Filtering Bridges

Alex Dupre

ale@FreeBSD.org

$FreeBSD: doc/it_IT.ISO8859-15/articles/filtering-bridges/article.sgml,v 1.14 2008/01/27 15:29:47 ale Exp $

Spesso è utile dividere una rete fisica (come una Ethernet) in due segmenti separati, senza dover creare sottoreti e usare un router per collegarli assieme. Il dispositivo che collega due reti insieme in questo modo è chiamato bridge. Un sistema FreeBSD con due interfacce di rete è sufficiente per raggiungere lo scopo.

Un bridge funziona individuando gli indirizzi del livello MAC (indirizzi Ethernet) dei dispositivi collegati ad ognuna delle sue interfacce di rete e inoltrando il traffico tra le due reti solo se il mittente e il destinatario si trovano su segmenti differenti. Sotto molti punti di vista un brigde è simile a uno switch Ethernet con solo due porte.

1 Perché usare un filtering bridge?

Sempre più frequentemente, grazie all'abbassamento dei costi delle connessioni a banda larga (xDSL) e a causa della riduzione del numero di indirizzi IPv4 disponibili, molte società si ritrovano collegate ad Internet 24 ore su 24 e con un numero esiguo (a volte nemmeno una potenza di 2) di indirizzi IP. In situazioni come queste spesso è desiderabile avere un firewall che regoli i permessi di ingresso e uscita per il traffico da e verso Internet, ma una soluzione basata sulle funzionalità di packet filtering dei router può non essere applicabile, vuoi per problemi di suddivisione delle sottoreti, vuoi perché il router è di proprietà del fornitore di accesso (ISP), vuoi perché il router non supporta tali funzionalità. È in questi casi che l'utilizzo di un filtering bridge diventa altamente consigliato.

Un firewall basato su bridge può essere configurato e inserito direttamente tra il router xDSL e il vostro hub/switch Ethernet senza alcun problema di assegnazione di indirizzi IP.

> **Nota:** La traduzione italiana di "firewall" è "muro anti incendio", *non* "muro di fuoco" come molti pensano. Nel corso dell'articolo, comunque, manterrò i termini tecnici nella loro lingua originale in modo da non creare confusione o ambiguità.

2 Metodi d'installazione

Aggiungere le funzionalità di bridge a una macchina FreeBSD non è difficile. Dalla release 4.5 è possibile caricare tali funzionalità come moduli anziché dover ricompilare il kernel, semplificando di gran lunga la procedura. Nelle prossime sottosezioni spiegherò entrambi i metodi di installazione.

> **Importante:** *Non* seguite entrambe le istruzioni: le procedure sono *a esclusione*. Scegliete l'alternativa che meglio si adatta alle vostre esigenze e capacità.

Prima di continuare è necessario assicurarsi di avere almeno due schede di rete Ethernet che supportino la modalità promiscua sia in ricezione che in trasmissione, difatti devono essere in grado di inviare pacchetti Ethernet con qualunque indirizzo, non solo il loro. Inoltre, per avere un buon rendimento, le schede dovrebbero essere di tipo PCI bus mastering. Le scelte migliori sono ancora le Intel EtherExpress™ Pro seguite dalle 3Com® 3c9xx subito dopo. Per comodità nella configurazione del firewall può essere utile avere due schede di marche differenti (che usino drivers differenti) in modo da distinguere chiaramente quale interfaccia sia collegata al router e quale alla rete interna.

2.1 Configurazione del Kernel

Così avete deciso di utilizzare il più vecchio e collaudato metodo di installazione. Per prima cosa bisogna aggiungere le seguenti righe al file di configurazione del kernel:

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

La prima riga serve a compilare il supporto per il bridge, la seconda il firewall e la terza le funzioni di logging del firewall.

Adesso è necessario compilare e installare il nuovo kernel. Si possono trovare le istruzioni nella sezione Building and Installing a Custom Kernel (http://www.FreeBSD.org/doc/it_IT.ISO8859-15/books/handbook/kernelconfig-building.html) dell'handbook.

