Implementando um firewall de camada 2 (bridge)

Ponte levadiça

Os firewalls são, tipicamente, implementados como roteadores. Entretanto, as coisas não precisam ser assim. Filtros de pacotes baseados no princípio de *bridging* possuem algumas vantagens, entre elas a de poderem ser inseridos em qualquer ponto de sua rede sem absolutamente alterar a topologia e a configuração de seus nós. POR RALF SPENNEBERG

O Linux alcançou uma respeitável reputação de excelente plataforma para firewalls. O kernel possui um eficientíssimo filtro de pacotes, o subsistema netfilter/iptables. Em um firewall tradicional, o netfilter é colocado para funcionar em uma máquina que serve de roteadora, onde divide a rede em duas ou mais subredes. Entretanto, adicionar um firewall em uma rede já estabelecida pode envolver uma drástica alteração na infra-estrutura do cabeamento e do próprio projeto lógico. A dor de cabeça pode ser tamanha que talvez seja necessário mudar todos os endereços IP das estações e até impor mais restrições no acesso aos serviços internos.

Em vez dessa complicação toda, que tal instalar uma *bridge*? São muito mais simples de operar e não causam nenhum impacto numa rede já existente. As bridges (em português, "pontes") são dispositivos que trabalham na camada 2 do modelo de referência OSI e normalmente inspecionam os endereços MAC (ou seja, os endereços físicos das placas de rede) em vez do endereço IP – consulte o quadro "Construindo Pontes". O Linux pode tirar partido dessa capacidade para criar firewalls "transparentes". Obviamente, a bridge não deixará de inspecionar pacotes dos protocolos de camadas mais altas (endereços IP, portas TCP e UDP) em sua tarefa de firewall. A grande vantagem é que os computadores da rede não vão sequer notar que há um firewall no caminho. Para eles, há uma ligação direta com a Internet, mas alguma "entidade espiritual" bloqueia o envio de pacotes ilegais.

Configuração do kernel

Lennert Buytenhek e Bart de Schuymer escreveram um patch que adiciona um modo de *bridging firewall* ao kernel 2.4. Se você usa o kernel 2.6 nem precisa aplicar nada: o recurso já existe, basta apenas configurar o núcleo do sistema de acordo (ver figuras 1 e 2).

Todas as opções de bridging no grupo *netfilter* são importantes. Por exemplo, o suporte a tabelas ARP (ARP Tables) nos módulos *IP_NF_ARPTBLES*, *IP_NF_ARPFILTER* e *IP_NF_ARP_MANGLE*. Essas funções podem ser implementadas como módulos ou compiladas de forma monolítica no kernel. A opção *Physdev match* também é importante; o módulo chama-se *IP_NF_MATCH_PHYSDEV*. Ela é necessária para que as séries 2.6 e superiores possam consultar a interface física (camada 2) e, assim, aplicar as regras de filtragem nos pacotes que passam por elas.

Depois de compilar sem erros (e colocar para rodar) o novo kernel, são necessárias mais algumas ferramentas no espaço do usuário para que possamos montar nosso bridgewall. Embora muitas distribuições já tenham, por padrão, o programa *iptables* disponível, é bem provável que seja necessário instalar os utilitários *arptables* e *ebtables* na maioria delas. Se você está rodando uma versão atual do kernel 2.6 em uma distribuição mais antiga (que vinha

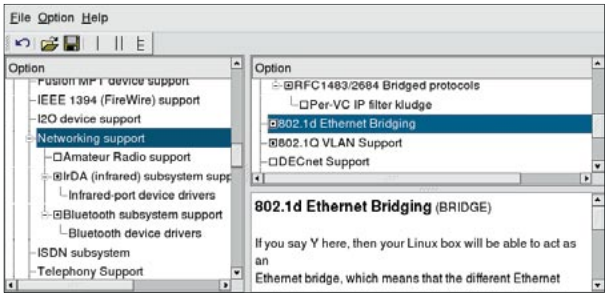

Figura 1: Para ativar o modo bridge no kernel 2.6, marque a opção *802.1d Ethernet Bridging* no grupo *Networking Support*.

com o kernel 2.4, por exemplo) também será necessário atualizar o pacote com o utilitário *iptables*.

