Pontes de Filtragem (Filtering Bridges)

Resumo

Muitas vezes é útil dividir uma rede física (como uma Ethernet) em dois segmentos separados sem precisar criar sub-redes e usar um roteador para conectá-los. O dispositivo que conecta as duas redes dessa forma é chamado de bridge. Um sistema FreeBSD com duas interfaces de rede é suficiente para atuar como uma ponte.

Uma bridge funciona analisando os endereços de nível MAC (endereços Ethernet) dos dispositivos conectados a cada uma de suas interfaces de rede e, em seguida, encaminhando o tráfego entre as duas redes apenas se a origem e o destino estiverem em segmentos diferentes. Em muitos aspectos, um bridge é semelhante a um switch Ethernet com apenas duas portas.

Índice

1. Por que usar uma ponte de filtragem?

Cada vez mais frequentemente, graças à redução nos custos das conexões de banda larga à Internet (xDSL) e também devido à redução dos endereços IPv4 disponíveis, muitas empresas estão conectadas à Internet 24 horas por dia e com poucos (às vezes nem mesmo 2) endereços IP. Nestas situações, muitas vezes é desejável ter um firewall que filtre o tráfego de entrada e saída da Internet, mas uma solução de filtragem de pacotes baseada em roteador pode não ser aplicável, seja devido a problemas de sub-rede, o roteador pertence ao provedor de conectividade (ISP), ou porque não suporta tais funcionalidades. Nestes cenários, é altamente recomendável o uso de uma ponte de filtragem (filtering bridge).

Um firewall baseado em bridge pode ser configurado e inserido entre o roteador xDSL e seu hub/switch Ethernet sem problemas de numeração IP.

2. Como Instalar

Adicionar funcionalidades de bridge a um sistema FreeBSD não é difícil. Desde a versão 4.5 é

possível carregar tais funcionalidades como módulos ao invés de ter que reconstruir o kernel, simplificando bastante o procedimento. Nas subseções seguintes, explicarei as duas formas de instalação.

Não siga ambas as instruções: um procedimento *exclui* o outro. Selecione a melhor opção de acordo com suas necessidades e habilidades.

Antes de continuar, certifique-se de ter pelo menos duas placas Ethernet que suportem o modo promíscuo tanto para recepção quanto para transmissão, uma vez que elas devem ser capazes de enviar pacotes Ethernet com qualquer endereço, não apenas o próprio. Além disso, para obter um bom desempenho, as placas devem ser placas de barramento PCI master. As melhores opções ainda são as placas Intel EtherExpress™ Pro, seguidas pela série 3Com® 3c9xx. Para simplificar a configuração do firewall, pode ser útil ter duas placas de fabricantes diferentes (usando drivers diferentes) para distinguir claramente qual interface está conectada ao roteador e qual está na rede interna.

2.1. Configuração do Kernel

Então você decidiu usar o método de instalação mais antigo, mas bem testado. Para começar, você precisa adicionar as seguintes linhas ao seu arquivo de configuração do kernel:

```
options BRIDGE
options IPFIREWALL
options IPFIREWALL_VERBOSE
```

A primeira linha é para compilar o suporte à bridge, a segunda é para o firewall e a terceira é para as funções de registro do firewall.

Agora é necessário compilar e instalar o novo kernel. Instruções detalhadas podem ser encontradas na seção Compilação e Instalação de um Kernel Personalizado do FreeBSD Handbook.

2.2. Carregamento de Módulos

Se você escolheu usar o novo e mais simples método de instalação, a única coisa a fazer agora é adicionar a seguinte linha ao /boot/loader.conf:

```
bridge_load="YES"
```

Dessa forma, durante a inicialização do sistema, o módulo bridge.ko será carregado juntamente com o kernel. Não é necessário adicionar uma linha semelhante para o módulo ipfw.ko, pois ele será carregado automaticamente após a execução das etapas na seção seguinte.

