Implementación de una DMZ

Reunion CUDI
Primavera 2006

Jorge A. Zárate Pérez / Mario Farias Elinos

DMZ

¿Que es un Firewall ?

Hablando de manera generica, un firewall es un método de protección de servidores o redes, que estan conectados a otros servidores o redes, existen diferentes maneras de “colocar” un firewall en la red, de manera muy concreta un firewall es un packet filter (filtrado de paquetes) entre los dispositivos anteriores, este filtrado de paquetes puede ser en varios niveles del modelo OSI.

El firewall define un conjunto de reglas y politicas para filtrar los paquetes que pasan por el.

¿Que no es un firewall ?

No es una bala de plata que proveea una protección total a la red o servidores.

Introducción

Hardware vs Software

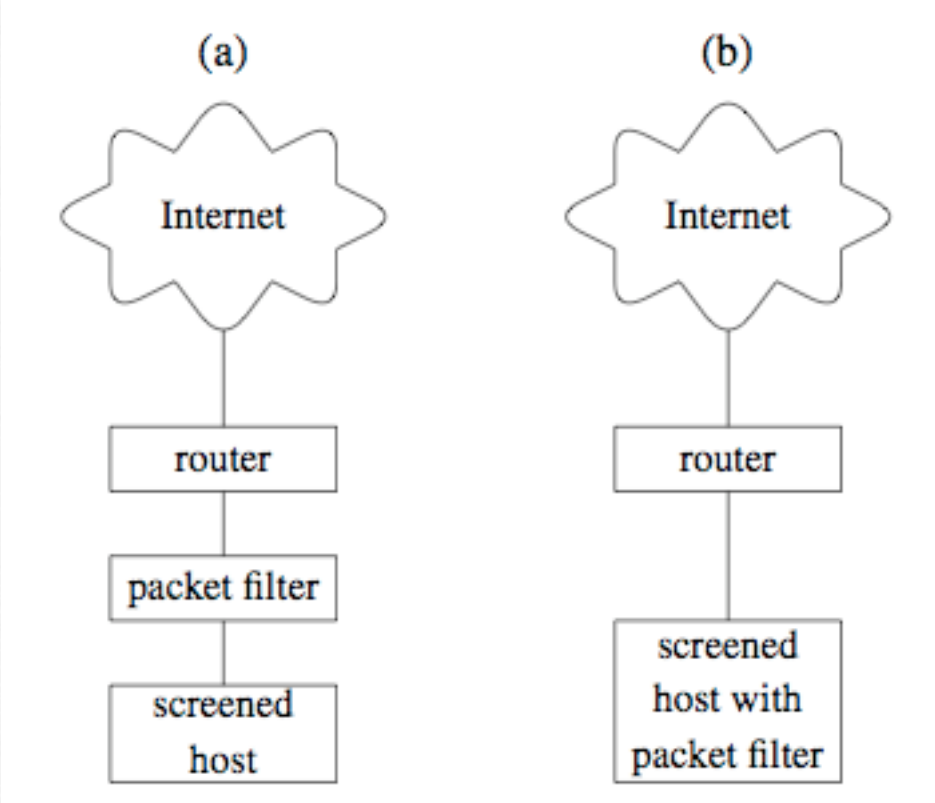
Muchas veces hablamos de firewall entre un firewall basado en software y los que estan en hardware, la diferencia si se analiza es que los firewalls que estan basados en hardware tiene un “software” en la EPROM... y por la naturaleza cambiante de las reglas de estos no es viable tener un circuito virtual fijo, la diferencia mas bien esta en la “caja” que viene con un manual y un conjunto de cursos para operarlo, y un respaldo de soporte “incondicional”.

¿Como son ?

Los sabores que podemos tener de firewalls de manera general serian los siguientes:

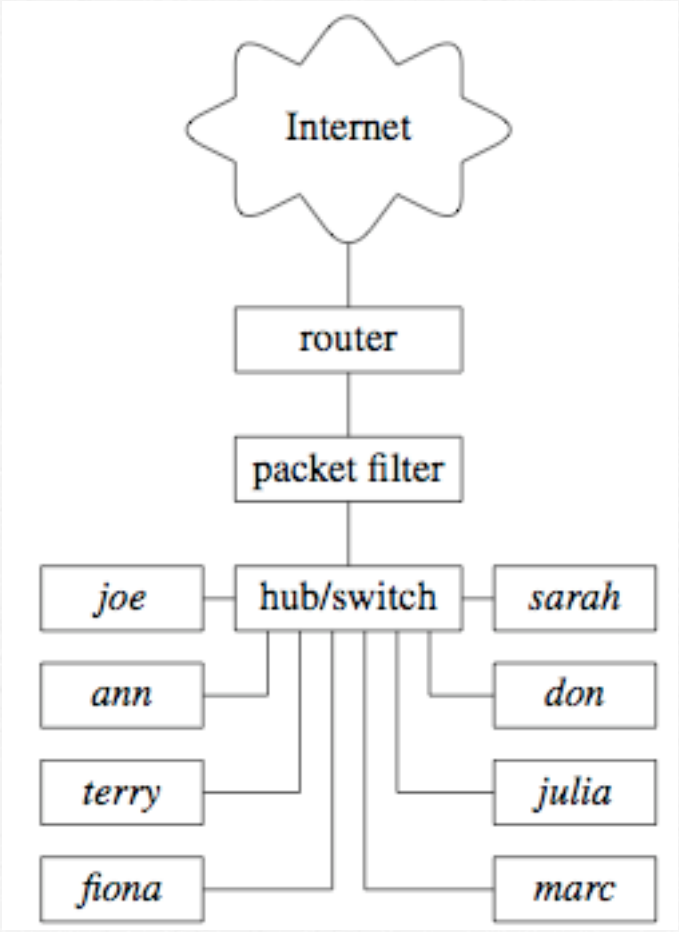
Screened Host:

Es un equipo que proteje una maquina de ataques externos, através de un filtrado de paquetes. Este filtrado es básico, en esencia no acepta paquetes del exterior que no sean respuesta de una petición de una máquina interna.



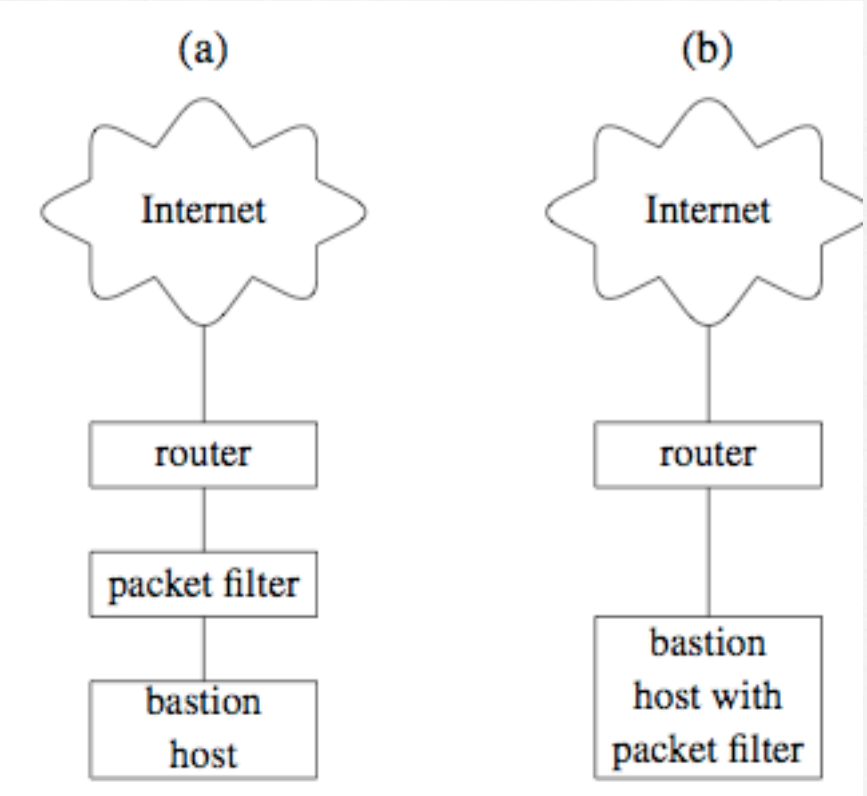
Screened LAN:

Es similar al anterior, excepto que no protejemos una sola maquina, si no dos o mas equipos (una red), este modelo no ofrece seguridad si el ataque es generado desde el interior a un equipo de la misma red.