2.2 Caricamento dei Moduli

Se avete deciso di usare il nuovo e più semplice metodo di installazione, l'unica cosa da fare è aggiungere la seguente riga al file `/boot/loader.conf`:

```
bridge_load="YES"
```

In questo modo all'avvio della macchina verrà caricato insieme al kernel anche il modulo `bridge.ko`. Non è necessario invece aggiungere una riga per il modulo `ipfw.ko` in quanto verrà caricato in automatico dallo script `/etc/rc.network` dopo aver seguito i passi della prossima sezione.

3 Preparativi finali

Prima di riavviare per caricare il nuovo kernel o i moduli richiesti (a seconda del metodo che avete scelto in precedenza), bisogna effettuare alcune modifiche al file `/etc/rc.conf`. La regola di default del firewall è di

rifiutare tutti i pacchetti IP. Per iniziare attiviamo il firewall in modalità `open`, in modo da verificare il suo funzionamento senza alcun problema di filtraggio pacchetti (nel caso stiate eseguendo questa procedura da remoto, tale accorgimento vi consentirà di non rimanere erroneamente tagliati fuori dalla rete). Inserite queste linee nel file `/etc/rc.conf`:

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

La prima riga serve ad attivare il firewall (e a caricare il modulo `ipfw.ko` nel caso non fosse già compilato nel kernel), la seconda a impostarlo in modalità `open` (come descritto nel file `/etc/rc.firewall`), la terza a non visualizzare il caricamento delle regole e la quarta ad abilitare il supporto per il logging.

Per quanto riguarda la configurazione delle interfacce di rete, il metodo più utilizzato è quello di assegnare un IP a solo una delle schede di rete, ma il bridge funziona egualmente anche se entrambe o nessuna delle interfacce ha IP settati. In quest'ultimo caso (IP-less) la macchina bridge sarà ancora più nascosta in quanto inaccessibile dalla rete: per configurarla occorrerà quindi entrare da console o tramite una terza interfaccia di rete separata dal bridge. A volte all'avvio della macchina qualche programma richiede di accedere alla rete, per esempio per una risoluzione di dominio: in questo caso è necessario assegnare un IP all'interfaccia esterna (quella collegata a Internet, dove risiede il server DNS), visto che il bridge verrà attivato alla fine della procedura di avvio. Questo vuol dire che l'interfaccia `fxp0` (nel nostro caso) deve essere menzionata nella sezione ifconfig del file `/etc/rc.conf`, mentre la `xl0` no. Assegnare IP a entrambe le schede di rete non ha molto senso, a meno che durante la procedura di avvio non si debba accedere a servizi presenti su entrambi i segmenti Ethernet.

C'è un'altra cosa importante da sapere. Quando si utilizza IP sopra Ethernet ci sono due protocolli Ethernet in uso: uno è IP, l'altro è ARP. ARP permette la conversione dell'indirizzo IP di una macchina nel suo indirizzo Ethernet (livello MAC). Affinché due macchine separate dal bridge riescano a comunicare tra loro è necessario che il bridge lasci passare i pacchetti ARP. Tale protocollo non fa parte del livello IP, visto che è presente solo con IP sopra Ethernet. Il firewall di FreeBSD agisce esclusivamente sul livello IP e quindi tutti i pacchetti non IP (compreso ARP) verranno inoltrati senza essere filtrati, anche se il firewall è configurato per non lasciar passare nulla.

Ora è arrivato il momento di riavviare la macchina e usarla come in precedenza: appariranno dei nuovi messaggi riguardo al bridge e al firewall, ma il bridge non sarà attivato e il firewall, essendo in modalità `open`, non impedirà nessuna operazione.

Se ci dovessero essere dei problemi, è il caso di scoprire ora da cosa derivino e risolverli prima di continuare.

4 Attivazione del Bridge

A questo punto, per attivare il bridge, bisogna eseguire i seguenti comandi (avendo l'accortezza di sostituire i nomi delle due interfacce di rete `fxp0` e `xl0` con i propri):

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

La prima riga specifica tra quali interfacce va attivato il bridge, la seconda abilita il firewall sul bridge ed infine la terza attiva il bridge.

