Cortafuegos con Dialup en FreeBSD

Marc Silver

marcs@draenor.org

$FreeBSD: doc/es_ES.ISO8859-1/articles/dialup-firewall/article.sgml,v 1.3 2007/11/08 21:49:10 carvay Exp $

En éste artículo se describe cómo configurar un cortafuegos que utiliza conexión PPP con FreeBSD e IPFW y más concretamente el uso de un cortafuegos en una conexión telefónica a la que se le asigna una IP dinámica. Éste documento no se ocupa de la configuración de la conexión PPP necesaria.

Traducción de José Vicente Carrasco Vayá <carvay@es.FreeBSD.org>.

1. Prefacio

Uso de un Cortafuegos en una Conexión Telefónica en FreeBSD

En éste documento se expone el proceso necesario para configurar un cortafuegos en FreeBSD cuando la dirección IP es asignada dinámicamente por el ISP. Aunque se ha hecho todo lo posible por hacer éste documento tan informativo y correcto como sea posible puede enviar comentarios y/o sugerencias al autor a <marcs@draenor.org>, que serán bien recibidas.

2. Configuración del Kernel

Lo primero que tendrá que hacer es recompilar su kernel. Si necesita más información sobre cómo hacerlo el mejor recurso es la sección de Handbook acerca de la configuración del kernel (../../books/handbook/kernelconfig.html). Necesitará añadir a su fichero de configuración del kernel las siguientes opciones:

`options IPFIREWALL`

Activa el código necesario para el cortafuegos en el kernel.

`options IPFW2`

Activa la nueva versión de IPFW.

> **Importante:** Esto solo debe hacerse en FreeBSD 4.X puesto que en las versiones más recientes vienen incluído por defecto.

`options IPFIREWALL_VERBOSE`

Envia los paquetes que se ha decidido sean incluídos en un `log` a la aplicación encargada de gestionar los `logs` del sistema.

`options IPFIREWALL_VERBOSE_LIMIT=`*`100`*

Limita el número de veces que una entrada que cumple las reglas puede ser incluída en los `logs` del sistema. Esto previene que sus `logs` se vean inundados por entradas repetidas. *`100`* es un número razonable, pero puede ajustarlo a sus necesidades.

`options IPDIVERT`

Activa los "sockets" *divert*, que serán descritos más tarde.

Hay otras entradas *opcionales* que pueden compilarse en el kernel para incrementar la seguridad. No hacen falta para el funcionamiento del cortafuegos pero algunos usuarios especialmente paranoicos pueden querer usarlos.

`options TCP_DROP_SYNFIN`

Ignorar paquetes TCP con SYN y FIN. Esto evita ser vulnerable al uso de herramientas como `security/nmap`, que permiten identificar la pila TCP/IP de la máquina, pero incumple el soporte a las extensiones incluídas en el RFC1644. *No* se recomienda hacer tal cosa si la máquina va a ejecutar un servidor web.

No reinicie tras recompilar el kernel. Si todo sale bien sólo necesitaremos reiniciar una vez en todo el proceso de instalación del cortafuegos.

3. Modificación de `/etc/rc.conf` para cargar el cortafuegos

Necesitamos hacer algunos cambios en `/etc/rc.conf` para darle ciertos detalles del cortafuegos. Es tan simple como añadir las siguientes líneas:

```
firewall_enable="YES"
firewall_script="/etc/firewall/fwrules"
natd_enable="YES"
natd_interface="tun0"
natd_flags="-dynamic"
```

Para más información sobre éstas entradas consulte `/etc/defaults/rc.conf` y lea rc.conf(5)

4. Desactivación de la Traducción de Direcciones de Red (NAT) de PPP

Es posible que ya esté usando la NAT que incluye PPP. Si es su caso tendrá que desactivarla puesto que los casos que vamos a usar emplean natd(8) para hacerlo.

Si ya dispone de un grupo de entradas para arrancar automáticamente PPP probablemente se parezca a ésto:

```
ppp_enable="YES"
ppp_mode="auto"
ppp_nat="YES"
ppp_profile="profile"
```

Si es su caso, tendrá que desactivar específicamente `ppp_nat` asegurándose de que `ppp_nat="NO"` existe en su in `/etc/rc.conf`. Tendrá tambié que borrar todas las entradas como `nat enable yes` o `alias enable yes` en `/etc/ppp/ppp.conf`.