O pacote bridge-utils [2] também deve ser instalado. Ele será usado para configurar a bridge. As distribuições modernas normalmente disponibilizam esse pacote por padrão. Nele encontramos o comando *brctl*, de uso exclusivo do usuário root. Se o emitirmos assim:

```
brctl addbr br0
```

criamos uma bridge chamada *br0*. Já o comando a seguir:

```
ip link show br0
```

confirma que a bridge existe. Como ela possui um nome, pode-se até rodar diversas bridges virtuais em uma única máquina com Linux!

A seguir, a bridge precisa saber qual placa de rede Ethernet deve administrar. O comando *brctl* permite adicionar interfaces à bridge:

```
brctl addif br0 eth0
brctl addif br0 eth1
```

As placas de rede não devem estar configuradas neste ponto; ou seja, não devem estar ativas (*UP*) ou possuir qualquer endereço IP. A idéia é ativá-las só depois de estarem associadas à nossa bridge.

```
ip link set eth0 up
ip link set eth1 up
```

Construindo Pontes

O termo *bridge* (ponte) refere-se a uma categoria de dispositivos que manipulam e redirecionam pacotes de rede na camada 2 do modelo OSI. Além das próprias bridges em si, muitos equipamentos populares, presentes em quase todas as redes do planeta, são um tipo de bridge. Um deles é o manjadíssimo *switch* (em português, *comutador*, embora infelizmente o termo em inglês seja mais usado). Para redirecionar pacotes na camada 2, a bridge precisa relacionar todos os endereços MAC da rede local e lembrar-se quais computadores estão ligados em cada uma de suas saídas, chamadas de *portas*. Quando a bridge recebe um pacote de um endereço MAC já conhecido, simplesmente redireciona o pacote para a saída correta, reduzindo bastante o tráfego nas outras portas. Se a bridge não souber para onde deve enviar o pacote, faz um *broadcast* (difusão) e o envia a todas as suas portas.

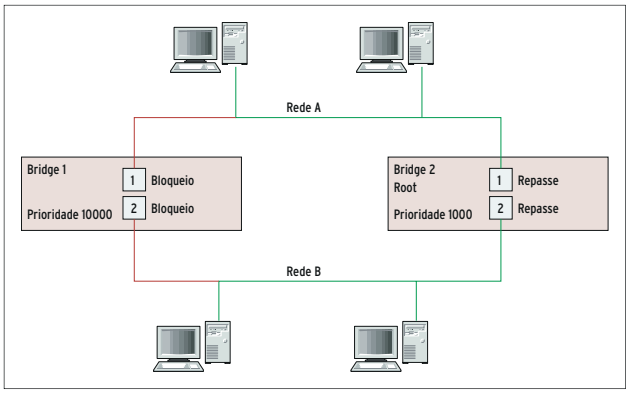

Figura 4: As bridges 1 e 2 conectam as redes A e B. A bridge de menor prioridade usa o protocolo STP para definir os caminhos válidos para pacotes enviados entre A e B.

O comando *brctl showmacs br0* lista todos os endereços MAC que a bridge já conhece. O resultado é mostrado em forma de tabela. A primeira coluna contém o número da porta para a qual o pacote deve ser enviado, pois o computador de destino está conectado a ela. A segunda coluna contém o endereço MAC desse computador. As demais colunas mostram outros dados sobre cada nó da rede.

Memória Fraca

Para ficar sempre atualizada, a bridge remove os endereços MAC mais antigos da tabela ARP. É possível especificar quanto tempo um endereço MAC que não esteja em uso pode ficar na tabela antes que a bridge o descarte. O comando apropriado é *brctl setageingtime br0 tempo_em_segundos*.

