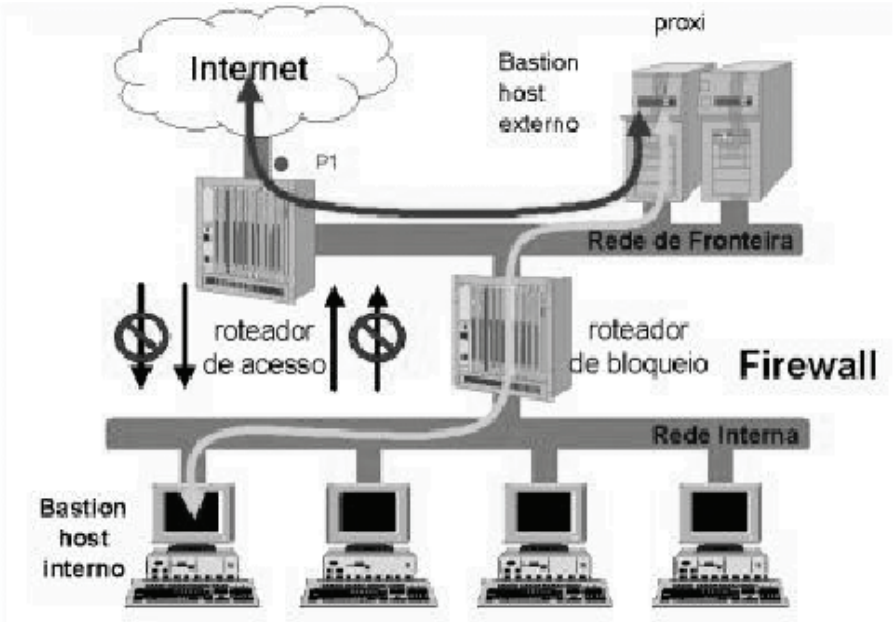


Figura 4 - Firewall

Evitando os acessos indevidos, os dados, os recursos e principalmente a reputação da empresa ficam protegidos.

3.1. Arquiteturas de Firewall

A arquitetura de um *firewall* consiste em um conjunto de componentes organizados de forma a garantir certos requisitos de segurança. Os componentes básicos para a construção de um *firewall* são os p*acket filters*, responsáveis pela filtragem dos pacotes que trafegam entre dois segmentos de rede e os b*astion hosts*, computadores responsáveis pela segurança de um ou mais serviços da rede.

3.1.1 Packet Filters

Como um primeiro passo ao se implementar uma barreira de segurança em uma rede de computadores é fundamental que se conheça os detalhes dos protocolos de comunicação utilizados. Na internet, a atenção deve ser voltada aos protocolos IP, TCP, UDP e ICMP. Estes são os principais protocolos que são considerados e examinados ao se estabelecer regras de filtragem em um *packet filter* para a internet. Este mecanismo de filtragem possibilita que se controle o tipo de tráfego de rede que pode existir em qualquer segmento de rede; conseqüentemente, pode-se controlar o tipo de serviços que pode existir no segmento de rede. Serviços que comprometem a segurança da rede podem, portanto, ser restringidos.

Com o exposto acima, fica evidente que um *packet filter* não se encarrega de examinar nenhum protocolo de nível superior ao nível de transporte, como por exemplo o nível de aplicação que fica como tarefa dos *application gateways* (*bastion hosts*). Portanto, qualquer falha de segurança de aplicação não pode ser evitada utilizando somente um *packet filter*.

O componente que realiza a filtragem de pacotes geralmente é um roteador dedicado, mas também pode ser um *host* de propósito geral configurado co-

mo roteador, e recebe a denominação de *screening router*. Deve-se ressaltar que o processo de filtragem de pacotes acarreta num overhead ao sistema; portanto, para uma situação de alto tráfego é necessário que se utilize um roteador com uma velocidade de processamento compatível com as necessidades.