5. Las Reglas del Cortafuegos

Casi hemos acabado. Lo único que nos falta es definir las reglas del cortafuegos, reiniciar y deberíamos tener nuestro cortafuegos funcionando perfectamente. Soy consciente de que cada cual tendrá necesidades distintas como reglas básicas. He intentado escribir unas reglas básicas que puedan cubrir las necesidades de un usuario de conexión telefónica normal. Vamos a comenzar por lo básico de un cortafuegos cerrado. Lo que se busca es rechazar todo por defecto y dejar pasar solamente lo que necesitemos. Las reglas deberían ir en la forma "al principio permitir, luego rechazar". La premisa es que vamos a an~adir reglas para lo que vamos a aceptar y luego rechazamos todo lo demás. :)

Ahora vamos a crear el directorio `/etc/firewall`. Sitúese en el directorio y edite el fichero `fwrules` tal y como hemos escrito dentro de `rc.conf`. Por favor, no olvide que puede cambiar el nombre del fichero por cualquier otro que prefiera. Éste documento solamente facilita un ejemplo del nombre del fichero.

Vamos a echar un vistazo a un ejemplo de fichero de configuración del cortafuegos que hemos comentado cuidadosamente.

```
# Definimos el comando con el que invocamos al cortafuegos
# (tal y como hemos incluído en /etc/rc.firewall) para
# facilitarnos la lectura.
fwcmd="/sbin/ipfw"

# Fuerza el borrado de todas las reglas existentes en nuestro
# cortafuegos antes de cargar el contenido de éste fichero.
$fwcmd -f flush

# Desvía todos los paquetes a través del interfaz tunnel.
$fwcmd add divert natd all from any to any via tun0

# Permite todas las conexiones incluídas en reglas
# dinámicas pero rechaza todas aquellas conexiones
# establecidas que no estén incluídas en alguna
# regla dinámica.
$fwcmd add check-state
$fwcmd add deny tcp from any to any established

# Aceptar todas las conexiones de localhost.
$fwcmd add allow tcp from me to any out via lo0 setup keep-state
$fwcmd add deny  tcp from me to any out via lo0
$fwcmd add allow ip  from me to any out via lo0 keep-state

# Aceptar todas las conexiones desde mi tarjeta de red que yo inicie.
$fwcmd add allow tcp from me to any out xmit any setup keep-state
$fwcmd add deny  tcp from me to any
$fwcmd add allow ip from me to any out xmit any keep-state

# Todo el mundo a lo largo y ancho de Internet puede conectarse
```

```
# a los siguientes servicios de la máquina.  Éste
# ejemplo permite específicamente las conexiones a sshd
# y al servidor web.
$fwcmd add allow tcp from any to me dst-port 22,80 in recv any setup keep-state

# Ésto envía un RESET a todos los paquetes ident.
$fwcmd add reset log tcp from any to me 113 in recv any

# Activa ICMP: borre el tipo 8 si no quiere que su máquina
# responda al ping.
$fwcmd add allow icmp from any to any icmptypes 0,3,8,11,12,13,14

# Rechazamos todo lo demás.
$fwcmd add deny log ip from any to any
```

Ya tiene usted un cortafuegos totalmente funcional que acepta todas las conexiones a los puertos 22 y 80 y registrará cualquier otro tipo de intento de conexión en un fichero `log`. Ahora podemos reiniciar tranquilamente y su cortafuegos debería empezar a trabajar tal y como le hemos dicho. Si le parece que hay algún dato incorrecto o tiene alguna sugerencia para mejorar éste documento por favor envíeme un correo electrónico.

6. Preguntas

1. ¿Por qué utiliza natd(8) y ipfw(8) cuando podría usar los filtros incluídos en ppp(8)?