Bastion Host:

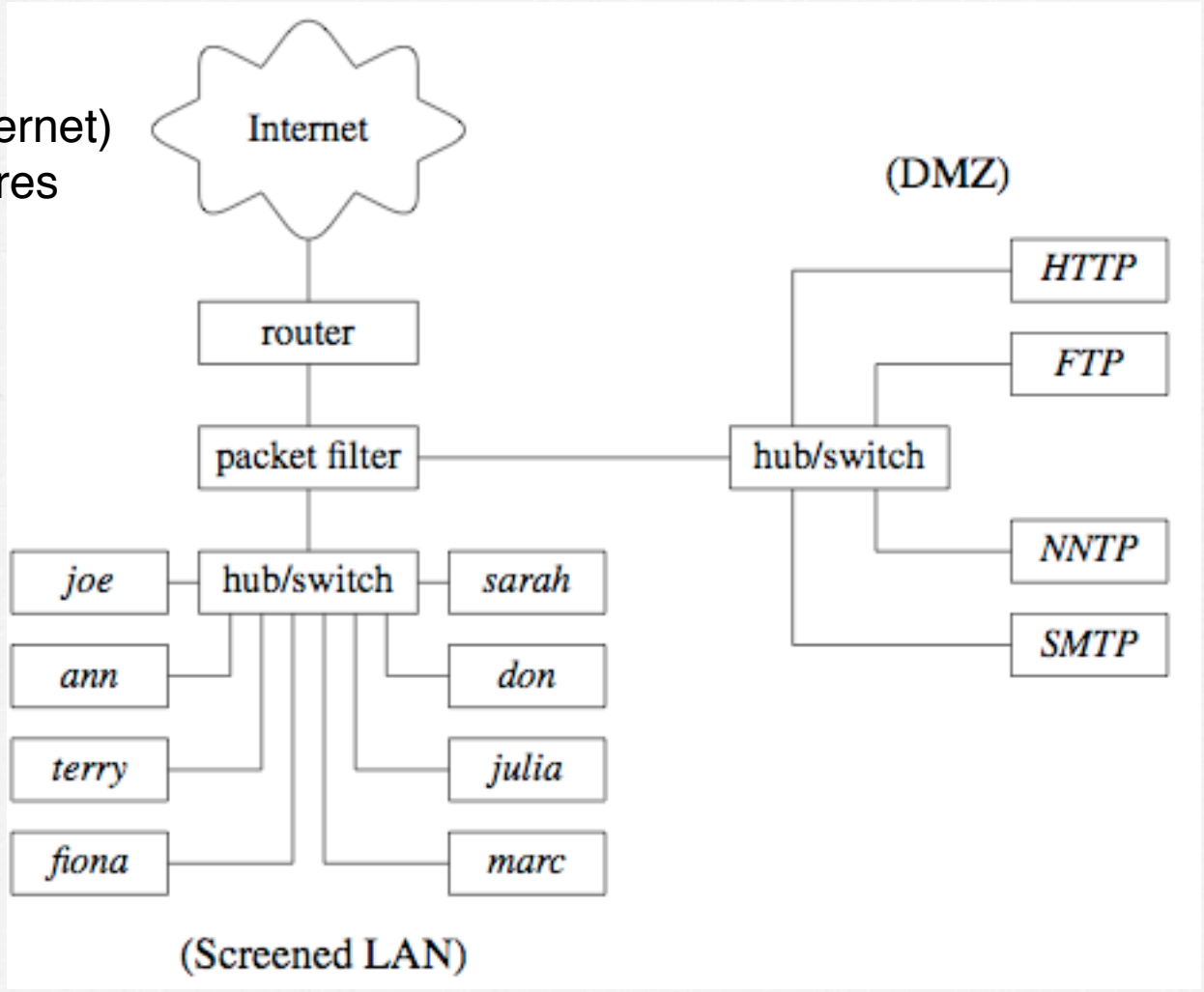
Este tiene características del screened host, con la diferencia que se permiten accesos desde afuera a ciertos servicios (SMTP, HTTP, DNS....) en la red interna.



Demilitarized Zone (DMZ)

Este diseño parecido al anterior, con la diferencia que tiene al menos 2 segmentos internos (zonas), las cuales se conectan por lo regular de la siguiente forma:

1.- Interface a Area Pública (internet)
2.- Interface a Zona de Servidores
3.- Interface a Usuarios



Las Características que nos ofrecerá nuesto DMZ serán:

- Filtrado de paquetes a cualquier zona

- NAT, Mapero Bidireccional

- Colas de tráfico y Prioridad

- Salidas redundantes / balanceo de carga

- Balanceo de carga a servicios

- Filtrado de contenido (web-cache)

- Monitoreo de tráfico en interfaces via netflow

Que necesitamos?

Hardware
- Una PC (Preferentemente servidor)
  HD 20 GB, P2 300 Mhz, 64 MB RAM, 3 NICs 10/100.
- 2 Switches

Software
- OpenBSD 3.8 http://www.openbsd.org/
- Squid cache http://www.squid-cache.org/
- Softflowd http://www2.mindrot.org/files/softflowd/

NOTA:
Se recomienda diseñar y probar el modelo en una maqueta antes de poner en producción

Filtrado de Paquetes

Nos permite definir la seguridad perimetral, el primer bloque de segmentos, área de usuarios, internet y servidores (DMZ). Para eso utilizaremos el Packet Filter (al que en este documento nos referiremos como PF) es el sistema de OpenBSD para filtrar el tráfico TCP/IP y llevar a cabo la Traducción de Direcciones de Red (a la que nos referiremos como NAT, Network Address Translation). Además de estas funciones, PF también es capaz de normalizar y acondicionar el tráfico TCP/IP y de proveer control del ancho de banda y la priorización de paquetes TCP/IP. PF ha formado parte del núcleo GENERIC del sistema OpenBSD desde la versión 3.0 del sistema.

Para activar PF y que lea archivo de configuración correspondiente durante el arranque, hay que modificar el archivo **/etc/rc.conf** de tal modo que la línea de PF quede como sigue:

```
pf=YES
```

También se puede activar y desactivar PF con el programa pfctl(8), mediante los siguientes comandos:

```
# pfctl -e
# pfctl -d
```

Configuración

PF lee las reglas de configuración desde el archivo /etc/pf.conf durante el arranque; este archivo se encargan de cargarlo los guiones de ejecución rc de inicio del sistema.

El archivo pf.conf consta de siete partes:

Macros: *Variables definidas por el usuario que pueden contener direcciones IP, nombres de interfaces, etc.*
Tablas: *Una estructura que se utiliza para contener listas de direcciones IP.*
Opciones: *Varias opciones para el control del funcionamiento de PF.*
Normalización (Scrub): *Reprocesamiento de paquetes para su normalización y desfragmentación.*
Formación de Colas: *Provee control del ancho de banda y priorización de paquetes.*
Traducción de Direcciones: *Controla la Traducción de Direcciones de Red (NAT) y el redireccionamiento de paquetes.*
Reglas de Filtrado: *Permite el filtrado selectivo o el bloqueo de paquetes según van pasando a través de cualquiera de las interfaces de red.*

PF ignorará las líneas en blanco, y no interpretará las líneas que empiecen con #, tratándolas como simples comentarios.

Control

Después del arranque del sistema operativo, se puede gestionar la operación de PF usando el programa pfctl. Algunos comandos de ejemplo son:

```
# pfctl -f /etc/pf.conf     Carga el archivo pf.conf
# pfctl -nf /etc/pf.conf    Analiza el archivo, pero no lo carga
# pfctl -Nf /etc/pf.conf    Carga sólo las reglas de NAT del archivo
# pfctl -Rf /etc/pf.conf    Carga sólo las reglas de filtrado del archivo

# pfctl -sn                 Muestra las reglas en vigor de NAT
# pfctl -sr                 Muestra las reglas en vigor de filtrado
# pfctl -ss                 Muestra la tabla de estado en vigor
# pfctl -si                 Muestra estadísticas y contadores del filtrado
# pfctl -sa                 Muestra TODO lo que puede
```

Listas

Una lista permite especificar múltiples criterios similares dentro de una misma regla, como puedan ser múltiples protocolos, números de puertos, direcciones, etc. De este modo, en lugar de escribir una regla de filtrado para cada dirección IP que haya que bloquear, se puede escribir una sola regla e indicar las direcciones IP en una lista aparte. Las listas se definen especificando los componentes dentro de { } llaves.

Así, cuando pfctl(8) está cargando el grupo de reglas y se encuentra con una lista, crea automáticamente múltiples reglas, una para cada uno de los componentes. Por ejemplo:

```
block out on fxp0 from { 192.168.0.1, 10.5.32.6 } to any
```

sería una contracción de:

```
block out on fxp0 from 192.168.0.1 to any
block out on fxp0 from 10.5.32.6 to any
```

Además, se pueden incluir varias listas dentro de una misma regla:

```
block out on fxp0 proto { tcp udp } from \
{ 192.168.0.1, 10.5.32.6 } to any port { ssh telnet }
```

Macros

Las macros son variables definidas por el usuario que pueden contener direcciones IP, números de puertos, nombres de interfaces, etc. Las macros reducen la complejidad de un grupo de reglas de PF, y también facilitan mucho el mantenimiento de un grupo reglas.

Los nombres de las macros deben empezar con una letra y pueden contener letras, dígitos y guiones bajos. Los nombres de las macros no pueden ser palabras reservadas (como pass, out, o queue).

```
ext_if = “fxp0”
```

block in on $ext_if from any to any

o bien se pueden combinar macros

```
host2 = “192.168.1.2”
all_hosts = “{“ $host1 $host2 “}”
```

Ahora, la macro $all_hosts se expandiría a 192.168.1.1, 192.168.1.2.

Filtrado de Paquetes

La acción de filtrar paquetes es bloquear o permitir el paso a los paquetes de datos de forma selectiva, según van llegando a una interfaz de red. Los criterios que usa pf para inspeccionar los paquetes los toma de la información existente en la capa 'Layer 3' (IPv4 y IPv6) y en la capa 'Layer 4' (TCP, UDP, ICMP, y ICMPv6) de las cabeceras de los paquetes. Los criterios que más se utilizan son los de la dirección de origen y de destino, el puerto de origen y de destino, y el protocolo.

Las reglas de filtrado especifican los criterios con los que debe concordar un paquete y la acción a seguir, bien sea bloquearlo o permitir que pase, que se toma cuando se encuentra una concordancia. Las reglas de filtrado se evalúan por orden de secuencia, de la primera a la última. A menos que el paquete concuerde con una regla que contenga la clave *quick*, se evaluará el paquete comparándolo con todas las reglas de filtrado antes de decidir una acción final. La última regla que concuerde será la «ganadora» y la que dictamine qué acción se tomará con el paquete. Al principio del grupo de reglas de filtrado hay un *pass all* implícito que indica que si algún paquete no concuerda con ninguna de las reglas de filtrado, la acción a seguir será pass, o sea permitirle el paso.

