Cortafuegos con software libre. Diez años de PF

Master oficial en Software Libre

Miguel Vidal

http://gsyc.urjc.es/~mvidal
Twitter: @mvidallopez

17 de noviembre de 2011

GSyC

Universidad Rey Juan Carlos

Cortafuegos y filtrado de paquetes TCP/IP

Tabla de contenidos

¿Qué es un cortafuegos?

- Un equipo protegido y fiable que funciona como punto de regulación entre un grupo de redes (normalmente una red privada y una red pública).
- Todo el tráfico de red entre las redes involucradas se encamina a través del cortafuegos.
- En grandes corporaciones incluso puede haber cortafuegos dentro de la red corporativa para aislar las zonas importantes de la organización.
- Crear cortafuegos es un arte: exige comprender muy bien la tecnología de red subyacente, pero también la filosofía de diseño de cortafuegos.

¿Qué es un cortafuegos?

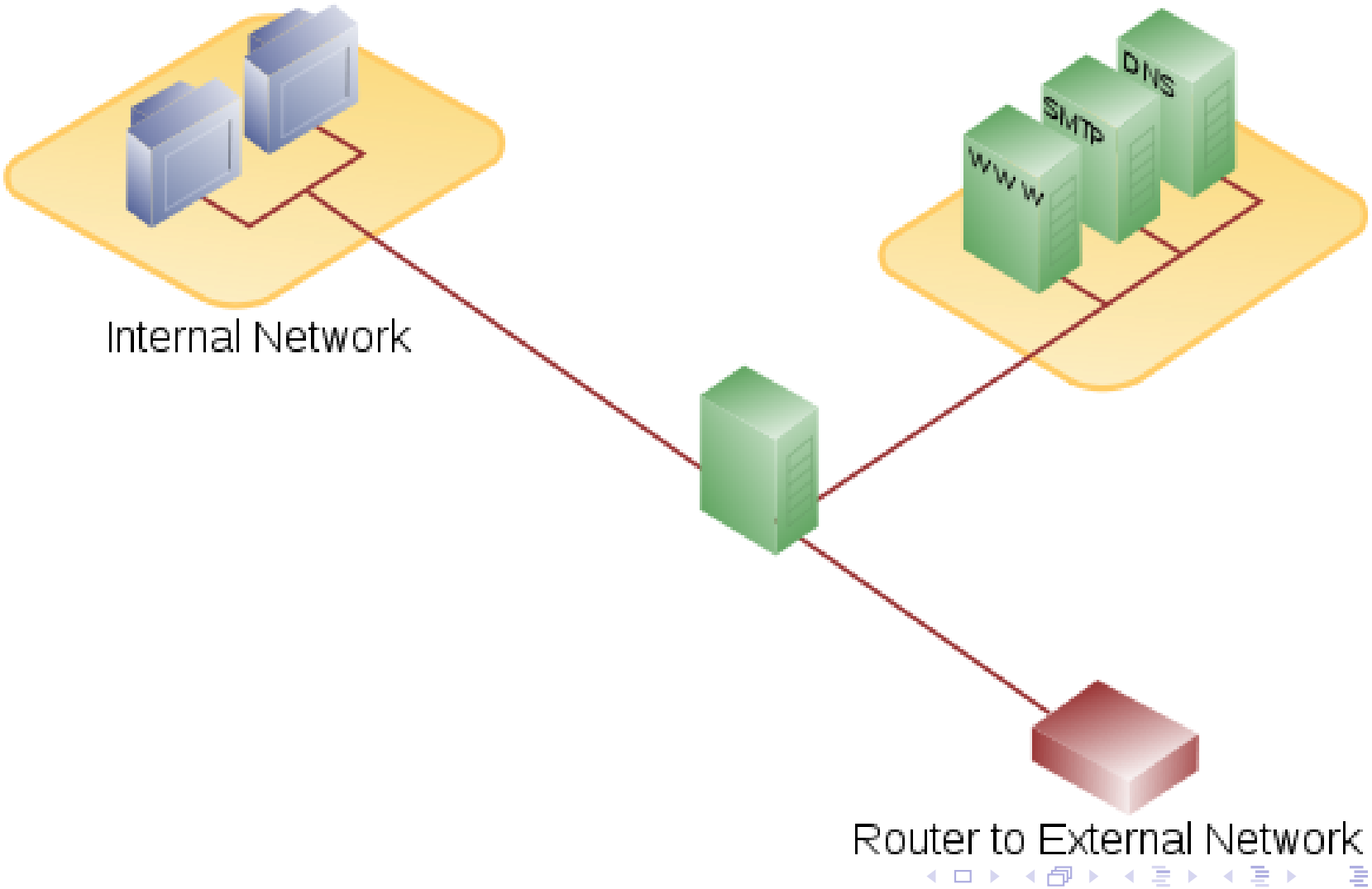
- Firewall (FW): término usado para referirse a cosas muy dispares en los últimos años.
- Se llama igual al FW casero que ponen en tu línea ADSL o al que le cuesta miles de dólares a una empresa.
- ¿Qué diferencias hay?
  - Funcionalidades que ofrece.
  - Hardware en el que corre.
  - Robustez y fiabilidad de su software.