3. Preparação Final

Antes de reiniciar para carregar o novo kernel ou os módulos necessários (de acordo com o método

de instalação escolhido anteriormente), você deve fazer algumas alterações no arquivo de configuração /etc/rc.conf. A regra padrão do firewall é rejeitar todos os pacotes IP. Inicialmente, configuraremos um firewall `open`, para verificar sua operação sem problemas relacionados à filtragem de pacotes (caso você esteja executando este procedimento remotamente, essa configuração evitará que você fique isolado da rede). Adicione estas linhas em /etc/rc.conf:

```
firewall_enable="YES"
firewall_type="open"
firewall_quiet="YES"
firewall_logging="YES"
```

A primeira linha ativará o firewall (e carregará o módulo ipfw.ko caso não esteja compilado no kernel), a segunda configurará o firewall no modo `open` (conforme explicado em /etc/rc.firewall), a terceira linha não exibirá o carregamento de regras e a quarta linha ativará o suporte a registro de logs.

Sobre a configuração das interfaces de rede, o modo mais comum é atribuir um endereço IP a apenas uma das placas de rede, mas a ponte funcionará igualmente mesmo que ambas as interfaces ou nenhuma tenha um endereço IP configurado. No último caso (sem IP), a máquina da ponte ainda será mais oculta e inacessível pela rede: para configurá-la, é necessário fazer login pelo console ou por meio de uma terceira interface de rede separada da bridge. Às vezes, durante a inicialização do sistema, alguns programas exigem acesso à rede, por exemplo, para resolução de domínio: nesse caso, é necessário atribuir um endereço IP à interface externa (a que está conectada à Internet, onde o servidor DNS reside), pois a bridge será ativada no final do procedimento de inicialização. Isso significa que a interface fxp0 (no nosso caso) deve ser mencionada na seção ifconfig do arquivo /etc/rc.conf, enquanto a xl0 não. Atribuir um endereço IP a ambas as placas de rede não faz muito sentido, a menos que, durante o procedimento de inicialização, aplicativos precisem acessar serviços em ambos os segmentos Ethernet.

Uma outra coisa importante a se saber é que, ao executar IP sobre Ethernet, existem na verdade dois protocolos Ethernet em uso: um é o IP e o outro é o ARP. ARP faz a conversão do endereço IP de um host em seu endereço Ethernet (camada MAC). Para permitir a comunicação entre dois hosts separados pela bridge, é necessário que a bridge encaminhe pacotes ARP. Esse protocolo não está incluído na camada IP, uma vez que ele existe apenas com IP sobre Ethernet. O firewall do FreeBSD filtra exclusivamente na camada IP e, portanto, todos os pacotes não-IP (incluindo ARP) serão encaminhados sem serem filtrados, mesmo se o firewall estiver configurado para não permitir nada.

Agora é hora de reiniciar o sistema e usá-lo como antes: haverá algumas novas mensagens sobre a bridge e o firewall, mas a bridge não será ativada e o firewall, estando no modo `aberto` , não evitará nenhuma operação.

Se houver algum problema, você deve resolvê-los agora antes de prosseguir.

4. Habilitando a Bridge

Neste ponto, para habilitar a ponte, você tem que executar os seguintes comandos (tendo a

perspicácia para substituir os nomes das duas interfaces de rede fxp0 e xl0 with your own ones) com as suas próprias):

```
# sysctl net.link.ether.bridge.config=fxp0:0,xl0:0
# sysctl net.link.ether.bridge.ipfw=1
# sysctl net.link.ether.bridge.enable=1
```

A primeira linha especifica quais interfaces devem ser ativadas pela bridge, a segunda irá habilitar o firewall na bridge e, finalmente, a terceira irá habilitar a bridge.

Neste ponto, você deverá ser capaz de inserir a máquina entre dois conjuntos de hosts sem comprometer as habilidades de comunicação entre eles. Se sim, o próximo passo é adicionar as partes `net.link.ether.bridge.[blah]=[blah]` dessas linhas ao arquivo /etc/sysctl.conf, para que elas sejam executadas na inicialização.