Sintaxis de las Reglas

La sintaxis general, muy simplificada, para las reglas de filtrado es:

```
action direction [log] [quick] on interfaz [af] [proto protocol] \
   from src_addr [port src_port] to dst_addr [port dst_port] \
   [tcp_flags] [state]
```

action
La acción a seguir para los paquetes que concuerden, ya sea pass o block. La acción pass permitirá el paso al paquete de vuelta hasta el núcleo del sistema, para que éste lo procese, mientras que la acción block actuará según se indique en la configuración de la opción de la política de bloqueo, block-policy. La acción predeterminada se puede anular especificando block drop (bloquear y eliminar el paquete) o block return (bloquear y devolver el paquete).

direction
La dirección en la que se mueve el paquete en una interfaz, que será in (entrante) o out (saliente).

Sintaxis de las Reglas (3)

protocol

El protocolo de la capa ‘Layer 4’ del paquete:
tcp
udp
icmp
icmp6

Un nombre de protocolo válido del archivo /etc/protocols
Un número de protocolo entre 0 y 255
Un grupo de protocolos que usen una lista.

src_addr, dst_addr
La dirección de origen y/o de destino en la cabecera IP. Las direcciones se pueden especificar como:
Una sola dirección IPv4 ó IPv6.
Un bloque de red CIDR.
Un «Nombre de Dominio Totalmente Cualificado» (FQDN) que se resolverá por el «Servicio de Nombres de Dominio» cuando se carguen las reglas. Todas las direcciones IP resultantes se sustituirán dentro de la regla.

Sintaxis de las Reglas (4)

src_port, dst_port

El puerto de origen y/o de destino en la capa ‘Layer 4’ de la cabecera IP. Los puertos se pueden especificar como:

Un número entre el 1 y el 65535
Un nombre de servicio válido del archivo /etc/services
Un grupo de puertos que usen una lista
Un indicador de campo:
!= (no es igual que)
< (menor que)
> (mayor que)
<= (menor o igual que)
>= (mayor o igual que)
>< (intervalo)
<> (intervalo inverso)
: (intervalo inclusivo)
El operador de intervalo inclusivo también es un operador binario e incluye el argumento en el intervalo.

Sintaxis de las Reglas (5)

tcp_flags
Especifica los indicadores que deben existir en la cabecera TCP cuando se usa proto tcp. Los indicadores se especifican como flags check/mask. Por ejemplo, flags S/SA instruye a PF para que sólo mire los indicadores S y A (SYN y ACK), y que acepte la concordancia si el indicador SYN está activo (“on”).

state
Especifica si se guarda la información sobre el estado en paquetes que concuerden con esta regla.

keep state - funciona con TCP, UDP y ICMP.
modulate state - sólo funciona con TCP. PF generará «Números de Secuencia Inicial» (ISNs, Initial Sequence Numbers) consistentes para los paquetes que concuerden con esta regla.

synproxy state - hace de proxy para las conexiones TCP entrantes con el fin de ayudar a proteger a los servidores de desbordamientos TCP SYN falsificados. Esta opción incluye las funcionalidades keep state y modulate state.

Denegación Predeterminada

La práctica recomendada para configurar un firewall es la de tomar una aproximación de «denegación predeterminada»; es decir, denegar el paso a todo y a partir de ahí ir permitiendo el paso a través del firewall de forma selectiva a cierto tráfico. Esta aproximación es la recomendada ya que los posibles fallos se cometerían a favor de la seguridad, y también por que hace más fácil la creación de grupos de reglas.

Para crear una política de filtrado de denegación predeterminada, las primeras dos reglas deben ser:

```
block in  all
block out all
```

Con esto se bloquea todo el tráfico en todas las interfaces en cualquier dirección, y desde cualquier origen, hasta cualquier destino.

Paso de Tráfico

Ahora hay que permitir de forma explícita y selectiva el paso del tráfico a través del firewall, o de lo contrario será bloqueado por la política de denegación predeterminada. Aquí es donde entran en juego los criterios del paquete, como son el puerto de origen/destino, la dirección de origen/destino, y el protocolo. Siempre que se permita el paso de cierto tráfico a través del firewall hay que escribir las reglas de un modo tan restrictivo como sea posible. Esto es para asegurarse de que sólo pasará el tráfico que se permita, y ningún otro.

```
# Permitir el paso al tráfico entrante en la interfaz dc0 de la red local,
# 192.168.0.0/24, hacia la dirección IP 192.168.0.1 de la máquina de OpenBSD.
# También permitir el paso al tráfico saliente que es enviado de vuelta en dc0.
pass in  on dc0 from 192.168.0.0/24 to 192.168.0.1
pass out on dc0 from 192.168.0.1 to 192.168.0.0/24

# Permitir el paso al tráfico entrante TCP en la interfaz fxp0 del servidor
# de web que se encuentra en la máquina de OpenBSD. El nombre de la
# interfaz, fxp0, se usa como la dirección de destino para que los paquetes
# sólo concuerden con esta regla si tienen como destino la máquina de OpenBSD.
pass in on fxp0 proto tcp from any to fxp0 port www
```

La Clave quick

Como se ha indicado anteriormente, cada paquete se evalúa con el grupo de reglas de filtrado, desde la primera hasta la última. El resultado predeterminado es el de marcar el paquete para que se le permita el paso; esto puede cambiar con cualquiera de las reglas por las que pasa, y podría cambiar varias veces antes de llegar al final de las reglas de filtrado. La última regla con la que concuerde marcará el resultado. Existe una excepción para esto: la opción quick en una regla de filtrado tiene el efecto de cancelar el procesamiento de cualquier regla consiguiente, y provoca que se ejecute la acción especificada sin revisar las siguientes reglas.

Mal:

```
block in on fxp0 proto tcp from any to any port ssh
pass  in all
```

En este caso, la línea block puede ser evaluada, pero nunca tendrá ningún efecto, ya que va seguida por una línea que permite el paso de todo.

Mejor:

```
block in quick on fxp0 proto tcp from any to any port ssh
pass  in all
```

Estas reglas se evalúan de una forma algo diferente. Si un paquete concuerda con la línea block, debido a la naturaleza de la opción quick, se bloqueará el paso a dicho paquete y se ignorará el resto del grupo de reglas.

Mantenimiento del Estado

Una de las funcionalidades importantes de PF es la del «mantenimiento del estado» (keeping state) o «inspección completa del estado» (stateful inspection). La inspección del estado se refiere a la capacidad de PF de llevar un seguimiento del estado, o del progreso, de una conexión de red. Almacenando información sobre cada conexión en una tabla de estado, PF puede determinar rápidamente si un paquete que está pasando a través del firewall pertenece a una conexión ya establecida. Si es así, se le permite pasar a través del firewall sin tener que pasar a través de la evaluación del grupo de reglas.

El mantenimiento del estado tiene muchas ventajas, entre otras que los grupos de reglas son más simples y se obtiene un rendimiento más alto del filtrado de paquetes. PF puede puede hacer que los paquetes que vayan en cualquier dirección concuerden con entradas en la tabla de estado, lo que quiere decir que no es necesario escribir reglas de filtrado que permitan el paso del tráfico de vuelta. Y, como los paquetes que concuerdan con conexiones stateful no pasan a través de la evaluación del grupo de reglas, el tiempo que tarda PF en procesarlos puede reducirse considerablemente.

Mantenimiento del Estado (2)

Cuando una regla tiene la opción keep state, el primer paquete que concuerda con ella crea un estado entre el remitente y el destinatario. A partir de ahí, los paquetes que vayan desde el remitente hacia el destinatario no serán los únicos que concuerden con la entrada de estado y que circunvalen la evaluación de las reglas, sino que también lo harán los paquetes de respuesta desde el destinatario hacia el remitente. Por ejemplo:

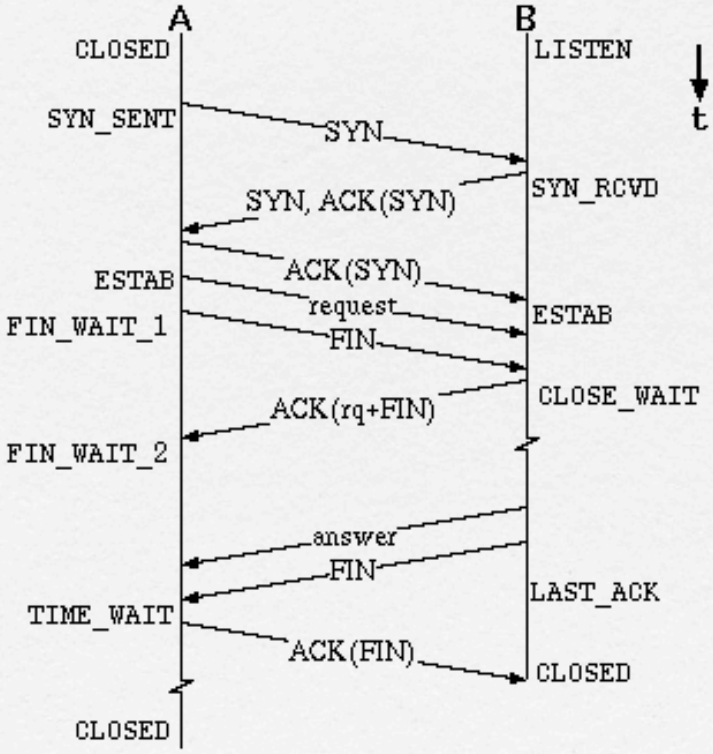
pass out on fxp0 proto tcp from any to any keep state

Esta regla permite el paso de cualquier tráfico TCP saliente en la interfaz fxp0, y también permite que el tráfico de respuesta pase de vuelta a través del firewall. El mantenimiento del estado es una funcionalidad muy útil, y su uso mejora de forma significativa el rendimiento del firewall, ya que las búsquedas de estados son mucho más rápidas que la evaluación de un paquete a través de todas las reglas de filtrado.