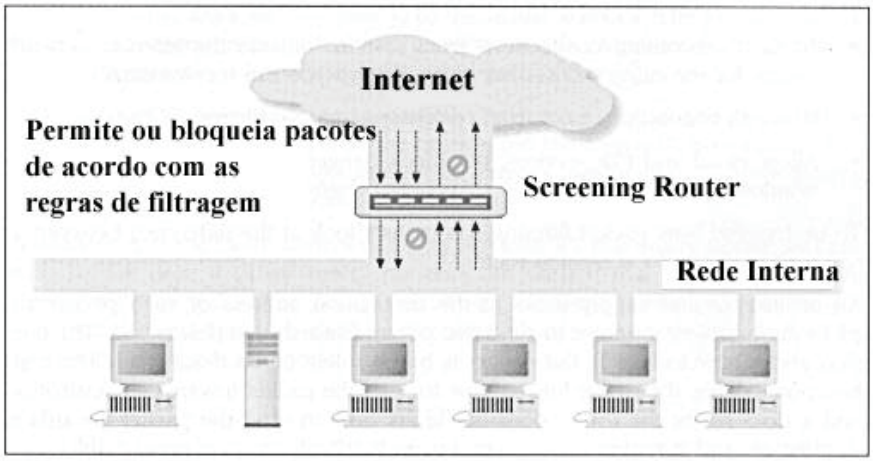


Figura 5 - Atuação de um Screening Router

Frisando que a filtragem dos pacotes não considera protocolos acima do nível de transporte, não é tomada nenhuma decisão baseada no conteúdo dos pacotes, ou seja, nos dados dos pacotes propriamente ditos.

A filtragem que a maioria dos *screening routers* realizam são baseadas nas seguintes informações:

- Endereço IP fonte;
- Endereço IP destino;
- Protocolo: Se o pacote é TCP, UDP ou ICMP;
- Portas TCP ou UDP fontes;
- Portas TCP ou UDP destino;
- Tipo de mensagem ICMP (se for o caso).

Algumas vantagens dos *packet filters* são:

- Pode ajudar a proteger toda uma rede, principalmente se este é o único roteador que conecta a rede interna à Internet;
- A filtragem de pacotes é transparente e não requer conhecimento nem cooperação dos usuários;
- Está disponível em muitos roteadores.

Algumas desvantagens são:

- As ferramentas de filtragem atualmente disponíveis não são perfeitas;
- Alguns protocolos não são bem adaptados para a filtragem;
- Algumas políticas não podem ser aplicadas somente com a filtragem de pacotes.

Quando se aplica alguma restrição em algum protocolo de mais alto nível, através de números de portas, espera-se que nada além do próprio serviço esteja associado àquela porta, entretanto, usuários internos mal intencionados podem subverter este tipo de controle colocando outro programa associado a essa porta. Um *firewall* não é apropriado para se defender de ameaças internas.

3.1.2 Bastion Hosts

Bastion host é qualquer máquina configurada para desempenhar algum papel crítico na segurança da rede interna; constituindo-se na presença pública na Internet, provendo os serviços permitidos segundo a política de segurança da empresa.

Um *bastion host* deve ter uma estrutura simples, de forma que seja fácil de garantir a segurança. É importante que se esteja preparado para o fato de que

o *bastion host* seja comprometido, considerando que ele provavelmente será alvo de ataques.

Este tipo de máquina também recebe a denominação de *application gateway* porque funciona como um *gateway* a nível de aplicação. Os servidores disponíveis nos *bastion host* são denominados de *proxy servers*; ou seja, servidores por procuração que atuam como intermediários entre o cliente e o servidor. Neste caso, os serviços só podem ser providos via *bastion host*, obrigando o cliente, por exemplo, via regras de filtragem nos roteadores, a acessar estas máquinas, portanto, este é um mecanismo que garante que o serviço será provido de forma segura para usuários internos e externos e impede que o *bastion host* seja desviado.

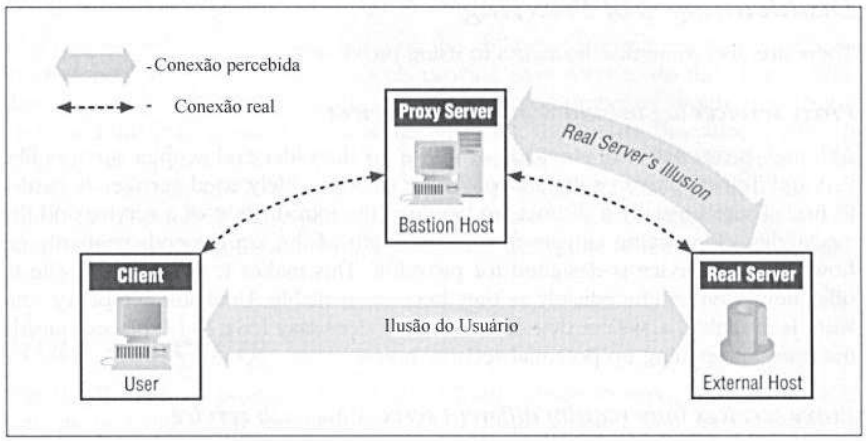


Figura 6 - Funcionamento genérico de um *Proxy Server*

Algumas vantagens de sistemas procuradores são:

- Permitem aos usuários acesso direto aos serviços na internet: apesar de haver um procurador atuando em nome do cliente, este mantém a ilusão de estar comunicando diretamente com o servidor remoto;
- Bons mecanismos de *log*: como todo o tráfego dos serviços procurados passa pelo servidor procurador, e tudo até o nível de aplicação,

uma grande quantidade de informações podem ser registradas de acordo com as necessidades de auditoria e segurança.

Algumas desvantagens dos servidores procuradores são:

- Há um atraso significativo entre o surgimento de um novo serviço e um correspondente servidor *proxy*;
- Pode ser necessário utilizar diferentes servidores procuradores para cada serviço;
- Geralmente requerem modificações nos clientes, nos procedimentos ou em ambos;
- Alguns serviços não são viáveis para operar via procuradores (exemplo: *talk*, que é parte baseado em TCP e parte em UDP);
- Um serviço por procuração não protege contra todas as deficiências dos protocolos, depende da habilidade de se determinar que operações são seguras em um determinado protocolo.

Quando um servidor procurador atende a um único serviço, recebe a denominação de servidor dedicado ou *application level*. Quando um servidor atende a uma certa gama de serviços este é denominado servidor genérico ou *circuit level*. Um procurador genérico não pode interpretar o protocolo de aplicação e precisa obter informações através de outros meios para poder atender aos serviços. Esta é a principal desvantagem de um servidor genérico.

4. IPChains

Todo o tráfego da rede é enviado em forma de pacotes. O início de cada pacote diz para onde ele está indo, de onde vem, o tipo do pacote e outros detalhes administrativos. O início do pacote é chamado de cabeçalho, *header*. O restante do pacote contendo o dado atual que está sendo transmitido é usualmente chamado de corpo, *body*.

Alguns protocolos, como TCP, que é usado para o tráfego na *web*, *mail*, e *logins* remotos, usam o conceito de conexão, ou seja, antes dos pacotes com os dados que serão enviados vários pacotes de configuração são trocados dizendo "eu desejo conectar", "ok" e "obrigado". Então os pacotes normais de dados são trocados.

Um filtro de pacotes é uma peça de *software* que olha o cabeçalho do pacote quando eles passam e decide o seu destino. Ele pode decidir negar, aceitar ou rejeitar o pacote. Vale lembrar a diferença entre negar e rejeitar. Quando nega-se um pacote ele é simplesmente ignorado e quando rejeita-se um pacote uma mensagem é enviada para quem o enviou dizendo que ele foi rejeitado.

No Linux, a filtragem de pacotes é embutida no *kernel*. O princípio geral é de olhar o cabeçalho dos pacotes e decidir seu destino.

Neste contexto entra o IPChains, responsável por administrar o filtro de pacotes IP no *kernel* do Linux 2.2. É uma evolução do IPFwadm pois esse antigo filtro de pacotes não negociava com fragmentos, tinha contadores de 32 bits, não permitia especificação de outros protocolos senão TCP, UDP ou ICMS, não fazia grandes alterações dinamicamente, não especificava regras contrárias, possuía alguns truques e era difícil de gerenciar. [3]

É necessário usar um filtro de pacotes em sua rede privada pois com ele temos maior:

- Controle - quando você está usando um computador Linux para conectar sua rede interna a outra rede, a internet por exemplo, você tem a oportunidade de permitir certos tipos de tráfego e desativar outros. O cabeçalho do pacote contém o endereço de destino do pacote, assim você pode preveni-los de irem para certas partes fora de sua rede interna.
- Segurança - quando um computador Linux é a única coisa entre a internet e sua rede privada é necessário conhecer como restringir o que vem da rede externa.
- Vigilância - muitas vezes uma máquina mal configurada na rede local pode decidir enviar pacotes para fora da rede. É bom dizer ao filtro de pacotes para avisá-lo se alguma coisa de anormal ocorrer.

Além disso, quando sua rede interna estiver ligada à internet pode querer que ninguém acesse um determinado site, que não se tenha acesso a um tipo de serviço e, mais que tudo, que ninguém fique bisbilhotando sua rede. Todas essas medidas podem ser realizadas com o IPChains.