¿Qué es un cortafuegos?

- Un cortafuegos se configura mediante un conjunto de reglas que determina qué tráfico puede pasar y cuál será bloqueado (con respuesta) o desechado (sin respuesta).
- Los cortafuegos pueden situarse de formas distintas:
  - La forma más simple (e insegura) es un solo equipo que además proporciona otros servicios.
  - La forma más sofisticada son las DMZ (o red perimetral), que puede involucrar a varios equipos cortafuegos.

Tabla de contenidos

Diseño de cortafuegos con un solo firewall



Diseño de cortafuegos con 2 firewalls (DMZ)

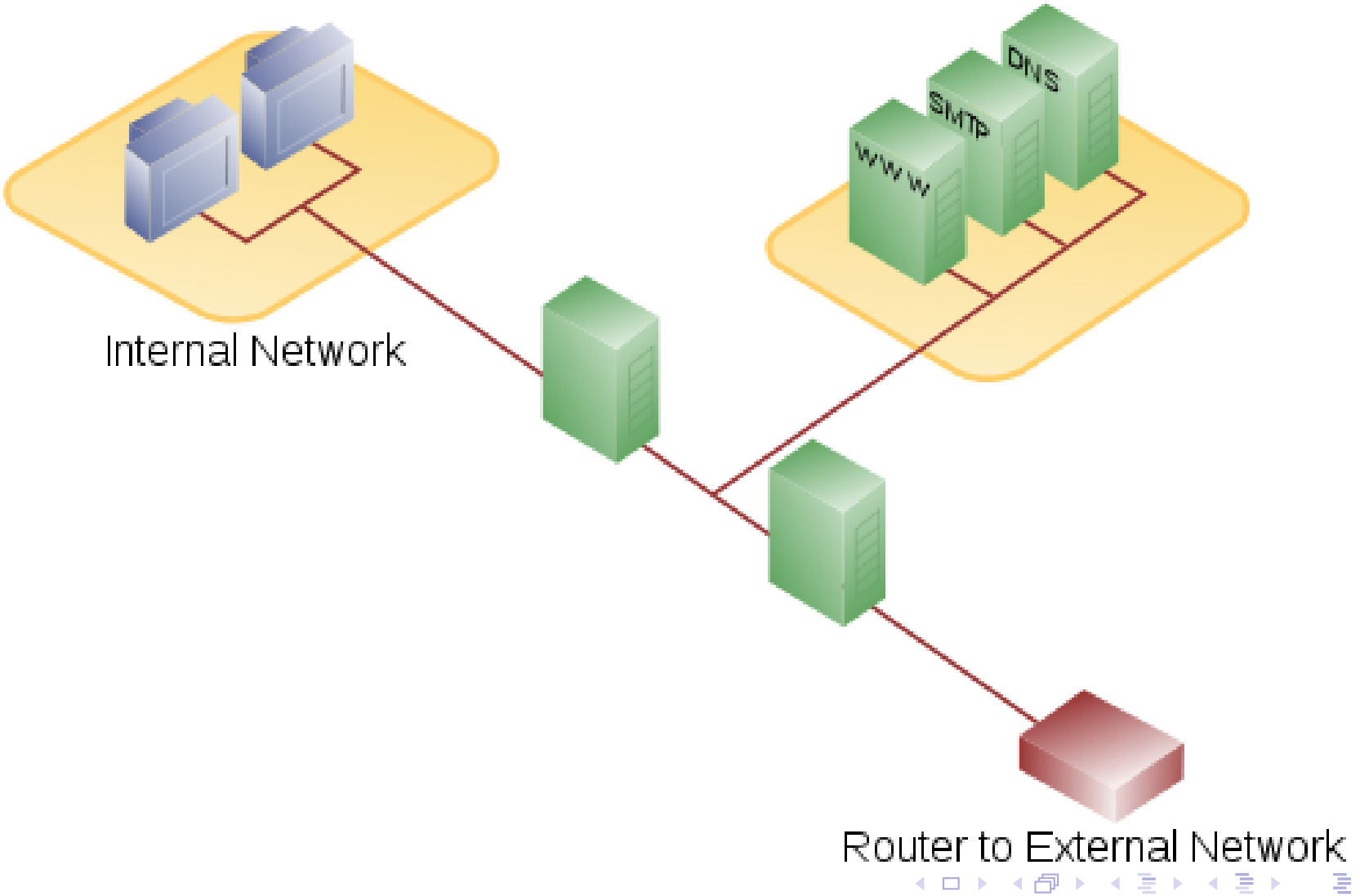


Tabla de contenidos

Tipos de procesado de paquetes

- Filtrado: decidir en distintos momentos del flujo si un paquete pasa o es bloqueado.
- Modificación: modificación mientras se mueve el flujo de paquetes
- Traducción (NAT): permite redirigir el tráfico de forma transparente mediante la modificación de la fuente, el destino o los puertos.

¿Qué es el filtrado IP?

El filtrado IP consiste en decidir qué paquetes se procesarán y cuáles serán rechazados. Algunos criterios posibles para filtrar:

- Tipo de protocolo: TCP, UDP, ICMP, etc.
- Número de puerto (para TCP/UDP)
- Tipo de paquete: SYN/ACK, datos, petición de eco ICMP...
- Origen del paquete
- Destino del paquete

Los conjuntos de reglas (*rulesets*) se componen mediante combinación de algunos de estos criterios.