Indicadores de TCP

La concordancia de paquetes TCP basada en indicadores es algo que se suele usar para filtrar paquetes TCP que estén intentando abrir una nueva conexión. Aquí se puede ver una lista de indicadores TCP y sus significados:

S : SYN - Synchronize; indica un requerimiento para iniciar la sesión
F : FIN - Finish; final de la sesión
R : RST - Reset; abandonar una conexión
P : PUSH - Push; el paquete es enviado inmediatamente
A : ACK - Acknowledgement
U : URG - Urgent
E : ECE - Explicit Congestion Notification Echo
W : CWR - Congestion Window Reduced



Bloqueo de Paquetes Falsificados (Spoofed Packets)

La falsificación de direcciones (spoofing) es cuando un usuario con malas intenciones falsifica la dirección IP de origen en los paquetes que se transmiten, con el objetivo de esconder su dirección real o de suplantar otro nodo en la red. Una vez que el usuario ha falsificado su dirección, puede lanzar un ataque a nivel de red sin revelar la dirección real de origen del ataque, o intentar obtener acceso a servicios de la red que estén restringidos para ciertas direcciones IP.

PF ofrece cierto nivel de protección contra la falsificación de direcciones mediante la clave antispoof:

```
antispoof [log] [quick] for interface [af]
```

antispoof for fxp0 inet
Cuando se carga un grupo de reglas, cualquier suceso de la clave antispoof se expandirá en dos reglas de filtrado. Asumiendo que la interfaz fxp0 tuviera una dirección IP 10.0.0.1 y una máscara de subred de 255.255.255.0 (o sea, un /24), la regla antispoof anterior se expandiría así:

```
block in on ! fxp0 inet from 10.0.0.0/24 to any
block in inet from 10.0.0.1 to any
```

Introducción a NAT

La Traducción de Direcciones de Red, o NAT (Network Address Translation), es un sistema que se utiliza para asignar una red completa (o varias redes) a una sola dirección IP. NAT es necesario cuando la cantidad de direcciones IP que nos haya asignado nuestro proveedor de Internet sea inferior a la cantidad de ordenadores que queramos que accedan a Internet. NAT se describe en el RFC 1631.

NAT nos permite aprovechar los bloques de direcciones reservadas que se describen en el RFC 1918. Generalmente, una red interna se suele configurar para que use uno o más de estos bloques de red. Estos bloques son:

10.0.0.0/8 (10.0.0.0 - 10.255.255.255)
172.16.0.0/12 (172.16.0.0 - 172.31.255.255)
192.168.0.0/16 (192.168.0.0 - 192.168.255.255)

Un sistema OpenBSD configurado para NAT tendrá como mínimo dos adaptadoras de red, una para Internet y la otra para la red interna. NAT se encargará de traducir los requerimientos desde la red interna, de modo que parezca que todos provienen del sistema OpenBSD en el que se encuentra configurado NAT.

Cómo Funciona NAT

Cuando un cliente en la red interna contacta con un máquina en Internet, envía paquetes IP destinados a esa máquina. Estos paquetes contienen toda la información de direccionamiento necesaria para que puedan ser llevados a su destino. NAT se encarga de estas piezas de información:

Dirección IP de origen (por ejemplo, 192.168.1.35)
Puerto TCP o UDP de origen (por ejemplo, 2132)
Cuando los paquetes pasan a través de la pasarela de NAT, son modificados para que parezca que se han originado y provienen de la misma pasarela de NAT. La pasarela de NAT registra los cambios que realiza en su tabla de estado, para así poder: a) invertir los cambios en los paquetes devueltos, y b) asegurarse de que los paquetes devueltos pasen a través del firewall y no sean bloqueados. Por ejemplo, podrían ocurrir los siguientes cambios:

IP de origen: sustituida con la dirección externa de la pasarela (por ejemplo, 24.5.0.5)
Puerto de origen: sustituido con un puerto no en uso de la pasarela, escogido aleatoriamente (por ejemplo, 53136)

Activación de NAT

Para activar NAT en una pasarela de OpenBSD, además de activar PF, también hay que activar el reenvío de paquetes IP (IP forwarding):

```
# sysctl net.inet.ip.forwarding=1
# sysctl net.inet6.ip6.forwarding=1 (si se usa IPv6)
```

Para que este cambio sea permanente, hay que añadir las siguientes líneas al archivo /etc/sysctl.conf:

```
net.inet.ip.forwarding=1
net.inet6.ip6.forwarding=1
```

Estas líneas ya existen en la instalación determinada, pero como comentarios (prefijadas con #). Hay que quitar el signo # y guardar el archivo. El reenvío de IP se activará cuando se reinicie la máquina.

Configuración de NAT

El formato general para las reglas de NAT en /etc/pf.conf es parecido al siguiente:

```
nat on extif [af] from src_addr [port src_port] to \
   dst_addr [port dst_port] -> ext_addr
```

extif El nombre de la interfaz de red.

af La familia de direcciones, que será inet para IPv4 ó inet6 para IPv6. PF suele ser capaz de determinar este parámetro basándose en la dirección (o direcciones) de origen/destino.

src_addr La dirección de origen (interna) de los paquetes que vayan a ser traducidos. La dirección de origen se puede especificar como:
Una dirección IPv4 ó IPv6 única.

src_port El puerto de origen en la capa Layer 4 de la cabecera del paquete. Los puertos se pueden especificar como:
Un número entre 1 y 65535

Configuración de NAT

dst_addr
La dirección de destino de los paquetes que hay que traducir. La dirección de destino se especifica del mismo modo que la dirección de origen.

dst_port
El puerto de destino en la capa Layer 4 de la cabecera del paquete. Este puerto se especifica del mismo modo que el puerto de origen.

ext_addr
La dirección externa (la traducción) en la pasarela NAT a la que se traducirán los paquetes. La dirección externa se puede especificar como:
Una dirección IPv4 ó IPv6 única.
Un bloque de red CIDR.
Un «Nombre de Dominio Totalmente Cualificado» (FQDN) que se resolverá por el «Servicio de Nombres de Dominio» (DNS) cuando se cargue el grupo de reglas.

```
nat on tl0 from 192.168.1.0:network to any -> 24.5.0.5
```

Esta regla indica que hay que realizar NAT en la interfaz tl0 para cualquier paquete que venga de 192.168.1.0/24, y sustituir la dirección IP de origen con 24.5.0.5.

Asignación Bidireccional (asignación 1:1)

Se puede establecer una asignación de tipo bidireccional usando la regla binat. Una regla binat establece una asignación de uno por uno entre la dirección IP interna y la dirección externa. Esto puede ser útil, por ejemplo, para colocar un servidor de web en la red interna con su propia dirección IP externa. Las conexiones desde Internet hacia la dirección externa se traducirán a la dirección interna, y las conexiones desde el servidor de web (como los requerimientos de DNS) se traducirán a la dirección externa.

```
web_serv_int = “192.168.1.100”
web_serv_ext = “24.5.0.6”
binat on tl0 from $web_serv_int to any -> $web_serv_ext
```

Excepciones a las Reglas de Traducción

Se pueden hacer excepciones a las reglas de traducción usando la clave no. Así, el ejemplo de NAT anterior se modificaría del siguiente modo:

```
no nat on tl0 from 192.168.1.10 to any
nat on tl0 from 192.168.1.0/24 to any -> 24.2.74.79
```

Y entonces los paquetes de toda la red 192.168.1.0/24 se traducirían a la dirección externa 24.2.74.79, a excepción de 192.168.1.10.

Redireccionamiento (Reenvío de Puertos)

Cuando hay un servidor de NAT delante de la red de una oficina, todas las máquinas de la red tienen acceso a Internet. Pero, ¿qué ocurre si se necesita acceder desde el exterior a una de las máquinas que hay detrás de la firewall de NAT? Aquí es donde entra en escena el redireccionamiento. El redireccionamiento permite enviar el tráfico entrante a una máquina que se encuentre detrás de una pasarela de NAT.

```
rdr on tl0 proto tcp from any to any port 80 -> 192.168.1.20
```

Esta regla redirecciona el tráfico TCP del puerto 80 (un servidor de web) a una máquina que se encuentra dentro de la red en 192.168.1.20. Así, aunque 192.168.1.20 esté detrás de la pasarela y dentro de la red, es accesible desde fuera de ella.

La parte from any to any de la regla rdr anterior puede ser bastante útil; si sabemos qué direcciones o subredes se supone que deben acceder al servidor de web por el puerto 80, podemos restringirlo con esa parte

```
rdr on tl0 proto tcp from 27.146.49.0/24 to any port 80 -> 192.168.1.20
```

Con esto se redireccionaría sólo la subred especificada. Nótese que esto implica que podemos redireccionar diversos servidores entrantes hacia diversas máquinas que se encuentren detrás de la pasarela.

Opciones en Tiempo de Ejecución

Las opciones se usan para controlar la operación de PF, y se especifican en el archivo pf.conf usando la directiva set.

set block-policy
Configuración del comportamiento predeterminado para las reglas de filtrado que especifiquen la acción de bloqueo, block.
drop - bloquea el paquete en silencio, sin devolver ningún paquete a modo de respuesta.
return - devuelve un paquete TCP RST por paquete TCP bloqueado, y un paquete ICMP Unreachable para el resto de paquetes.
Nótese que las reglas de filtrado individuales pueden anular la respuesta predeterminada.

set debug
Establece el nivel de depuración de pf.
none - ningún mensaje de depuración es mostrado.
urgent - mensajes de depuración generador por errores serios. Predeterminado.
misc - mensajes de depuración generados por varios errores (ejemplo: para ver el estado del normalizador de paquetes y las fallas al crear estados).
loud - mensajes de depuración generados por condiciones comunes (ejemplo: para ver el estado de la identificación pasiva del SO).