5. Configurando o Firewall

Agora é hora de criar seu próprio arquivo com regras personalizadas de firewall, para proteger a rede interna. Haverá alguma complicação em fazer isso porque nem todas as funcionalidades do firewall estão disponíveis em pacotes encaminhados através de uma bridge. Além disso, há uma diferença entre os pacotes que estão em processo de encaminhamento e os pacotes que estão sendo recebidos pela máquina local. Em geral, os pacotes de entrada são filtrados apenas uma vez, não duas vezes como normalmente ocorre; de fato, eles são filtrados apenas no momento do recebimento, portanto, as regras que usam `out` ou `xmit` nunca darão match. Pessoalmente, eu uso `in via`, que é uma sintaxe mais antiga, mas que faz sentido quando você lê. Outra limitação é que você está restrito a usar apenas comandos `pass` ou `drop` para pacotes filtrados por uma ponte. Coisas sofisticadas como `divert`, `forward` ou `reject` não estão disponíveis. Tais opções ainda podem ser usadas, mas apenas no tráfego destinado ou originado na máquina que atua como bridge em si (se ela tiver um endereço IP).

O conceito de filtragem stateful é novo no FreeBSD 4.0. Esta é uma grande melhoria para o tráfego UDP, que geralmente é uma solicitação que sai, seguida logo depois por uma resposta com o mesmo conjunto de endereços IP e portas (mas com origem e destino invertidos, é claro). Para firewalls que não têm o controle de estado, quase não há como lidar com esse tipo de tráfego como uma sessão única. Mas com um firewall que pode "lembrar" de um pacote UDP de saída e, nos próximos minutos, permitir uma resposta, lidar com serviços UDP é trivial. O exemplo a seguir mostra como fazer isso. É possível fazer a mesma coisa com pacotes TCP. Isso permite evitar alguns ataques de negação de serviço e outros truques desagradáveis, mas também faz com que a tabela de estado cresça rapidamente em tamanho.

Vamos dar uma olhada em um exemplo de configuração. Note primeiro que no início de /etc/rc.firewall, já existem regras padrão para a interface de loopback lo0, então não precisamos mais nos preocupar com elas. As regras personalizadas devem ser colocadas em um arquivo separado (digamos /etc/rc.firewall.local) e carregadas na inicialização do sistema, modificando a linha de /etc/rc.conf onde definimos o firewall `open`:

```
firewall_type="/etc/rc.firewall.local"
```

Você tem que especificar o *caminho completo*, caso contrário ele não será carregado com o risco de permanecer isolado da rede.

Para o nosso exemplo, imagine ter a interface fxp0 conectada ao exterior (Internet) e a xl0 ao interior (LAN). A máquina bridge tem o IP 1.2.3.4 (não é possível que o seu provedor de serviços de Internet possa lhe fornecer um endereço exatamente como este, mas para o nosso exemplo é suficiente).

```
# Things that we have kept state on before get to go through in a hurry
add check-state

# Throw away RFC 1918 networks
add drop all from 10.0.0.0/8 to any in via fxp0
add drop all from 172.16.0.0/12 to any in via fxp0
add drop all from 192.168.0.0/16 to any in via fxp0

# Allow the bridge machine to say anything it wants
# (if the machine is IP-less do not include these rows)
add pass tcp from 1.2.3.4 to any setup keep-state
add pass udp from 1.2.3.4 to any keep-state
add pass ip from 1.2.3.4 to any

# Allow the inside hosts to say anything they want
add pass tcp from any to any in via xl0 setup keep-state
add pass udp from any to any in via xl0 keep-state
add pass ip from any to any in via xl0

# TCP section
# Allow SSH
add pass tcp from any to any 22 in via fxp0 setup keep-state
# Allow SMTP only towards the mail server
add pass tcp from any to relay 25 in via fxp0 setup keep-state
# Allow zone transfers only by the secondary name server [dns2.nic.it]
add pass tcp from 193.205.245.8 to ns 53 in via fxp0 setup keep-state
# Pass ident probes.  It is better than waiting for them to timeout
add pass tcp from any to any 113 in via fxp0 setup keep-state
# Pass the "quarantine" range
add pass tcp from any to any 49152-65535 in via fxp0 setup keep-state

# UDP section
# Allow DNS only towards the name server
add pass udp from any to ns 53 in via fxp0 keep-state
# Pass the "quarantine" range
add pass udp from any to any 49152-65535 in via fxp0 keep-state

# ICMP section
```

```
# Pass 'ping'
add pass icmp from any to any icmptypes 8 keep-state
# Pass error messages generated by 'traceroute'
add pass icmp from any to any icmptypes 3
add pass icmp from any to any icmptypes 11

# Everything else is suspect
add drop log all from any to any
```