4.1. IPChains em Detalhes

O IPChains, escrito por *Rusty Russel*, não é um software simples mas conhecendo o seu mecanismo e seus conceitos o administrador de rede pode escolher a política de segurança de sua rede. [8]

A ferramenta IPChains comunica-se com o *kernel* e diz a ele quais pacotes filtrar, inserindo ou apagando regras da seção de filtragem de pacotes do *kernel*. Ele substituiu o IPFwadm, que foi usado pelo antigo código de *firewall* de IP como dito anteriormente.

O *kernel* inicia com três listas de regras. Estas listas são chamadas *firewall chains* ou simplesmente *chains*. Os três *chains* são chamados *input*, *output* e *forward*. Quando um pacote chega, o *kernel* usa o *chain* de entrada e decide seu destino. Se ele sobreviver a este passo então o próximo passo do *kernel* é decidir para onde enviar o pacote, isto é chamado roteamento. Se o seu destino for outra máquina ele consulta o *chain* forward. Finalmente, antes de simplesmente o pacote ir para fora da rede o *kernel* consulta o *chain output*.

Um *chain*, daí o seu nome, é uma lista de checagem de regras. Cada regra diz "se o cabeçalho do pacote parece com isto, então aqui está o que fazer com o pacote". Se a regra não confere com o pacote, então a próxima regra no *chain* é consultada. Finalmente, se não existem mais regras a consultar, então o *kernel* procura no policiamento do *chains* para decidir o que fazer.

Estas são as etapas que o pacote passa quando chega em um servidor Linux com IPChains:

- *Checksum* - teste para verificar se o pacote não está corrompido de alguma forma.
- *Sanity* - muitos pacotes mal formados podem deixar o código de checagem de regras confuso e estes são negados aqui.
- *Input chain* - este é o primeiro *firewall chain* que será testado no pacote.
- *Routing decision* (decisão do roteamento) - o campo de destino é examinado pelo código de roteamento para decidir se o pacote deve ir para um processo local ou direcionado para uma máquina remota.
- *Local process* (processo local) - um processo sendo executado em uma máquina pode receber pacotes após o passo de decisão do roteamento e pode enviar pacotes.
- *Interface lo* - se pacotes de um processo local são designados a um processo local, eles vão através do *chain output* com a interface con-

figurada para *lo*, então retorna pelo *chain input* também com a interface *lo*. A interface *lo* é normalmente chamada de interface *loopback*.

- Local - se o pacote não foi criado por um processo local então o *forward chain* é checado, caso contrário o pacote vai através do *chain output*.
- *Forward chain* - este *chain* é usado em qualquer pacote que está tentando passar de uma máquina para outra.
- *Output chain* - este *chain* é usado em todos os pacotes antes de serem enviados para fora da rede.

4.1.1. Regras do IPChains

Existem diferentes outras coisas que você pode fazer com IPChains para isso é necessário iniciar os três *chains*, *input*, *output* e *forward*, que não podem ser apagados. Contudo pode criar outros *chains*.

Tabela1 - Comandos para a criação de chains.

Descrição	Comando
Criar um novo chain	-N
Deletar um chain vazio	-X
Listar as regras em um chain	-L
Zerar os contadores de pacote e byte em todas as regras no chain	-Z

Um alvo diz ao *kernel* o que fazer com um pacote que confere com uma regra. O IPChains usa "-j". Existem seis alvos especiais. Os três primeiros, *ACCEPT*, *REJECT* e *DENY* são muito simples. *ACCEPT* aceita o pacote, *DENY* bloqueia o pacote, é como se nunca o tivesse recebido. *REJECT* rejeita o pacote, gera uma resposta para a origem para dizer que o destino está inalcançável.

Tem ainda o alvo *MASQ* que diz ao *kernel* para mascarar o pacote. Este alvo é somente válido para pacotes atravessando o *chain forward*. O alvo *REDIRECT* que diz ao *kernel* para enviar um pacote para uma porta local em vez de enviar para o destino original. Este alvo é somente válido para pacotes atravessando o *chain input*.