Opciones en Tiempo de Ejecución (2)

set fingerprints archivo
Establece el archivo desde el que se cargarán los fingerprints de los sistemas operativos. Para su uso con la identificación pasiva del SO. El archivo predeterminado es /etc/pf.os.

set limit
frags - número máximo de entradas en el pool de memoria (almacén de memoria) usado para el reensamblaje de paquetes. El número predeterminado es de 5000.
src-nodes - número máximo de entradas en la reserva de memoria usada para seguirle la pista a las direcciones IP de origen (generadas por las opciones sticky-address y source-track). El número predeterminado es 10000.
states - número máximo de entradas en el pool de memoria usado para las entradas en la tabla de estado (reglas de filtrado que especifican keep state). El número predeterminado es de 10.000.

set loginterface int
Indicar la interfaz desde la que PF debe recoger estadísticas, como el número de bytes que han entrado o salido y los paquetes que han pasado o que se han bloqueado. Estas estadísticas sólo se pueden recoger para una sola interfaz por vez. Nótese que los contadores match, bad-offset, etc... y los contadores de la tabla de estado se registrarán aunque no se haya configurado logininterface.

Opciones en Tiempo de Ejecución (3)

Optimizar PF para uno de los siguientes entornos de red:
normal - sirve para para casi todas las redes. Es el predeterminado.
high-latency - redes de gran latencia, como las conexiones por satélite.
aggressive - fuerza en modo agresivo la terminación de conexiones de la tabla de estado que hayan alcanzado el límite de inactividad. Esto puede reducir mucho los requisitos de memoria en un firewall con mucha actividad, aunque a riesgo de cortar otras conexiones inactivas demasiado pronto.
conservative - un tipo de configuración muy conservadora. Evita cortar las conexiones inactivas, aunque a coste de una mayor utilización de la memoria y de un ligero incremento en el uso del procesador.

set state-policy
Establece el comportamiento de PF en lo que respecta al mantenimiento de estados. Este comportamiento puede ser anulado en cada regla. Vea Manteniendo estados.
if-bound - los estados son vinculados con la interfaz en la que fueron creados. Si el tráfico coincide con una entrada en la tabla de estados pero no está cruzando la interfaz registrada en esa entrada, la coincidencia es rechazada. El paquete debe entonces coincidir con una regla de filtrado o será desechado/rechazado del todo.
group-bound - el mismo comportamiento que if-bound excepto que los paquetes pueden cruzar interfaces del mismo grupo, es decir, todas las interfaces ppp, etc.

Scrubbing: Normalización de Paquetes

“Scrubbing” es la normalización de paquetes con el objetivo de que no existan ambigüedades de interpretación en el destino final del paquete. La directiva scrub también realiza el reensamblaje de paquetes fragmentados, protegiendo a algunos sistemas operativos de ciertos tipos de ataques y bloqueando los paquetes TCP que lleven una combinación no válida del indicador de TCP (véase flag). Una forma simple de la directiva scrub sería:

```
scrub in all
```

Con este ejemplo de scrub se normalizarían todos los paquetes entrantes en todas las interfaces.

Un motivo para no usar scrub en una interfaz sería que se estuviera pasando tráfico de NFS a través de PF. Algunas plataformas no OpenBSD envían (y esperan recibir) paquetes extraños, paquetes fragmentados con el bit “do not fragment” activado, y éstos son rechazados con motivo por scrub. Este problema se puede solucionar usando la opción no-df. Otro motivo para no usar scrub es que algunos juegos de jugadores múltiples tienen problemas de conexión al pasar a través de PF con la directiva scrub activada. A parte de estos casos poco usuales, el uso de scrub en todos los paquetes es una práctica muy recomendable.

Anclajes y Subconjuntos de Reglas con Nombre

Además del conjunto de reglas principal, PF también puede evaluar subconjuntos de reglas. Los subconjuntos de reglas se pueden manipular sobre la marcha usando pfctl(8), para lo que disponen de un modo conveniente para alterar dinámicamente un conjunto de reglas activo. Mientras que una tabla se usa para contener una lista dinámica de direcciones, un subconjunto de reglas se usa para contener un conjunto dinámico de reglas de filtrado, nat, rdr, y binat.

Estos subconjuntos de reglas van ligados al conjunto de reglas principal mediante el uso de reglas de anclaje; las reglas de anclaje se crean con la clave anchor:

anchor nombre - evalúa todas las reglas de filtrado en el anclaje nombre
binat-anchor nombre - evalúa todas las reglas de binat en el anclaje nombre
nat-anchor nombre - evalúa todas las reglas de nat en el anclaje nombre
rdr-anchor nombre - evalúa todas las reglas de rdr en el anclaje nombre

Las reglas de anclaje anchor sólo se pueden definir en el conjunto de reglas principal.
Subconjuntos de Reglas con Nombre

Queueing: Colas de Procesamiento

Poner algo en cola es almacenarlo en orden, a la espera de ser procesado. En una red de computadoras, cuando se envían paquetes desde un servidor, éstos entran en un sistema de colas en el que permanecen hasta ser procesados por el sistema operativo. Entonces el sistema operativo decide qué cola debe procesar y qué paquete o paquetes de dicha cola. El orden en el que el sistema operativo selecciona los paquetes que va a procesar puede afectar al rendimiento de la red. Pongamos por ejemplo un usuario que estuviera ejecutando dos aplicaciones de red: SSH y FTP. Lo ideal sería procesar los paquetes de SSH antes que los de FTP, por la propia naturaleza de SSH; cuando se pulsa una tecla en el cliente SSH se espera obtener una respuesta inmediata, mientras que un retraso de unos pocos segundos en una transferencia por FTP pasa casi inadvertido. Pero, ¿qué ocurriría si el enrutador que maneja estas conexiones procesara una gran parte de paquetes de la conexión de FTP antes de procesar la conexión de SSH? Los paquetes de la conexión de SSH se quedarían en la cola (o incluso serían rechazados por el enrutador si la cola no fuera lo suficientemente grande como para mantener todos los paquetes) y podría parecer que hay retrasos en la sesión de SSH, o que va muy lenta. Al modificar la estrategia de la cola en uso, las diversas aplicaciones, usuarios y ordenadores pueden compartir bastante bien el ancho de banda de la red.

Schedulers:

El scheduler es lo que decide qué colas hay que procesar y en qué orden deben ser procesadas. Por definición, OpenBSD usa un scheduler tipo FIFO (el primero en entrar es el primero en salir). Una cola FIFO funciona como la cola de un supermercado o la de un banco, en donde el primer producto de la cola es también el primero que se procesa. Según van llegando nuevos paquetes, éstos se van añadiendo al final de la cola. Si la cola se llena, los nuevos paquetes que vayan llegando van siendo bloqueados. Esto se conoce como “tail-drop”.

OpenBSD tiene soporte para dos schedulers adicionales:

Colas Basadas en Clases (Class Based Queueing)
Colas Basadas en Prioridades (Priority Queueing)
Colas Basadas en Clases

Las colas CBQ se ordenan de un modo jerárquico. En la parte superior de la jerarquía se encuentra la cola matriz, que define la cantidad total de ancho de banda disponible. Las colas derivadas de ésta se crean bajo la cola matriz, y a cada una de ellas se les puede asignar alguna porción del ancho de banda de la cola matriz.

Colas Basadas en Prioridades

Las PRIQ (Priority Queueing) asignan colas múltiples a una interfaz de red, y dan a cada cola un nivel de prioridad único. Una cola con un nivel de prioridad más alto se procesa siempre antes que una cola con un nivel de prioridad más bajo.

La estructura de formación de colas en PRIQ es estricta, y no se pueden definir colas dentro de otras colas. Se define sólo la cola matriz, en la que se decide la cantidad total de ancho de banda disponible y, a partir de ahí, las «sub-colas» se definen bajo la cola matriz. Consideremos el siguiente ejemplo:

Cola Matriz (2Mbps)
Cola A (prioridad 1)
Cola B (prioridad 2)
Cola C (prioridad 3)
La cola matriz está definida con un ancho de banda de 2Mbps disponible para sí misma, y se han definido tres subcolas. La cola con la prioridad más alta (el número de prioridad más alto) es la que se procesa primero. Una vez que se han procesado todos los paquetes en esa cola, o si la cola estuviera vacía, PRIQ pasa a la cola que tenga el siguiente nivel de prioridad más alto. Dentro de una cola cualquiera, los paquetes se procesan del modo FIFO.

Configuración de la Formación de Colas

La formación de colas se configura en pf.conf.

altq on - activa la formación de colas en un interfaz, define el scheduler que se usará, y crea la cola matriz
queue - define las propiedades de una cola derivada

```
altq on interface scheduler bandwidth bw qlimit qlim tbrsize size queue {queue_list}
```

interface - la interfaz de red en la que se vaya a activar la formación de colas.
scheduler - el scheduler que se usará para la formación de colas. Los valores que acepta son cbq y priq. Sólo se puede activar un scheduler en una interfaz al mismo tiempo.
bw - la cantidad total de ancho de banda disponible para el scheduler. Se puede especificar como un valor absoluto usando los sufijos b, Kb, Mb y Gb para representar bits, kilobits, megabits, y gigabits por segundo respectivamente, o como porcentaje del ancho de banda de interface.
qlim - el número máximo de paquetes que puede contener la cola. Este parámetro es opcional, y la configuración predeterminada es de 50.
size - el tamaño del regulador de prueba en bytes. Si no se especifica, el tamaño se configurará basándose en el ancho de banda de interface.
queue_list - una lista de las colas derivadas que se crearán bajo la cola matriz.