O esforço computacional interno seria desnecessariamente alto se a bridge descartasse imediatamente endereços MAC inativos. Em vez disso, a bridge primeiro marca os endereços ociosos e os remove posteriormente a intervalos regulares. O processo é conhecido como *garbage collection* (numa tradução aproximada, "coleta de lixo"). Esse intervalo pode ser ajustado com o comando *brctl setgcint br0 tempo_em_segundos*. O valor padrão é 0 (zero).

Spanning Tree Protocol

Switches modernos usam o *spanning tree protocol* (protocolo de árvore distribuída ou STP) para oferecer uma configuração de alta disponibilidade. Com esse protocolo, dois ou mais switches conectam subredes entre si e, dessa forma, descobrem todas as rotas possíveis entre elas. Depois de eleito, um switch mestre (ou "raiz") define os caminhos ativos e inativos para acesso a cada uma das subredes e propagam essa informação a todos os switches envolvidos.

Os switches restantes bloqueiam os caminhos inativos e suas interfaces, impedindo assim que pacotes duplicados (ou seja, um mesmo pacote que tomou dois caminhos diferentes) cheguem até a rede de destino (ver figura 4). Se um switch apresenta algum mau funcionamento, os switches remanescentes descobrem quais caminhos alternativos ainda estão disponíveis e contornam o equipamento defeituoso.

O Linux suporta o protocolo STP; entretanto, o administrador precisa emitir o comando *brctl stp br0 on* para ativá-lo. Um valor de prioridade entre 0 e 65535 pode ser associado à bridge: *brctl setbridgeprio br0 prioridade*. A bridge com o menor número de prioridade (portanto, com a prioridade mais alta) assume a função de "raiz".

1, 2, 3: testando...

As bridges verificam se suas colegas estão "vivas" mandando mensagens de teste ("hello") a intervalos regulares. Os administradores podem determinar o intervalo de envio dessas mensagens com o comando *brctl sethello br0 tempo_em_segundos*. o comando *brctl setmaxage br0 tempo_em_segundos* especifica quanto tempo as outras bridges devem esperar para receber a mensagem de "hello" das companheiras. Como um grande formigueiro, todo o sistema de bridges assume que a bridge que não responder depois desse intervalo está "morta".

Quando uma bridge está conectada a uma rede, precisa esperar um período de tempo arbitrário antes de começar a redirecionar pacotes. Essa espera é necessária para verificar se a rede está usando o protocolo STP. O comando *brctl setfd br0 tempo_em_segundos* ajusta o tempo de espera.

Apesar da explicação acima, o protocolo STP deve estar *completamente desativado* em uma bridge que vá atuar como firewall por filtragem de pacotes: *brctl stp br0 off*. O firewall tem que se basear em seu próprio conjunto de regras e não pode, sob hipótese alguma, ser desabilitado por *spoofing* de STP.

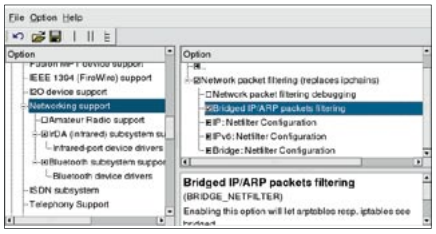

Figura 2: Não esqueça de marcar a opção *Bridged IP/ARP packets filtering* na configuração do netfilter dentro do kernel!

```
ip link set br0 up
```

Nossa bridge está, a partir de agora, pronta para entrar em ação! Veja as informações obtidas com o comando *ip link show* (figura 3). A bridge redireciona os pacotes e mantém sua própria tabela ARP, usando esse cache para rastrear quais endereços MAC estão conectados a (isto é, podem ser acessados por) cada interface (veja mais detalhes no quadro "Construindo Pontes", na página ao lado).

Bridgewalling

Como qualquer firewall, a bridge deve possuir um conjunto de regras que defina quais pacotes podem passar e quais devem ser bloqueados. Há três comandos para criação dessas regras: *iptables*, *ebtables* e *arptables*.