Existem diversos meios de manipular regras dentro do *chain*:

Tabela 2 - Comandos para a manipulação de regras

Descrição	Comando
Adiciona uma nova regra no chain	-A
Insere uma nova na mesma posição no chain	-I
Substitui uma regra na mesma posição no chain	-R
Deleta uma regra na mesma posição no chain	-D
Deleta a primeira regra que confere no chain	-D

As regras do IPChains especificam uma configuração de condições em que o pacote se encontra e o que fazer quando a encontra, alvo. Por exemplo, você pode desejar negar todos os pacotes ICMS vindo do endereço IP 127.0.0.1. Assim neste caso nossas condições são que o protocolo deve ser ICMS e que o endereço de origem deve ser 127.0.0.1. Nosso alvo é NEGAR. Então nós adicionamos (-A) no *chain input* uma regra especificando que para pacotes vindo de 127.0.0.1 ("-s 127.0.0.1") com o protocolo ICMS ("-p ICMS") devem ser negados ("-j DENY"). Então a regra IPChains fica a seguinte: ipchains -A input -s 127.0.0.1 -p ICSM -j DENY.

Você deve sempre usar "-p" para especificar um protocolo, e "-s" para especificar um endereço de origem, mas existem outras opções que nós podemos usar para especificar características do pacote.

Endereços IPs de origem (-s) e destino (-d) podem ser especificados em quatro meios. O meio mais comum é usar o nome completo, como "*localhost*" ou "*www.linuxhq.com*". O segundo método é especificar o endereço IP como

"127.0.0.1". O terceiro e quarto meio permite especificar um grupo de endereços IPs, como "199.95.207.0/24" ou "199.95.207.0/255.255.255.0". Estes dois especificam qualquer endereços IP de 192.95.207.0 para 192.95.207.255 inclusive; os dígitos depois da "/" dizem que a parte do endereço IP é significante. "/32" ou "/255.255.255.255" é o padrão (abrange todos os endereços IPs). Para especificar qualquer endereço IP, deve ser usado "/0", como no exemplo: ipchains -A input -s 0/0 -j DENY. Isto é raramente usado, com o efeito acima é o mesmo como não especificar a opção "-s".

O protocolo pode ser especificado com a opção "-p". O protocolo pode ser um número ou um nome para casos especiais de "TCP", "UDP" ou "ICMP". Caso especificar "tcp" também funcionará como "TCP".

Para casos especiais onde um protocolo TCP ou UDP é especificado, existem argumentos extras indicando a porta TCP ou UDP, ou uma faixa de portas. Uma faixa é representada pelo caracter ":", como "6000:6010", que abrange 11 números de portas, de 6000 até 6010. Se o menor número é omitido, o padrão será 0. Se o maior número for omitido, o padrão será 65535. Assim para especificar conexões TCP vindo de portas abaixo de 1024, a sintaxe deve ser como "-p TCP -s 0.0.0.0/0 :1023". Números de portas podem ser especificadas por nomes, exemplo "*www*".

Note que a especificação de portas devem ser precedidas por uma "!", que inverte a regra. Assim para especificar qualquer pacote TCP menos uma pacote *www*, você deve especificar -p TCP -d 0.0.0.0/0 ! www. É importante mostrar que a especificação -p TCP -d ! 192.168.1.1 www é muito diferente de -p TCP -d 192.168.1.1 ! www. A primeira especifica qualquer pacote TCP para a porta *www* em qualquer máquina menos 192.168.1.1. A segunda especifica qualquer conexão TCP para qualquer porta em 192.168.1.1 menos a porta *www*.

5. IPTables

IPTables é um administrador de filtros de pacotes IP usado para montar, manter e inspecionar as tabelas de regras do filtro de pacotes no *kernel* do Linux. Há várias tabelas que podem ser definidas, cada tabela contém várias cadeias (*chains*) embutidas e pode conter cadeias definidas pelo usuário. Cada cadeia é uma lista de regras que são testadas nos pacotes: cada regra especifica o que vai acontecer com o pacote no momento em que ele chega no servidor. Isto é chamado de *target* (objetivo) que pode ser um salto para a uma cadeia definida pelo usuário da mesma tabela.

5.1. Targets (Objetivos)

Uma regra de *firewall* especifica critérios para um pacote, e um objetivo. Se o pacote não combina, a próxima regra da cadeia o examinará; se combina, então a próxima regra é especificada pelo valor do *target* que pode ser o nome de uma cadeia definida pelo usuário ou um dos especiais valores *ACCEPT* (aceita), *DROP* (nega), *QUEUE* (fila) ou *RETURN* (retorno).

ACCEPT deixa passar o pacote. *QUEUE* passa o pacote para *userspace*. *RETURN* é especificado pela política de segurança da empresa que determina o destino do pacote.