Configuración de la Formación de Colas

```
queue name [on interface] bandwidth bw [priority pri] [qlimit qlim] \
   scheduler ( sched_options ) { queue_list }
```

name - el nombre de la cola. Debe coincidir con el nombre de una de las colas definidas por queue_list en la directiva altq on. Para cbq, también puede coincidir con el nombre de una cola definida por queue_list en la directiva queue anterior. Los nombres de las colas no deben exceder los 15 caracteres.
interface - la interfaz de red en la que la cola es válida. Este valor es opcional, y cuando no se especifique la cola será válida en todas las interfaces.
bw - la cantidad total de ancho de banda disponible para la cola. Se puede especificar como un valor absoluto usando los sufijos b, Kb, Mb, y Gb para representar bits, kilobits, megabits, y gigabits por segundo respectivamente, o como porcentaje del ancho de banda de la cola principal. Este parámetro sólo es aplicable cuando se usa el scheduler cbq.
pri - la prioridad de la cola. Para cbq el rango de prioridad va de 0 a 7, y para priq el rango va de 0 a 15. La prioridad 0 es la más baja. Si no se especifica, se usará un valor predeterminado de 1.
qlim - el número máximo de paquetes que puede contener la cola. Si no se especifica, se usará un valor predeterminado de 50.
scheduler - el scheduler que se utilice, ya sea cbq o priq. Debe ser el mismo que para la cola matriz.

Configuración de la Formación de Colas

```
queue name [on interface] bandwidth bw [priority pri] [qlimit qlim] \
   scheduler ( sched_options ) { queue_list }
```

sched_options - opciones adicionales que se pueden pasar al scheduler para controlar su comportamiento:

default - define una cola predeterminada a la que irán todos los paquetes que no coincidan con ninguna otra cola. Es necesaria la definición de al menos una cola predeterminada.
red - activa RED en esta cola.
rio - activa RED con IN/OUT. En este modo, RED mantendrá tamaños medios de cola múltiples y valores de umbrales múltiples, uno por cada nivel de Calidad de Servicio de IP (IP Quality of Service).
ecn - activa ECN en esta cola. ecn implica red.
borrow - la cola puede tomar ancho de banda prestado de su cola matriz. Esto sólo se puede especificar cuando se usa el scheduler cbq.
queue_list - una lista de colas derivadas que se crearán bajo esta cola. Una queue_list sólo se puede definir cuando se usa el scheduler cbq.

Ejemplo de Queue

```
DMZ = 192.168.100.0/24

altq on xl0 cbq bandwidth 100Mb queue { std_in, webext, webint }
        queue std_in bandwidth 10Mb cbq(default)
        queue webext bandwidth 500Kb  priority 5
        queue webint bandwidth 80Mb priority 3

pass in on xl0 proto tcp from any to any port 80 queue webext
pass out on xl0 proto tcp from any to any port 80 queue webext
pass in on xl0 proto tcp from  $DMZNET to any port 80 queue webint
pass out on xl0 proto tcp from any to $DMZNET port 80 queue webint
```

Este ejemplo nos permite fragmentar el ancho de banda de un segmento (Area de usuarios), a internet o bien a la DMZ, dado que la velocidad de acceso a cada una de estas zonas diferente, colocamos el tráfico en colas diferentes, asi el tráfico que va a internet esta limitado a 500 Kb, comparado con el tráfico web que va ala DMZ el cual esta en 80 Mb, tambien estamos agregando una prioridad diferente en ambas colas.

Reservas (pools) de direcciones y balanceo de carga

Una reserva de direcciones es un grupo de dos o más direcciones cuyo uso comparten un grupo de usuarios. Una reserva de direcciones puede aparecer como la dirección de redirección en las reglas rdr, como la dirección de traducción en las reglas nat y como la dirección de destino en las opciones route-to, reply-to, y dup-to de las reglas de filtrado de paquetes.

bitmask - incrusta la parte referente a la red que corresponde a la reserva de direcciones en la dirección que se esté modificando (que sería la dirección de origen para las reglas de nat, y la dirección de redirección para las reglas rdr).
random - selecciona una de las direcciones de la reserva de forma aleatoria.
source-hash - usa el método hash con dirección de origen para determinar qué dirección de la reserva debe usar. Este método asegura que una dirección de origen que venga indicada se asigne siempre a la misma dirección de la reserva.
round-robin - este método realiza una rotación en secuencia por la reserva de direcciones. Es el método predeterminado en PF, y también el único método permitido cuando se especifica una reserva de direcciones usando una tabla.
A excepción del método round-robin, la reserva de direcciones se debe expresar como un bloque CIDR. El método round-robin aceptará más de una dirección individual cuando se use una lista o una tabla.

Balanceo de carga en conexiones entrantes

Las reservas de direcciones también se pueden usar para el balanceo de carga de las conexiones entrantes. Por ejemplo, se pueden distribuir a través de varios servidores de web las conexiones entrantes al servidor de web:

```
web_servers = “{ 10.0.0.10, 10.0.0.11, 10.0.0.13 }”

rdr on $ext_if proto tcp from any to any port 80 -> $web_servers \
    round-robin sticky-address
```

Las conexiones sucesivas se redireccionarán a los servidores de web de acuerdo con el método round-robin, enviando al mismo servidor web las conexiones provenientes del mismo origen. Esta “conexión pegajosa” existirá mientras haya estados que hagan referencia a esta conexión. Una vez que los estados expiren, esta “conexión pegajosa” lo hará también. Las siguientes conexiones provenientes de este host serán redirigidos al siguiente servidor web en el round robin.

Balanceo de carga en tráfico saliente

Las reservas de direcciones se pueden usar en combinación con la opción de filtrado route-to, con el fin de balancear la carga de dos o más conexiones de Internet cuando no se encuentre disponible un protocolo de enrutamiento de múltiples caminos apropiado (como BGP4). Usando route-to con una reserva de direcciones round-robin, se pueden distribuir las conexiones salientes a partes iguales entre múltiples caminos salientes.

Un pieza adicional de información necesaria para esto es la dirección IP del enrutador adyacente en cada conexión de Internet. Esta información se pasa a la opción route-to para controlar el destino de los paquetes salientes.

```
lan_net = “192.168.0.0/24”
int_if  = “dc0”
ext_if1 = “fxp0”
ext_if2 = “fxp1”
ext_gw1 = “68.146.224.1”
ext_gw2 = “142.59.76.1”

pass in on $int_if route-to \
   { ($ext_if1 $ext_gw1), ($ext_if2 $ext_gw2) } round-robin \
   from $lan_net to any keep state
```

Balanceo de carga en tráfico saliente

Para asegurarse de que los paquetes con una dirección de origen que pertenezca a $ext_if1 sean siempre enrutados a $ext_gw1 (y lo mismo para $ext_if2 y $ext_gw2), hay que incluir las siguientes dos líneas en el grupo de reglas:

```
pass out on $ext_if1 route-to ($ext_if2 $ext_gw2) from $ext_if2 to any
pass out on $ext_if2 route-to ($ext_if1 $ext_gw1) from $ext_if1 to any
```

Finalmente, también se puede usar NAT en cada una de las interfaces salientes:

```
nat on $ext_if1 from $lan_net to any -> ($ext_if1)
nat on $ext_if2 from $lan_net to any -> ($ext_if2)
```

Registros de Bitácora

El registro de paquetes en PF se realiza mediante pflogd(8), que está a la escucha en la interfaz pflog0 y graba los paquetes en un archivo de registro (generalmente /var/log/pflog) en formato binario de tcpdump(8). Las Reglas de Filtrado en las que se especifican las claves log o log-all registran los paquetes de este modo.

El archivo de registro grabado por pflogd se encuentra en formato binario, y no se puede leer usando un editor de texto. Hay que usar tcpdump para ver el registro.

```
# tcpdump -n -e -ttt -r /var/log/pflog
```

Nótese que el uso de tcpdump para leer el archivo pflog no ofrece una visión en tiempo real. Para obtener una visión en tiempo real de los paquetes registrados, hay que usar la interfaz pflog0:

```
# tcpdump -n -e -ttt -i pflog0
```

SQUID

Squid se comporta como un caché proxy: recibe peticiones de objetos por parte de los clientes (en este caso navegadores web) y las reenvía al servidor. Cuando recibe los objetos solicitados del servidor, los envía al cliente y almacena una copia de los mismos en un caché de disco.

La ventaja del caching consiste en que cuando varios clientes solicitan el mismo objeto, este puede proporcionárseles desde el caché de disco. De este modo, los clientes obtiene los datos mucho más rápidamente que si lo hicieran desde Internet y se reduce al mismo tiempo el volumen de transferencias en red.

Además del caching, Squid ofrece múltiples prestaciones tales como la definición de jerarquías de servidores proxys para distribuir la carga del sistema, establecer estrictas reglas de control de acceso para los clientes que quieran acceder al proxy, permitir o denegar el acceso a determinadas páginas web con ayuda de aplicaciones adicionales o producir estadísticas sobre las páginas webs más visitadas y por tanto sobre los hábitos de *navegación* del usuario.

Squid no es un proxy genérico. Actúa como proxy entre conexiones vía HTTP y soporta también los protocolos FTP, Gopher, SSL y WAIS, pero no soporta otros protocolos de Internet como por ejemplo Real Audio, News o videoconferencia. Squid sólo soporta el protocolo UDP para realizar comunicaciones entre diferentes cachés, con lo que muchos programas multimedia quedarán igualmente excluidos.