Seré honesto y diré que no hay una razón clara por la que use `ipfw` y `natd` en lugar de los filtros que incorpora `ppp`. Tras hablarlo con mucha gente el consenso parece ser que `ipfw` es mucho más potente y configurable que el filtrado de `ppp` pero lo que se gana en funcionalidad lo pierde en facilidad de personalización. Una de las razones por las que prefiero usar esas aplicaciones es que creo má conveniente ejecutar un cortafuegos desde el kernel que desde una aplicación de entorno de usuario.

2. Me aparecen mensajes como "`limit 100 reached on entry 2800`" y después de eso ya no me aparecen más entradas indicando tráfico rechazado en mis `logs`. ¿Funciona mi cortafuegos?

Esto significa simplemente que se ha alcanzado el máximo de entradas que pueden incluírse en el `log` cuando una determinada regla se ha cumplido. Esa regla sigue funcionando pero no enviaría más entradas al `log` hasta que el contador vuelva a cero. Puede poner a cero ese contador mediante `ipfw resetlog`. Además es posible elevar el límite de entradas a introducir en el `log` incluyendo la "option" `IPFIREWALL_VERBOSE_LIMIT` en el fichero de de configuración del kernel. Por otra parte puede modificar ese valor (sin modificar su kernel y en consecuencia sin reiniciar la máquina) mediante el valor de sysctl(8) net.inet.ip.fw.verbose_limit.

3. Supongamos que estoy usando direcciones privadas internas, por ejemplo el rango 192.168.0.0. ¿Puedo añadir una regla al cortafuegos mediante un comando como `$fwcmd add deny all from any to 192.168.0.0:255.255.0.0 via tun0` para prevenir intentos de acceso desde el exterior para conectar con máquinas de mi red?.

La respuesta corta es no. La razón es que `natd` efectúa la traducción de direcciones para *cualquier cosa* que sea redirigida a través del dispositivo `tun0`. Eso significa que todos los paquetes entrantes hablarán exclusivamente con la IP asignada dinámicamente y *no* con la red interna. Hay que tener en cuenta sin embargo que es posible añadir una regla como `$fwcmd add deny all from 192.168.0.4:255.255.0.0 to any via tun0`, que evitaría que una de las máquinas de esa red enviara tráfico al exterior a través del cortafuegos.

4. Debo de haber hecho algo mal. He seguido las instrucciones al pie de la letra y no tengo acceso a Internet.

Estamos asumiendo que está usando *userland-ppp*, en consecuencia el conjunto de reglas que aquí se proponen operan en el interfaz `tun0`, que corresponde a la primera conexión efectuada mediante ppp(8) (más conocido como *user-ppp*). Las conexiones efectuadas más tarde recibirán nombres como `tun1`, `tun2` y así sucesivamente.

Hay que tener también presente que pppd(8) en cambio utiliza el interfaz `ppp0`, de modo que si se inicia la conexión con pppd(8) hay que sustituír `tun0` por `ppp0`. A continuación se muestra una forma muy limpia de modificar las reglas del cortafuegos. Conservaremos un fichero con las reglas originales con el nombre de `fwrules_tun0`.

```
% cd /etc/firewall
/etc/firewall% su
Password:
/etc/firewall# mv fwrules fwrules_tun0
/etc/firewall# cat fwrules_tun0 | sed s/tun0/ppp0/g > fwrules
```

Para saber exactamente si está usando ppp(8) o pppd(8) examine la salida de ifconfig(8) una vez que establezca la conexión. V.g., en una conexión hecha mediante pppd(8) debería encontrarse con algo muy similar a lo siguiente (se muestran sólo las líneas relevantes):

```
% ifconfig
(eliminado...)
ppp0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1524
                inet xxx.xxx.xxx.xxx --> xxx.xxx.xxx.xxx netmask 0xff000000
(eliminado...)
```

Si por el contrario la conexión fué establecida mediante ppp(8) (*user-ppp*) ésto es más o menos lo que se encontraría:

```
% ifconfig
(eliminado...)
ppp0: flags=8010<POINTOPOINT,MULTICAST> mtu 1500
(skipped...)
tun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1524
        (IPv6 stuff skipped...)
                inet xxx.xxx.xxx.xxx --> xxx.xxx.xxx.xxx netmask 0xffffff00
                Opened by PID xxxxx
        (eliminado...)
```