5.2. Tabelas

Há três tabelas atuais. A opção -t especifica em qual tabela deve-se olhar as regras. Temos as seguintes tabelas: *INPUT* para pacotes que entram, *FORWARD* para pacotes que são roteados e *OUTPUT* para pacotes gerados localmente. A tabela *nat*, tabela de tradução de endereços de rede, é consultada quando um pa-

cote é criado com uma nova conexão encontrada. Consiste de três construções: *PREROUTING* para alterar pacotes assim que eles entrarem, *OUTPUT* para alterar pacotes gerados localmente antes do roteamento e *POSTROUTING* para alterar pacotes como eles estão saindo. Tem duas cadeias imbutidas: *PREROUTING* para alterar pacotes que entram antes do roteamento e *OUTPUT* para alterar localmente pacotes gerados antes do roteamento.

Tabela 3 - Comandos no IPTables.

Descrição	Comando
Adiciona uma regra no final do chain	-A
Deleta uma regra no final do chain	-D
Troca uma regra do chain	-R
Lista todas as regras do chain	-L

Por exemplo, para ignorar os pedidos de *telnet* de um computador que tem o endereço IP 200.200.200.1 a regra ficaria a seguinte:

- iptables -A INPUT -s 200.200.200.1 -p tcp --destination-port telnet -j DROP

5.3. Compatibilidade com o IPChains

O IPTables é bem parecido com o IPChains. A diferença principal é que as cadeias *INPUT* e *OUTPUT* só são atravessadas pelos pacotes que vêm do *host* local e originário do *host* local respectivamente. Conseqüentemente todo pacote só passa por uma das três cadeias; previamente um pacote remetido passaria por todas as três.

O *target DENY* do IPChains agora é *DROP*.

A outra diferença principal é que a opção -i recorre à *interface* de entrada; -o recorre à interface de saída e ambos são disponível para pacotes que entram na cadeia *FORWARD*.

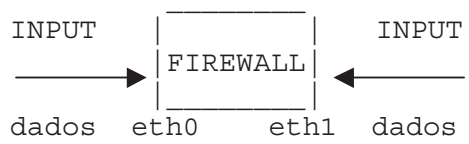
IPTables é um puro filtro de pacote ao usar a tabela de filtro *default*, com módulos de extensão opcionais. Isto deve simplificar muito a confusão prévia em cima do combinação de IP mascarado e filtro de pacote.

Há várias outras mudanças em IPTables, uma delas é quanto ao fluxo de dados que será explicado no capítulo seguinte.

6. Fluxo com IPChains e IPTables (Kernel 2.2 e 2.4 respectivamente)

6.1. Fluxo com IPChains (Kernel 2.2)

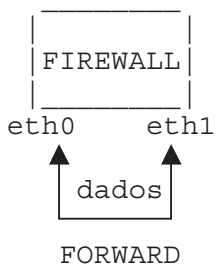
6.1.1 Input



Quaisquer pacotes de dados cujo sentido seja o *firewall* é considerado como *INPUT* não importando de onde venham.

As regras de *INPUT* são consideradas as mais importantes regras em um firewall e que normalmente a configuramos como sendo *DENY* por *default*, pois sempre será mais seguro negar tudo no filtro e apenas liberar as portas e IPs que se deseja utilizar e nunca o inverso.

6.1.2 Forward



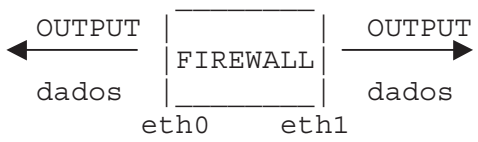
Quaisquer passagens de pacotes entre interfaces no *firewall*, é considerado como sendo *FORWARD*, não importando o sentido do pacote e sim a mudança de *interface*.

Não existe regra para se restringir acesso entre interfaces no *FORWARD*. Não se pode criar uma regra que negue determinado acesso vindo pela *interface* eth0 e que cujo destino seja a *interface* eth1. As regras são limitadas apenas a IPs, portas e protocolos em um *FORWARD*.

Veremos que isso irá mudar com o IPTables.

Também defini-lo por *default* como sendo *DENY*. Não existem regras de *FORWARD* para máquinas que possuam apenas uma interface de rede conectadas ao micro, mas pode passar a existir se nessa mesma máquina for utilizado um modem para acesso remoto a outra rede. Exemplo: eth0 - interface de rede e ppp0 - interface representada pelo *modem*. Nesse caso passa a ter duas interfaces e com isso um possível *FORWARD*. Podemos até admitir que na máquina onde haja roteamento entre redes poderemos implementar regras de *FOWARD*.