Listas de Control de Acceso

Squid dispone de un elaborado sistema para controlar el acceso al proxy que, gracias al uso de ACLs, puede ser configurado de forma fácil y flexible. Se trata de listas de normas procesadas secuencialmente. Las ACLs deben ser definidas antes de poder utilizarse. Algunas ACLs como *all* y *localhost* ya están predefinidas. La mera definición de una ACL no tiene ningún efecto. Es necesario que se aplique por ejemplo en combinación con *http_access* para que puedan procesarse las reglas definidas anteriormente.

acl <acl_nombre> <tipo> <datos>
Una ACL necesita por lo menos tres especificaciones para definirla. El nombre <*acl_nombre*> se puede elegir arbitrariamente. El <*tipo*> se puede elegir de entre diferentes opciones disponibles en la sección *ACCESS CONTROLS* del archivo /etc/squid/squid.conf. La parte de datos depende del tipo de ACL y también puede ser leída desde un archivo que contenga, por ejemplo, nombres de máquinas, direcciones IP o bien URLs. A continuación algunos ejemplos:

```
acl usuarios srcdomain .mi-dominio.com
acl profesores src 192.168.1.0/255.255.0.0
acl alumnos src 192.168.7.0-192.168.9.0/255.255.0.0
acl mediodía time MTWHF 12:00-15:00
```

Listas de Control de Acceso (2)

http_access allow <acl_nombre>
http_access determina a quién le está permitido usar el proxy y quién puede acceder a Internet. Para ello deben definirse ACLs que permitan o denieguen el acceso mediante *allow* o *deny* (*localhost* y *all* ya han sido definidas con anterioridad). Se puede crear una lista completa de entradas *http_access* que será procesada de arriba a abajo y, dependiendo de qué regla pueda aplicarse en primer lugar, se permitirá o no el acceso a Internet para cada URL. Por eso la última entrada de todas debe ser *http_access deny all*. En el ejemplo siguiente *localhost* (el ordenador local) dispone de acceso libre mientras que todos los otros hosts tienen el acceso denegado.

```
http_access allow localhost
http_access deny all
```

Esto puede ser mejorado con el uso de squidGuard que aun que no le cubrimos en este taller es útil para el filtrado de contenido de manera avanzada y automática

Instalación

La instalación en OpenBSD es como sigue (version port):

pkg_add http://osmirrors.cerias.purdue.edu/pub/OpenBSD/3.8/packages/i386/squid-2.5.STABLE12-transparent.tgz

Configuración

El archivo de configuración del squid se encuentra en /etc/squid/squid.conf, se hacen los siguientes cambios:

```
http_port 127.0.0.1:3128
http_access deny to_localhost
acl our_networks src 10.0.0.0/8
http_access allow our_networks
visible_hostname cache.midominio.edu.mx
httpd_accel_host virtual
httpd_accel_port 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

Configuración

Ahora configuramos el archivo pf para que sea transparente a los usuarios:

```
int_if="gem0"
ext_if="kue0"

# Redireccionamos
rdr on $int_if inet proto tcp from any to any port www -> 127.0.0.1 port 3128

# Damos acceso
pass in on $int_if inet proto tcp from any to 127.0.0.1 port 3128 keep state
```

Nota: squid debe poder abrir el archivo del /dev/pf asi que le ponemos permisos de grupo para que lo accese..

```
# chgrp _squid /dev/pf
# chmod g+rw /dev/pf
```

Comandos Básicos para el squid

Primero debemos generar la estructura de directorios del cache conel sig. comando:

```
# squid -z
2003/04/11 16:26:13| Creating Swap Directories
```

Despues de eso iniciar squid con:

```
# squid
```

Se se realizan cambios en el squid.conf se cargan

```
# squid -k reconfigure
```

Para detenerlo usar:

```
# squid -k shutdown
```

Información útil de squid

Squid genera archivos importantes, como es el access.log, cache.log, squid.pid, estos se encuentran por default en el directorio /var/squid/logs/, aun que pueden ser cambiados en el archivo de configuración, se recomienda de ser posible tener el cache y logs en un filesystem diferente al del sistema operativo, un disco duro para esto seria ideal ya que la cantidad de lecturas/escrituras que se efectuan hacen que los discos duros despues de una par de años tengan problemas, de esta forma podemos cambiar el disco duro, crearle un File System nuevo y volver a producción en unos pocos minutos.

Instalación del FTP Proxy

Utilizamos el ftp-proxy que incluye el openbsd, su instalación es muy sencilla, editamos el archivo de /etc/inetd.conf y descomentamos la siguiente linea

```
127.0.0.1:8021 stream tcp nowait root /usr/libexec/ftp-proxy ftp-proxy -n
```

El parámetro -n es para indicar que los usuarios estan via NAT. Ahora redirigimos el flujo que salga a internet al puerto 21 a nuestro proxy (Esto en el /etc/pf.conf) aplicable a la interface de usuarios.

rdr on $int_users proto tcp from any to any port 21 -> 127.0.0.1 port 8021

reiniciamos el pf y el inetd y listo..

pfctl -Fa
pfctl -f /etc/pf.conf