> **Nota:** Se hai FreeBSD 5.1-RELEASE o precedenti le variabili sysctl sono chiamate in modo differente. Guarda bridge(4) per i dettagli.

A questo punto dovrebbe essere possibile inserire la macchina tra due gruppi di host senza che venga compromessa qualsiasi possibilità di comunicazione tra di loro. Se è così, il prossimo passo è quello di aggiungere le parti `net.link.ether.bridge.`*`[blah]=[blah]`* di queste righe al file `/etc/sysctl.conf`, in modo che vengano eseguite all'avvio della macchina.

5 Configurazione del Firewall

Ora è arrivato il momento di creare il proprio file con le regole per il firewall, in modo da rendere sicura la rete interna. Ci sono delle complicazioni nel fare questo, perché non tutte le funzionalità del firewall sono disponibili sui pacchetti inoltrati dal bridge. Inoltre, c'è una differenza tra i pacchetti che stanno per essere inoltrati dal bridge e quelli indirizzati alla macchina locale. In generale, i pacchetti che entrano nel bridge vengono processati dal firewall solo una volta, non due come al solito; infatti vengono filtrati solo in ingresso, quindi qualsiasi regola che usi `out` oppure `xmit` non verrà mai eseguita. Personalmente uso `in via` che è una sintassi più antica, ma che ha un senso quando la si legge. Un'altra limitazione è che si possono usare solo i comandi `pass` e `drop` per i pacchetti filtrati dal bridge. Cose avanzate come `divert`, `forward` o `reject` non sono disponibili. Queste opzioni possono ancora essere usate, ma solo per il traffico da e verso la macchina bridge stessa (sempre che le sia stato assegnato un IP).

Nuovo in FreeBSD 4.0 è il concetto di stateful filtering. Questo è un grande miglioramento per il traffico UDP, che consiste tipicamente di una richiesta in uscita, seguita a breve termine da una risposta con la stessa coppia di indirizzi IP e numeri di porta (ma con mittente e destinatario invertiti, ovviamente). Per i firewall che non supportano il mantenimento di stato, non c'è modo di gestire questo breve scambio di dati come una sessione unica. Ma con un firewall che può "ricordarsi" di un pacchetto UDP in uscita e permette una risposta nei minuti successivi, gestire i servizi UDP è semplice. L'esempio seguente mostra come fare. La stessa cosa è possibile farla con i pacchetti TCP. Questo permette di evitare qualche tipo di attacco denial of service e altri sporchi trucchi, ma tipicamente fa anche crescere velocemente la tabella di stato.

Vediamo un esempio di configurazione. Bisogna notare che all'inizio del file `/etc/rc.firewall` ci sono già delle regole standard per l'interfaccia di loopback `lo0`, quindi non ce ne occuperemo più ora. Le regole personalizzate andrebbero messe in un file a parte (per esempio `/etc/rc.firewall.local`) e caricate all'avvio modificando la riga del file `/etc/rc.conf` dove era stata definita la modalità `open` con:

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

> **Importante:** Bisogna specificare il path *completo* del file, altrimenti non verrà caricato con il rischio di rimanere tagliati fuori dalla rete.

Per il nostro esempio immaginiamo di avere l'interfaccia `fxp0` collegata all'esterno (Internet) e la `x10` verso l'interno (LAN). La macchina bridge ha assegnato l'IP `1.2.3.4` (è impossibile che il vostro ISP vi assegni un indirizzo simile a questo, ma per l'esempio va bene).

```
# Le connessioni di cui abbiamo mantenuto lo stato vengono fatte passare subito
add check-state
```

```
# Esclude le reti locali definite nell'RFC 1918
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

# Permette alla macchina bridge di connettersi con chi vuole
# (se la macchina è IP-less non includere questi comandi)
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

# Permette agli host della rete interna di connettersi con chi vogliono
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

# Sezione TCP
# Permette SSH
add pass tcp from any to any 22 in via fxp0 setup keep-state
# Permette SMTP solo verso il mail server
add pass tcp from any to relay 25 in via fxp0 setup keep-state
# Permette i trasferimenti di zona solo dal name server secondario [dns2.nic.it]
add pass tcp from 193.205.245.8 to ns 53 in via fxp0 setup keep-state
# Lascia passare i controlli ident:
# è meglio che aspettare che vadano in timeout
add pass tcp from any to any 113 in via fxp0 setup keep-state
# Permette connessioni nel range di "quarantena".
add pass tcp from any to any 49152-65535 in via fxp0 setup keep-state

# Sezione UDP
# Permette DNS solo verso il name server
add pass udp from any to ns 53 in via fxp0 keep-state
# Permette connessioni nel range di "quarantena".
add pass udp from any to any 49152-65535 in via fxp0 keep-state

# Sezione ICMP
# Abilita le funzioni di 'ping'
add pass icmp from any to any icmptypes 8 keep-state
# Permette il passaggio dei messaggi di errori del comando 'traceroute'
add pass icmp from any to any icmptypes 3
add pass icmp from any to any icmptypes 11

# Tutto il resto è sospetto
add drop log all from any to any
```