Todos os pacotes que a bridge redireciona (entram por uma porta e saem por outra) passam pela cadeia *FORWARD* do netfilter. Há algumas coisas a observar quando usamos o comando *iptables* em uma bridge. Se uma regra especifica que os pacotes devem atravessar a bridge em apenas uma direção, deve-se usar a opção *-m physdev* (tabela 1). Isso permite que a política de conexão identifique a porta da bridge pela qual o pacote entrou, ou mesmo se ele chegou a ser manipulado por ela.

O exemplo a seguir permite conexões SSH ao endereço IP 192.168.0.16 (porta TCP/22) em apenas uma direção. O servidor SSH está conectado à placa de rede *eth1*. As conexões só podem ser estabelecidas por clientes conectados à *eth0*. Precisamos de dois comandos *iptables* para definir a regra:

```
iptables -A FORWARD -m physdev
  --physdev-in eth0 ➥
  --physdev-out eth1 ➥
  --dport 22 -d 192.168.0.16➥
   -m state ➥
  --state NEW -j ACCEPT
iptables -A FORWARD -m physdev ➥
  --physdev-is-bridged -m state ➥
  --state ESTABLISHED,RELATED ➥
  -j ACCEPT
```

O primeiro comando cuida do estabelecimento da conexão, que deve ocorrer em apenas uma direção. O segundo comando permite que os pacotes que pertencem a essa conexão já estabelecida possam passar pelo firewall.

Coleta de endereços

Os firewalls baseados em bridges normalmente são usados para melhorar a segurança de redes existentes e já bem grandinhas. Com eles, não existe a necessidade de mudar a estrutura da rede e, muito menos, os endereços IP. Um dos pontos em que um firewall desse tipo pode ser bastante útil é na frente de um roteador que o administrador não possa alterar ou que não possua funcionalidade de firewall. Mas os

Figura 3: A bridge virtual já está no ar! O comando *ip link show* mostra os dados da bridge após o *4: br0:*.

bridgewalls são realmente magistrais quando usados para dividir grandes redes em áreas menores, com menos tráfego e gerenciamento melhor.

Nesse caso, os endereços IP de ambos os lados da bridge não podem ser alterados para pertencer a classes ou subredes diferentes, como seria o caso de um firewall comum baseado em roteador. Portanto, para que possamos filtrar o tráfego baseado nos números IP precisamos recorrer a um truque. O comando *ipset* dá aos administradores um meio de subdividir a rede, criando um banco de endereços no qual se podem coletar IPs arbitrários. Os comandos a seguir criam um banco chamado *left* e adicionam três endereços IP ao banco:

```
ipset -F; ipset -X; ipset -N ➥
left iphash
ipset -A left 192.168.0.5
ipset -A left 192.168.0.17
ipset -A left 192.168.0.18
```

O novo banco pode ser usado nominalmente em regras do iptables, com a opção *-m set*. O comando *--set nome_do_banco* indica o banco a ser considerado:

```
iptables -A FORWARD -m physdev ➥
--physdev-in eth0 ➥
--physdev-out eth1 ➥
--dport 22 -m set ➥
--set left -m state ➥
--state NEW -j ACCEPT
```

Tabela 1: Regras com *physdev*

Opção	Significado
--physdev-in *Nome*	Especifica a porta pela qual o pacote precisa entrar na bridge para que esta regra se aplique a ele.
--physdev-out *Nome*	Especifica a porta pela qual o pacote precisa sair da bridge para que esta regra se aplique a ele.
--physdev-is-in	O pacote veio de uma interface diretamente conectada à bridge.
--physdev-is-out	O pacote deixará o computador por uma interface diretamente conectada a uma bridge.
--physdev-is-bridged	O pacote trafega por dentro da bridge.