```
kill -HUP  `cat /var/run/inetd.pid`
```

NetFlow

Desarrollado por Darren Kerr y Barry Bruins en Cisco Systems en 1996 (US Patent 6,243,667)

Inicialmente due diseñado para las rutas en los switch

Netflow es ahora latecnología mas utilizada para llevar la contablidad de red, Contesta las preguntas del trafico, Quién, Qué, Cómo, Cuando y Donde

Define 7 llaves únicas:

Dirección IP Origen

Dirección IP Destino

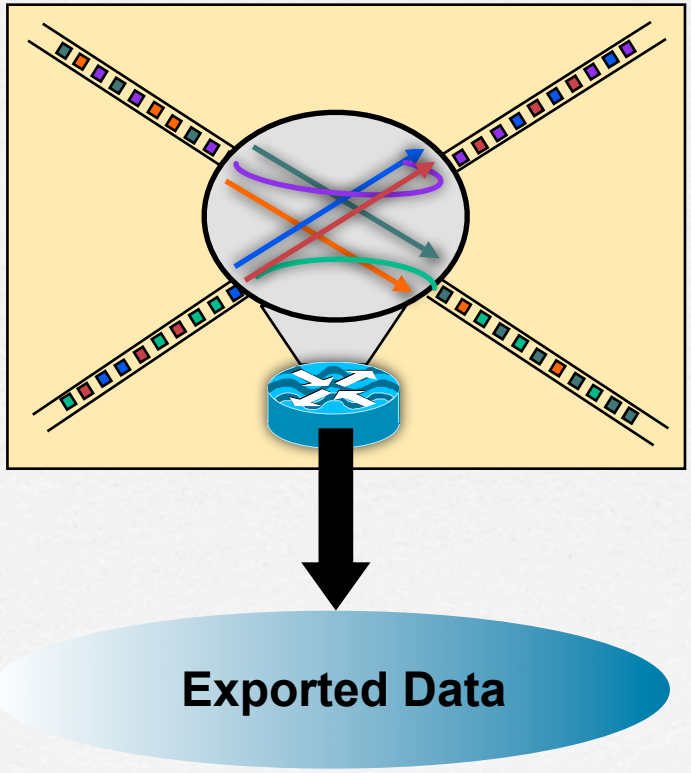
Puerto Origen

Puerto Destino

Tipo de protocolo (Paca 3

Byte de TOS

Interáce Lógica



1. Create and update flows in NetFlow Cache

SrcIf	SrcIPadd	DstIf	DstIPadd	Protocol	TOS	Flgs	Pkts	SrcPort	SrcMsk	SrcAS	DstPort	DstMsk	DstAS	NextHop	Bytes/Pkt	Active	Idle
Fa1/0	173.100.21.2	Fa0/0	10.0.227.12	11	80	10	11000	00A2	/24	5	00A2	/24	15	10.0.23.2	1528	1745	4
Fa1/0	173.100.3.2	Fa0/0	10.0.227.12	6	40	0	2491	15	/26	196	15	/24	15	10.0.23.2	740	41.5	1
Fa1/0	173.100.20.2	Fa0/0	10.0.227.12	11	80	10	10000	00A1	/24	180	00A1	/24	15	10.0.23.2	1428	1145.5	3
Fa1/0	173.100.6.2	Fa0/0	10.0.227.12	6	40	0	2210	19	/30	180	19	/24	15	10.0.23.2	1040	24.5	14

2. Expiration

- Inactive timer expired (15 sec is default)
- Active timer expired (30 min (1800 sec) is default)
- NetFlow cache is full (oldest flows are expired)
- RST or FIN TCP Flag

SrcIf	SrcIPadd	DstIf	DstIPadd	Protocol	TOS	Flgs	Pkts	SrcPort	SrcMsk	SrcAS	DstPort	DstMsk	DstAS	NextHop	Bytes/Pkt	Active	Idle
Fa1/0	173.100.21.2	Fa0/0	10.0.227.12	11	80	10	11000	00A2	/24	5	00A2	/24	15	10.0.23.2	1528	1800	4

3. Aggregation?

No

Yes

e.g. Protocol-Port Aggregation Scheme becomes

Protocol	Pkts	SrcPort	DstPort	Bytes/Pkt
11	11000	00A2	00A2	1528

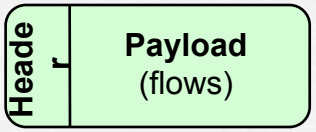
4. Export Version

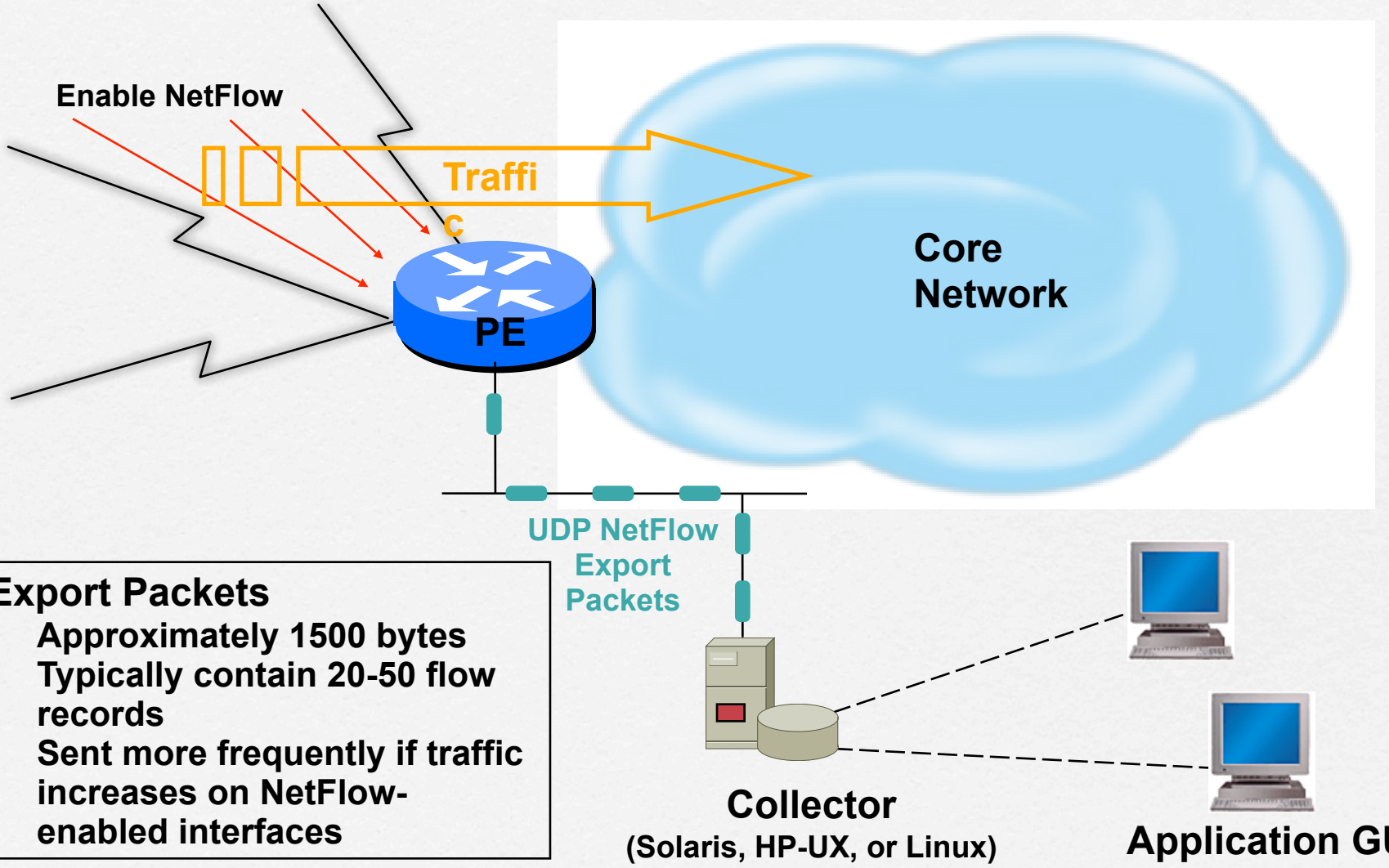
Non-Aggregated Flows – export **Version 5 or 9**

Aggregated Flows – export **Version 8 or 9**

5. Transport Protocol

Export Packet





Export Packets

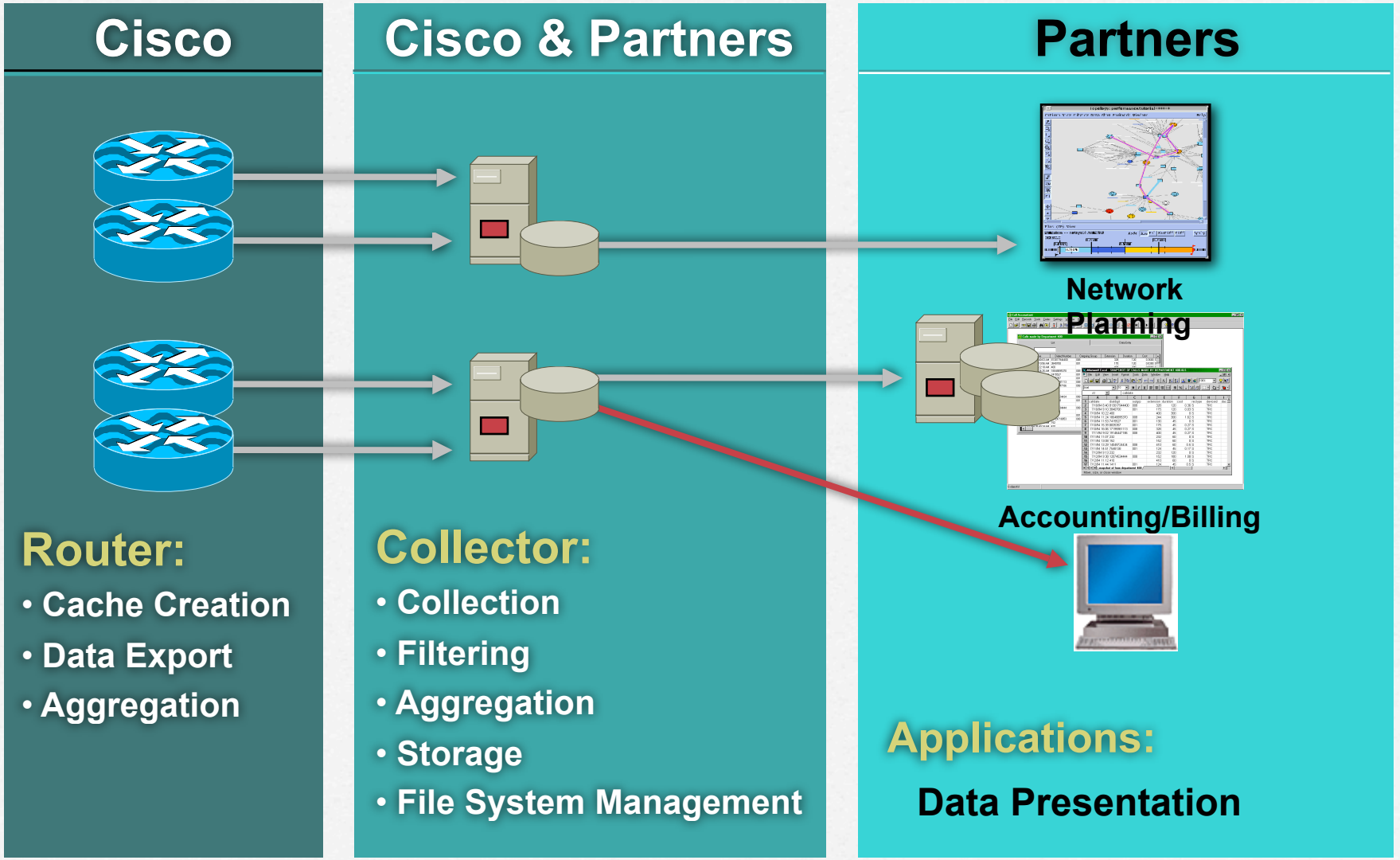
- Approximately 1500 bytes
- Typically contain 20-50 flow records
- Sent more frequently if traffic increases on NetFlow-enabled interfaces

Principios de NetFlow

Versiones

NetFlow Version	Comments
1	Original
5	La más común y estandard
7	Usada en equipos cisco Catalyst y no muy estandard Similar a la version 5, pero no incluye AS, interface, Banderas de TCP, y TOS
8	Se pueden escoger entre 11 esquemas de agregación
9	Flexible, extensible, exportable, y modular.

Infraestructura



Intalación y Configuración de NetFlow en OpenBSD

lynx http://www2.mindrot.org/files/softflowd/softflowd-0.9.7.tar.gz
tar xzvf softflowd-0.9.7.tar.gz
cd softflowd-0.9.7
./configure
make
make install

Indicamos la interfaz que se examine, y mandamos el flujo a un NMS que pueda interpretar la información como puede ser ntop.

/usr/local/sbin/softflowd -i fxp0 -n 10.1.1.2:2055 -v5 -D

(donde fxp0 es la interface donde vamos a aplicar el netflow, 10.1.1.2 es la maquina NMS y el 2055 es el puerto UDP por donde se envia el flujo, v5 indica la versión del netflow puede ser 5 o 9)

Conclusiones

Este servidor ofrece una serie de características importantes:

- Se ha probado con buenos resultados por mas de 6 años en una red de 1 y 2 E1, con salida alternativa a internet via infinitum.
- Los discos duros tienden a “volarse” despues de 2 o 3 años.
- La reinstalación es fácil y rápida.
- No requiere de compra de equipo adicional.
- Permite tener un nivel de seguridad aceptable.
- Nos muestra el comportamiento de la red (útil para detección de virus y worms)
- Control de acceso a nivel de usuario-servicio
- Disponibilidad aceptable (99.989% anual)
- Bajo mantenimiento.
- En caso de problemas... no hay soporte comercial :-(
- Inspección de paquetes a nivel de aplicación (ejemplo: dns, ntp, etc), no lo hace
- Puede tener soporte a <spamd> y autenticación por usuario el proxy.
- El costo de implementación es económico, al igual que el mantenimiento
- El tiempo de implementación pueder ser largo (un par de semanas)

Referencias

Internet
http://www.openbsd.org/
http://www.squid-cache.org/
http://www.mindrot.org/softflowd.html
http://www.ntop.org/

Libros

Building Firewalls with OpenBSD and PF, Autor Jacek Artymiak, ed http://www.devguide.net/
Absolute OpenBSD: UNIX for the Practical Paranoid, Autor Michael W. Lucas, ed. No Strach Press
Linux System Security, Autor Scott Mann, ed. Prentice Hall
Linux Firewalls, Robert L. Ziegler, Ed. New Raiders
Building Linux and OpenBSD Firewalls, Autor Wes Sonnenreich, Ed. Wiley
Mastering Network Security, Autor Chris Breton, Ed. Sybex
Building Internet Firewalls, Autor Elizabeth D. Zwicky, Simon Cooper, Ed. Oreilly

Gracias

Jorge A. Zárate Pérez
jorge.zarate@itoaxaca.edu.mx