6.1.3. Output

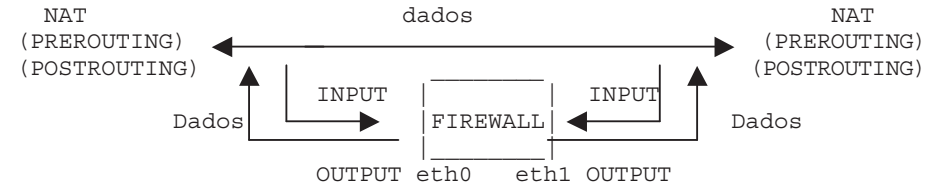


O mesmo acontece com qualquer saída de pacote de dados do *firewall*, todas as saídas são consideradas *OUTPUT*.

As regras de *OUTPUT* nem sempre são utilizadas pois são as últimas regras do filtro pelos quais os pacotes de dados passam antes de irem para a rede. Nesse momento alguns administradores admitem terem filtrado ao máximo os dados que passaram pelas regras de *INPUT* e *FORWARD* do *firewall*.

6.2. Fluxo com IPTables (Kernel 2.4)

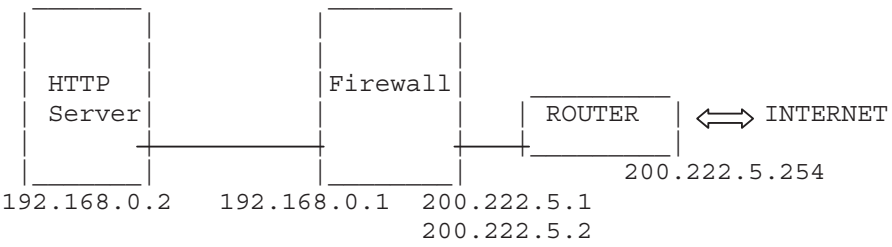
6.2.1 Forward



Como podemos observar, temos algumas fases que não existiam no IPChains e as que existiam tiveram o seu fluxo de dados alterado. Vamos então analisá-los:

Para os administradores que queriam usar o NAT (*Network Address Translation*) agora podem implementá-lo com o IPTables no *kernel*. [10]

6.3 Alguns Exemplos



Firewall:

eth1 = 192.168.0.1

eth0 = 200.222.5.1

eth0:0 = 200.222.5.2

NAT = 192.168.0.2 <==> 200.222.5.2

Com o *NAT* podemos trocar o IP não público (192.168.0.2) pelo IP público (200.222.5.2) e vice-versa. Todo pacote que vir pela internet para acessar o servidor 200.222.5.2, passa pela regra *PREROUTING* do *firewall* e logo em seguida é feito um DNAT (*destination NAT*) que troca o IP destino 200.222.5.2 por 192.168.0.2 e em seguida o pacote é encaminhado para o filtro, verificado nas regras de *FORWARD* e aí então chega ao servidor destino. Quando sair um pacote do IP 192.168.0.2 com destino a internet, o mesmo passa novamente pelas regras de *FORWARD* e logo em seguida passa para o *POSTROUTING* que faz o SNAT (source *NAT*) onde é trocado o IP 192.168.0.2 pelo IP 200.222.5.2 e aí sim vai para a internet. O nome dado a esse tipo de acesso é NAT 1:1.

Percebe-se que os pacotes tanto de entrada quanto de saída não passaram pelas regras de *INPUT* e *OUTPUT*, porque essas regras só dizem respeito a pocotes direcionados ao *firewall*.

Qualquer pacote cujo destino seja o *firewall* então o mesmo passa pelas regras de *INPUT* do *firewall* e todo pacote que sair do *firewall* passa pelas regras de *OUTPUT* do mesmo.

Dessa forma temos como proteger mais ainda o *firewall* pois podemos usar *DENY* para qualquer regra *INPUT* pois isso não afetaria o acesso ao servidor *HTTP* mostrado acima.

Para todos os pacotes de dados que passam de uma rede para outra através do *firewall*, passam pelas regras de *FORWARD* e agora com o IPTables, podemos especificar regras que controlem pacotes que entrem por uma interface em específico e saiam por outra interface:

- iptables -A FORWARD -p tcp --dport 80 -s 0/0 -i eth0 -o eth1 -j ACCEPT

O "-i" indica interface de entrada (*input*) e o "-o" interface de saída (*output*). Lembrando que esses parâmetros não funcionam com o *FORWARD* do IPChains.