¿Qué es el filtrado de IP?

- El filtrado IP es una utilidad de capa de red (layer-3).
- No conoce nada de las aplicaciones que usan las conexiones de red.
- Por ejemplo, si filtramos por puerto, ese mismo servicio podría ejecutarse en otro puerto y el firewall no lo impediría.
- Para solucionar esto, se usan servidores proxy, que gestionan la conexión y sí comprenden el servicio.

Conceptos básicos

- *default accept* versus *default deny*.
- Inspección de paquetes: *Stateful* vs *stateless*
- En los FW de primera generación no habia estado, lo que facilitaba el *spoofing*.
- La inspección de estado guarda registros de todas las conexiones de red que pasan por el cortafuegos.
- Establecimiento de la comunicación TCP en tres pasos (*Three-way handshake*):
  - SYN packet: solicitud de sincronización
  - SYN+ACK packet: sincronización y acuse de recibo del servidor
  - ACK packet: acuse de recibo (*acknowledgment*) del cliente.

Filtrado de paquetes

- Un firewall avanzado puede hacer más cosas además de bloquear.
- Realiza otras funcionalidades importantes: enmascaramiento, NAT, auditorías, gestión de ancho de banda, balanceo de carga, filtrado por criterios específicos, redundancia...

Herramientas libres para filtrado de paquetes

- `iptables`: Linux
- `ipfilter`: *Solaris, illumos, FreeBSD, NetBSD, Linux, HP-UX, IRIX
- `PF (Packet Filter)`: OpenBSD (nativo), FreeBSD, NetBSD, DragonFly.