Listagem 1: MAC-NAT

```
ebtables -t nat -A PREROUTING -p ➥
ARP --arp-ip-dst  -j arpreply ➥
--arpreply-mac 0:ff:90:2b:a6:16
ebtables -t nat -A PREROUTING ➥
-p IPv4 -d 0:ff:90:2b:a6:16 ➥
--ip-dst 192.168.0.16 -j ➥
dnat --to-dst fe:fd:0:0:0:1 ➥
--dnat-target ACCEPT
ebtables -t nat -A POSTROUTING ➥
-p IPv4 -s fe:fd:0:0:0:1 -j ➥
snat --to-src 0:ff:90:2b:a6:16 ➥
--snat-target ACCEPT
```

Além dos pacotes IP, os pacotes ARP são úteis em regras de firewall. Muitos ataques originados dentro da própria rede são baseados em requisições e respostas ARP forjadas (*ARP spoofing*).

Tabelas ARP

O comando *arptables* pode filtrar pacotes ARP. Afora operações de *bridging*, o comando só deve ser usado nas cadeias *INPUT* e *OUTPUT*, pois os roteadores não encaminham pacotes ARP de uma interface a outra. Entretanto, em uma bridge, é possível filtrar pacotes ARP também na cadeia *FORWARD*. A sintaxe do comando é similar à do *iptables*. O *arptables* usa os alvos *ACCEPT* e *DROP*; *REJECT* não faz nenhum sentido em se tratando de pacotes ARP.

```
arptables -A FORWARD -s ! ➥
192.168.0.15 --destination-mac ➥
fe:fd:00:00:00:01 -j DROP
```

O comando acima descarta todos os pacotes ARP de resposta enviados ao computador cujo endereço MAC é *fe:fd:00:00:00:01* e que não foram originados pelo computador cujo IP é 192.168.0.15. As respostas ARP informam à entidade requisitante qual o endereço MAC do computador cujo IP foi especificado na pergunta. Aqui, o computador com o endereço MAC *fe:fd:00:00:00:01*, que vive em uma rede do outro lado da nossa bridge, só consegue enxergar o endereço MAC do computador cujo IP é 192.168.0.15.

Tabelas Ethernet

O comando *ebtables* é muito mais poderoso, permitindo que se faça coisas fabulosas como, por exemplo, NAT de endereços MAC! Com NAT, a bridge pode evitar que invasores descubram endereços MAC de computadores conectados a outras portas. A bridge manda seu próprio MAC como resposta a uma requisição ARP e, a partir daí, faz a tradução MAC-NAT para todos os pacotes IP que a atravessarem. O primeiro comando da listagem 1 diz à bridge para responder com seu próprio MAC (00:ff:90:2B:A6:16) a qualquer requisição ARP destinada a descobrir o endereço MAC do IP 192.168.0.16.

O endereço IP do computador que se quer esconder atrás da bridge precisa ser escrito atrás da opção *--arp-ip-dst*. A opção *--arpreply-mac* é o endereço MAC da bridge. Para MAC-NAT de pacotes IP, pode-se precisar também dos comandos nas linhas 2 e 3 da Listagem 1. Em nosso exemplo, 192.168.0.16 é o endereço IP do computador a ser escondido por detrás da bridge; e seu endereço MAC é *fe:fd:00:00:00:01*.

A documentação no site oficial do ebtables [3] fornece mais informações sobre as capacidades deste comando.

Nas profundezas da rede

A técnica de *bridgewalling* dá aos administradores uma nova classe de filtros de pacotes que abre imensas possibilidades de controle sobre protocolos de camada 2. Mas a coisa mais espetacular em relação ao bridging é a possibilidade de inserir um firewall transparente entre dois pontos de uma rede pré-existente. A bridge simplesmente substitui o hub, o switch ou mesmo um cabo *cross-over*. E se for necessário bloquear alguns computadores suspeitos, é possível fazê-lo sem ter que reprojetar toda a numeração IP de sua rede. Basta colocar em operação um *bridgewall*. ■

INFORMAÇÕES

[1] iptables: *http://www.iptables.org*
[2] Linux bridge: *http://bridge.sf.net*
[3] ebtables: *http://ebtables.sf.net*