Coloro che hanno configurato un firewall in precedenza potrebbero aver notato che manca qualcosa. In particolare, non ci sono regole contro lo spoofing, difatti *non* abbiamo aggiunto:

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

Ovvero, non far entrare dall'esterno pacchetti che affermano di venire dalla rete interna. Questa è una cosa che solitamente viene fatta per essere sicuri che qualcuno non provi a eludere il packet filter, generando falsi pacchetti che sembrano venire dall'interno. Il problema è che c'è *almeno* un host sull'interfaccia esterna che non si può

ignorare: il router. Solitamente, però, gli ISP eseguono il controllo anti-spoof sui loro router e quindi non ce ne dobbiamo preoccupare.

L'ultima riga sembra un duplicato della regola di default, ovvero non far passare nulla che non sia stato specificatamente permesso. In verità c'è una differenza: tutto il traffico sospetto verrà loggato.

Ci sono due regole per permettere il traffico SMTP e DNS verso il mail server e il name server, se ne avete. Ovviamente l'intero set di regole deve essere personalizzato per le proprie esigenze, questo non è altro che uno specifico esempio (il formato delle regole è spiegato dettagliatamente nella man page ipfw(8)). Bisogna notare che, affinché "relay" e "ns" siano interpretati correttamente, la risoluzione dei nomi deve funzionare *prima* che il bridge sia attivato. Questo è un chiaro esempio che dimostra l'importanza di settare l'IP sulla corretta scheda di rete. In alternativa è possibile specificare direttamente l'indirizzo IP anziché il nome host (cosa necessaria se la macchina è IP-less).

Le persone che sono solite configurare un firewall probabilmente avranno sempre usato una regola `reset` o `forward` per i pacchetti ident (porta 113 TCP). Sfortunatamente, questa non è una scelta applicabile con il bridge, quindi la cosa migliore è lasciarli passare fino alla destinazione. Finché la macchina di destinazione non ha un demone ident attivo, questa tecnica è relativamente sicura. L'alternativa è proibire le connessioni sulla porta 113, creando qualche problema con servizi tipo IRC (le richieste ident devono andare in timeout).

L'unica altra cosa un po' particolare che potete aver notato è che c'è una regola per lasciar comunicare la macchina bridge e un'altra per gli host della rete interna. Ricordate che questo è dovuto al fatto che i due tipi di traffico prendono percorsi differenti attraverso il kernel e di conseguenza anche dentro il packet filter. La rete interna passerà attraverso il bridge, mentre la macchina locale userà il normale stack IP per le connessioni. Perciò due regole per gestire due casi differenti. Le regole `in via fxp0` funzionano in entrambi i casi. In generale, se usate regole `in via` attraverso il packet filter, dovrete fare un'eccezione per i pacchetti generati localmente, in quanto non entrano tramite nessuna interfaccia.

6 Contributi

Alcune parti di questo articolo sono state prese, tradotte e adattate da testi sui bridge, appartenenti alla documentazione di FreeBSD in lingua inglese, a cura di Nick Sayer e Steve Peterson.

Un grosso ringraziamento va a Luigi Rizzo per l'implementazione delle funzionalità di bridging in FreeBSD e per il tempo che mi ha dedicato rispondendo ad alcune mie domande a riguardo.