Comparativa:
`http://en.wikipedia.org/wiki/Comparison_of_firewalls`

Diez años de Packet Filter (PF)

¿Qué es PF?

- Es un filtro de paquetes (o firewall) de tráfico TCP/IP basado en configuración dinámica (*stateful rules*).
- Considerado el mejor software libre para cortafuegos, balanceo de carga y gestión de tráfico de red.
- Comparable en funcionalidad a las soluciones privativas más caras (Cisco, Juniper, etc., $>$ 50K dólares).
- Desarrollado y mantenido por el equipo de desarrollo de OpenBSD (portado también a otros BSD).
- Equivalente (aunque mucho más funcional) a `iptables` en Linux.

Tabla de contenidos

¿Qué es PF?

- PF es parte de la pila de red del kernel.
- No solo protege de ataques, sino que permite redundancia (HA) y escalabilidad (combinado con CARP y pfsync).
- También NAT y control de ancho de banda: calidad del servicio (QoS) y ALTQ (priorización de colas).
- Busca la sencillez de las reglas, la consistencia y la legibilidad.
- Filtra basándose en cualquier paquete o conexión: dirección de origen o destino, protocolo, puerto, etc.
- A partir de estos criterios, PF ejecuta la acción que especifiquemos.

Tabla de contenidos

Antes de PF: Sobre BSD

- BSD es el acrónimo de Berkeley Software Distribution.
- Originalmente se refiere a un conjunto de software para Unix desarrollado en la Universidad de Berkeley (1975-1990).
- Por ejemplo, FFS o la implementación más popular de TCP/IP (llamada Net/2).
- Con el tiempo se convirtió en Unix libre completo: 4.4BSD.
- Dio lugar a una familia de sistemas Unix: FreeBSD, NetBSD, OpenBSD, DragonFly BSD, y, para algunas definiciones, Mac OS X de Apple.
- PF nace en el ámbito del proyecto OpenBSD.

Antes de PF: ipfilter

- OpenBSD era ya en los 90 el BSD más orientado a seguridad.
- OpenBSD usaba un subsistema llamado IPFilter, escrito por Darren Reed.
- Su código usaba una *extraña* variante de la licencia BSD.

Licencia de ipfilter (2000)

```
/*
* Copyright (C) 1993-2000 by Darren Reed.
*
* The author accepts no responsibility for the use of this software
* and provides it on an ‘‘as is’’ basis without express or implied
* warranty.
*
* Redistribution and use in source and binary forms are permitted
* provided that this notice is preserved and due credit is given
* to the original author and the contributors.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
*
* I hate legaleese, don’t you ?
*/
```

Antes de PF: ipfilter

- Se le pide a su autor, Darren Reed, que "clarifique" la licencia.

Aclaración a la licencia de ipfilter (2001)

```
/*
/* Copyright (C) 1993-2001 by Darren Reed.
*
* The author accepts no responsibility for the use of this software
* and provides it on an ‘‘as is’’ basis without express or implied
* warranty.
*
* Redistribution and use in source and binary forms are permitted
* provided that this notice is preserved and due credit is given
* to the original author and the contributors.
*
* Yes, this means that derivitive or modified works are not
* permitted without the author’s prior consent.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
* /
```

Antes de PF: ipfilter

- En 2001, aclara que esta variante no permitía modificar el código sin permiso de Darren. ¡No era software libre!
- El 30 de mayo de 2001 ipfilter se borra del árbol principal de OpenBSD.

Theo de Raadt anuncia que IPFilter será reemplazado

```
Date: Tue, 29 May 2001 19:13:11 -0600
From: Theo de Raadt <deraadt@cvs.openbsd.org>
Subject: ipf

sometime in the next 20 hours, i will be removing ipf from the source
tree since it does not meet our freedom requirements, as have been
outlined in policy.html and goals.html since the start of our project.

we will have to work on an alternative.
```

Auditoría de licencias

- OpenBSD decidió auditar las licencias del árbol de código al completo.
- Se encontraron un buen número de licencias problemáticas.
- La mayoría se resolvieron hablando con el autor.
- Unas cuantas se resolvieron reescribiendo el código o eliminándolo.
- ¡El "drama" de la licencia de IPFilter se resolvió con un nuevo firewall y con fiabilidad de la licencias en el sistema base!