Aqueles que já configuraram firewalls antes podem notar que algumas coisas estão faltando. Em particular, não há regras anti-spoofing, na verdade, *não* adicionamos:

```
add deny all from 1.2.3.4/8 to any in via fxp0
```

Isso significa descartar pacotes que estão chegando de fora e alegam ser da nossa rede. Isso é algo que normalmente se faz para garantir que alguém não tente evadir o filtro de pacotes, gerando pacotes maliciosos que parecem ser provenientes de dentro da rede. O problema é que há *pelo menos* um host na interface externa que você não deseja ignorar: o roteador. Mas geralmente, o ISP faz o anti-spoofing em seu roteador, então não precisamos nos preocupar tanto com isso.

A última regra parece ser uma duplicata exata da regra padrão, ou seja, não deixa passar nada que não seja especificamente permitido. Mas há uma diferença: todo tráfego suspeito será registrado.

Existem duas regras para permitir o tráfego SMTP e DNS direcionado ao servidor de e-mail e servidor de nomes, se você tiver esses servidores. Obviamente, todo o conjunto de regras deve ser personalizado de acordo com as preferências pessoais, este é apenas um exemplo específico (o formato da regra é descrito com precisão na página do manual ipfw(8)). Observe que, para o "relay" e o "ns" funcionarem, as consultas ao serviço de nomes devem funcionar *antes* de habilitar a ponte. Isto é um exemplo para garantir que você configure o IP no cartão de rede correto. Alternativamente, é possível especificar o endereço IP em vez do nome do host (obrigatório se a máquina não possuir um IP).

Pessoas acostumadas a configurar firewalls provavelmente também estão acostumadas a ter uma regra de `reset` ou `forward` para pacotes ident (porta TCP 113). Infelizmente, essa não é uma opção aplicável com a bridge, então a melhor coisa a fazer é simplesmente passá-los para o destino. Contanto que a máquina de destino não esteja executando um ident daemon, isso é relativamente inofensivo. A alternativa é descartar conexões na porta 113, o que cria alguns problemas com serviços como IRC (a sondagem ident deve expirar).

A única outra coisa um pouco estranha que você pode ter notado é que há uma regra para permitir que a máquina da bridge fale, e outra para os hosts internos. Lembre-se de que isso ocorre porque os dois conjuntos de tráfego seguirão caminhos diferentes através do kernel e do filtro de pacotes. A rede interna passará pela ponte, enquanto a máquina local usará o stack de IP normal para falar. Portando, existem duas regras para lidar com os casos diferentes. As regras `in via fxp0` funcionam para ambos os caminhos. Em geral, se você usar regras `in via` em todo o filtro, precisará fazer uma exceção para pacotes gerados localmente, porque eles não entraram por nenhuma de nossas interfaces.

6. Colaboradores

Muitas partes deste artigo foram retiradas, atualizadas e adaptadas de um antigo texto sobre bridges, editado por Nick Sayer. E um par de inspirações vieram de uma introdução sobre bridges de autoria de Steve Peterson.

Um grande obrigado ao Luigi Rizzo pela implementação do código de ponte (bridge) no FreeBSD e pelo tempo que ele dedicou a mim respondendo a todas as minhas perguntas relacionadas.

Agradeço também a Tom Rhodes, que olhou para o meu trabalho de tradução do italiano (a língua original deste artigo) para o inglês.