Para fazer o *NAT* acima bastariam 2 regras:

- iptables -A PREROUTING -t nat -d 200.222.5.2/32 -j DNAT --to 192.168.0.2
- iptables -A POSTROUTING -t nat -s 192.168.0.2/32 -j SNAT --to 200.222.5.2

Para funcionar é necessário criar o alias eth0:0 no *firewall* com o IP 200.222.5.2 .

Além disso o novo filtro traz o tão esperado *Stateful Packet* utilizado por programas como o *Firewall-1*. Com esse recurso o filtro consegue gerenciar, por exemplo, as conexões feitas pelas estações e assim não deixar aberturas para scanners atuarem como acontecia com o tão conhecido protocolo *FTP* e outros.

Antigamente se fazia uma regra assim no *firewall* para permitir que estações pudessem acessar páginas na internet:

- ipchains -A input -p tcp -s 0/0 80 -d 0/0 1024: ! -y -j ACCEPT

Isto diz ao *firewall* para aceitar qualquer pacote cuja porta origem seja a 80/tcp que não seja do tipo SYN e isso pode ser muito bem utilizado por scanners que utilizam pacotes ACK para checarem serviços ativos.

Com o novo filtro ficaria assim:

- iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
- iptables -A FORWARD -m state --state ESTBLISHED,RELATED -j ACCEPT

Dessa forma o *firewall* só aceita conexões estabelecidas pelas estações e não precisamos deixar em aberto acessos vindos da porta 80/tcp, pois o *firewall* recusaria. Dessa forma os *scanners* foram anulados. Podemos também trabalhar com o ftp em modo ativo, pois para cada conexão 21/tcp o *firewall* espera uma outra conexão 20/tcp de retorno, o filtro vai gerenciar isso.

7. Conclusões

As diferenças entre o IPChains e o IPTables não são muitas mas as principais são de sintaxe na formação das regras.

Apesar do IPTables ser uma evolução do IPChains, o primeiro ainda está muito recente portanto não existe muita documentação a seu respeito e até mesmo um certo receio em utilizá-lo. Já o IPChains está bem consolidado e muito bem documentado, tanto escrita quanto on-line.

Neste contexto os administradores estão adotando a nova ferramenta IPTables mas com muita ponderação e cuidado. Uma grande vantagem do IPTables é que ele já suporta a nova versão do IP, o IPV6.

Como a internet cresceu muito na última década e tende a crescer cada muito mais, cada vez mais nós precisamos de segurança em nossa rede privada e todas as ferramentas que podem ser utilizadas para garantir esse segurança são úteis.

É provável que o IPTables, daqui a algum tempo, substituirá totalmente o IPChains uma vez que o IPChains substituiu o IPFwadm seguindo a evolução da tecnologia.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] MANN, S.; MITCHELL, E.L. **Linux System Security**. Editora Pretince Hall PTR, 2000. 564p.

[2] TANENBAUM, A.S. **Redes de Computadores**, Editora Campus Ltda 1997. 923p.

[3] ANÔNIMO. **Segurança Máxima Para Linux**. Editora Campus, 2000. 761p.

[4] SOARES, L.F.; LEMOS, G.; COLCHER, S. **Redes de Computadores**. Editora Campus, 1995. 705p.

[5] CONECTIVA, Inc. **CONECTIVA S.A.**, 2001 http://conectiva.com.br, Maio 2001.

[6] SOFTWARE ENGINEERING INSTITUTE e CARNEGIE MELLON UNIVERSITY. **CERT Coordination Center**, 2001. http://www.cert.org, Junho 2001.

[7] ATOMICTANGERINE, Inc. **SecurityPortal**, 2001. http://www.securityportal.com. Junho 2001.

[8] Securenet – O Portal de Segurança da Informação, 2001 http://www.securienet.com.br, Junho 2001.

[9] CIPSGA, Inc. **Cipsga**, 2001.
http://www.cipsga.org.br, Junho 2001.

[10]UnderLinux.com.br – O Partal do Administrador de Sistemas, 2001.
http://www.underlinux.com.br,. Junho 2001.

[11] Linux.com :: Linux.com Security:: Firewalls, Iptables and Rules, 2001.
http://www.linux.com, Junho 2001.

[12] Linux Security Brasil, Inc **Linux Security Brasil**, 2001.
http://www.linuxsecurity.com.br, Junho 2001.