Tabla de contenidos

Origen de PF

- En paralelo al problema con la licencia de ipfilter (2001), Daniel Hartmeier inició un proyecto de filtrado de paquetes.
- En junio de 2001 ya tenía un prototipo (24 de junio, primer commit).
- Durante varias semanas, OpenBSD-current no dispuso de software para firewall.
- 1 de diciembre: primera versión de PF (OpenBSD 3.0)
- Inicialmente era casi un clon de ipf.
- La compatibilidad con ipfilter dejó de ser una prioridad una vez que los usuarios de OpenBSD habían migrado.
- Hoy no debe asumirse compatibilidad con ipfilter (requiere trabajo de conversión).

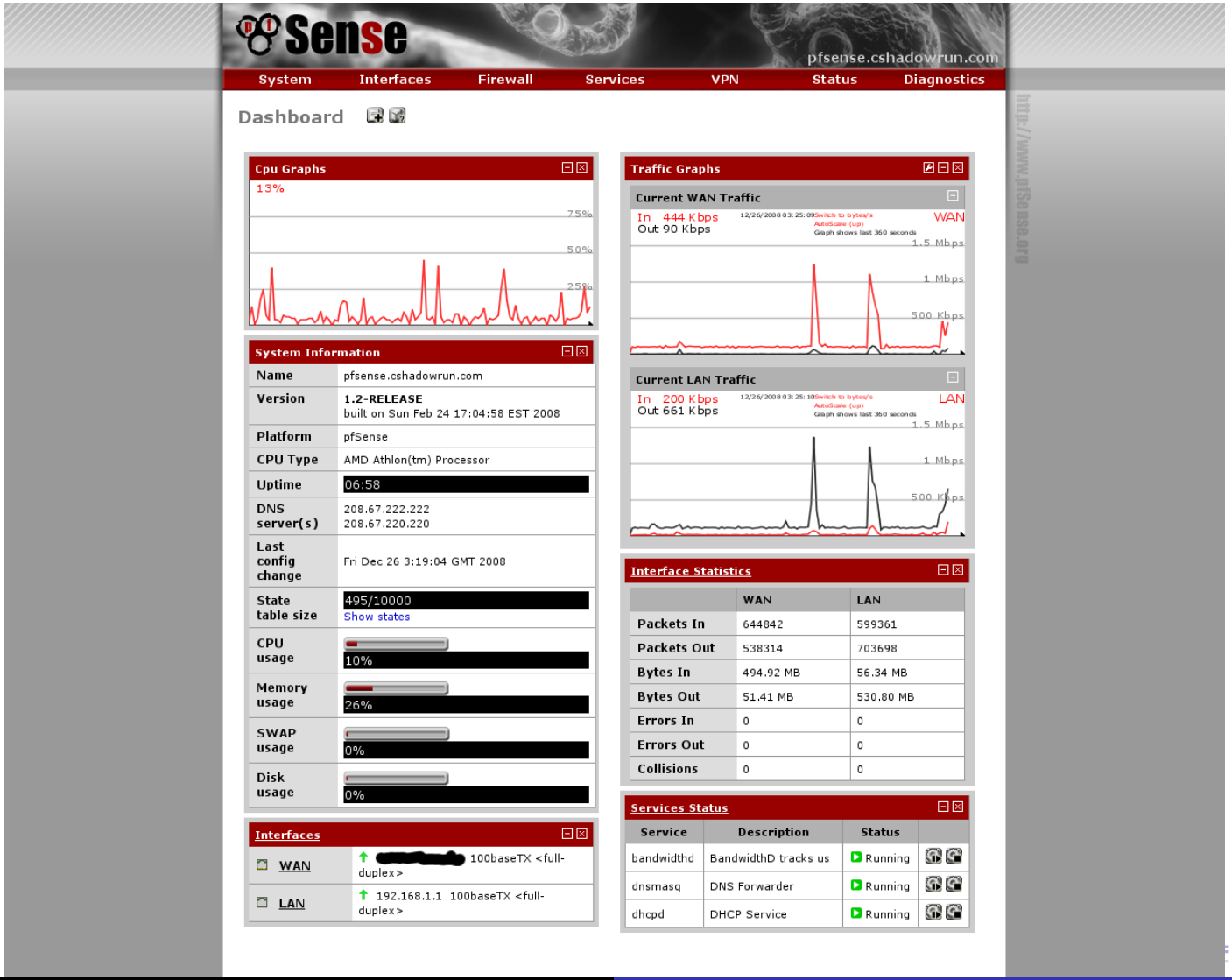
Origen de PF

- PF fue escrito desde cero por desarrolladores expertos en seguridad.
- *Paper* de Harmeier en USENIX con comparativas de rendimiento (2002).
- PF 3.1 se comportaba igual o mejor bajo estrés que IPFilter, ambos en OpenBSD.
- También superaba a iptables de Linux.

Adopción de PF

- Despertó la curiosidad de otros BSD y Unix.
- FreeBSD lo adoptó gradualmente: primero como paquete y luego (>5.3) en sus sistema base junto a ipfilter.
- También fue incorporado por NetBSD y DragonFly BSD.
- pfSense: distro de FreeBSD con PF, y con una GUI muy sofisticada que permite gestionar los conjuntos de reglas gráficamente.

GUIs: pfSense



Adopción de PF: ¿Y Linux?

- Intentos de portarlo sin éxito.
- Desarrollo profundamente integrado con la pila de red de OpenBSD.
- Exigiría reescribir grandes partes de PF.
- Otros BSD conservan un origen común con OpenBSD, lo que permitió portarlo.
- ¡Hay software libre más allá de Linux!

Tabla de contenidos

Funcionalidades de PF

- Network Address Translation (NAT)
- Gestión de ancho de banda (QoS), priorización de colas (vía ALTQ)
- Balanceo de carga
- ftp-proxy
- *Logging* y estadísticas
- pfsync y CARP para Alta Disponibilidad

Conjuntos de reglas con PF

Tabla de contenidos

Un conjunto mínimo de reglas

Un conjunto mínimo de reglas

```
block in all
pass out all keep state
```

En OpenBSD 4.1 y superiores: `keep state` por defecto (se deja por legibilidad)

Cargar las reglas

```
$ sudo pfctl -ef /etc/pf.conf
```

Macros

Se pueden definir variables (macros) para que las reglas sean más legibles y manejables:

Ejemplos de Macros

webserver = 192.0.2.12
webport = 80

Macros dentro de una regla

pass in proto tcp from any to $webserver port $webport

Listas

Las listas son dos o más objetos del mismo tipo agrupables en una regla:

Ejemplo de Lista

```
pass proto tcp to port { 22 80 443 }
```

{ 22 80 443 } es una lista.

Macros y listas pueden combinarse

```
web_servers=``{ 192.0.2.12,192.0.2.13,192.0.2.14 }''
web_ports=``{ 80 443 }''
```

Tablas

Las tablas (entre < >) sirven para agrupar direcciones IPv4 o IPv6:

Ejemplo de Tabla

```
table <goodguys>  192.0.2.0/24, !192.0.2.5
table <spammers> persist file ‘‘/etc/spammers’’
pass in on fxp0 from <goodguys> to any
block in on fxp0 from <spammers> to any
```

Traducción de Direcciones de Red (NAT)

- Permite mapear redes enteras
- Necesario cuando tenemos IPs públicas limitadas por ISP
- Nos permite aprovechar las direcciones RFC 1918 (rangos privados):
  - 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)

Ejemplo de NAT

```
pass out on em0 from 192.168.1.0/24 to any nat-to
24.5.0.5
```

Hace NAT en la interfaz em0 para cualquier paquete que venga de 192.168.1.0/24, y sustituye la dirección IP de origen con 24.5.0.5.

Redireccionamiento de tráfico

Permite acceder desde el exterior a servicios de la red interna.

Ejemplo de redireccionamiento

```
pass in on em0 proto tcp from any to any port 80 \
    rdr-to 192.168.1.10
```

Se redirecciona el tráfico TCP del puerto 80 (un servidor web) a una máquina dentro de la red interna (192.168.1.10).

El redireccionamiento tiene implicaciones de seguridad. El sistema expuesto al exterior se suele aislar en una DMZ.

Antispoofing

Previene la falsificación de la dirección IP de origen (con el propósito de esconder la dirección real o de suplantar otro nodo en la red):

Filtrado de paquetes falsificados por interfaz

```
antispoof for em0
```

Balanceo de carga

Tipos de balanceo de carga mediante reserva de IPs (*address pooling*):

- round-robin: rotación secuencial. Modo por defecto.
- random: envía cada conexión a una IP aleatoria.
- source-hash: usa un hash de la IP para asignar una conexión del pool de IPs.
- bitmask: un modo de hacer NAT entre dos bloques de direcciones IPs de igual tamaño.

Ejemplo de balanceo de carga entrante

```
web_servers = ``{ 10.0.0.10, 10.0.0.11, 10.0.0.13 }''
match in on $ext_if proto tcp to port 80 rdr-to \
     $web_servers round-robin
```

Comandos básicos

Control de PF con pfctl

```
pfctl -e      #activa PF
pfctl -f /etc/pf.conf    #carga las reglas
pfctl -nf /etc/pf.conf #chequea sintaxis de las reglas sin cargarlas
pfctl -vf /etc/pf.conf #modo verboso, vemos expansión de reglas
pfctl -s rules    #ver reglas actuales
pfctl -s all    #ver todos los parámetros
pfctl -d    #desactiva PF
sysctl net.inet.ip.forwarding=1    #Gateway. En /etc/sysctl.conf
```

/etc/pf.conf

Todo se configura y controla desde `/etc/pf.conf`. Este debe ser el orden de procesamiento de las reglas:

- Macros
- Tablas
- Opciones
- Normalización de tráfico (scrubbing)
- Gestión de ancho de banda
- Traducción (NAT)
- Redirección
- Filtrado de paquetes

Registros de bitácora

- Demonio pflogd, por defecto /var/log/pflog.
- Logs en formato binario, legibles por `tcpdump -r`.

Activar log de estadísticas en iface externa

```
set loginterface em0
```

Leer los logs

```
$ sudo tcpdump -n -ttt -r /var/log/pflog
$ sudo tcpdump -nettti pflog0     # tráfico en vivo
```

Un ejemplo completo de conjunto de reglas

```
# Macros y listas pueden combinarse
int_if=``em1''
tcp_services=``{ 22, 113 }''
udp_services=``{ domain }''
icmp_types=``echoreq''
# Opciones
set block-policy return
set loginterface em0
set skip on lo
# NAT
match out on egress inet from !(egress) to any nat-to (egress:0)
# Filtrado - lo primero bloqueamos trafico en todas direcciones
block in log
pass out quick
antispoof quick for { lo $int_if } #Antispoofing
# Permitimos paso a protocolos y puertos autorizados
pass in on egress inet proto tcp from any to port $tcp_services
pass proto udp to port $udp_services
pass in inet proto icmp all icmp-type $icmp_types #ping
pass in on $int_if # confiamos en tráfico de interfaz interno
```

Referencias

- Peter N.M. Hansteen, *The Book of PF*, 2nd Edition, No Starch, 2011.
- Daniel Hartmeier, "Design and Performance of the OpenBSD Stateful Packet Filter (pf)" (Usenix paper, 2002) `http://www.benzedrine.cx/pf-paper.html`

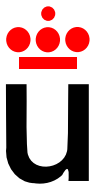
Cortafuegos con software libre. Diez años de PF

Master oficial en Software Libre

Miguel Vidal

http://gsyc.urjc.es/~mvidal
Twitter: @mvidallopez

17 de noviembre de 2011



Universidad
Rey Juan Carlos