

Securizando Bind9 bajo FreeBSD

Álvaro Navarro Clemente, Luis López Hernández
{anavarro,llopez}@gsync.escet.urjc.es
Universidad Rey Juan Carlos, Móstoles, España

Mayo del 2004

Resumen

Un entorno jail, a diferencia que chroot, proporciona un sistema virtual completo bajo un único sistema real. Estos entornos sirven para trabajar como si tuviéramos varias máquinas diferentes, cada una de ellas ofreciendo sus propios servicios a la red. Esto puede ser muy útil para, por ejemplo, construir servidores de red seguros, evitando que intrusos accedan al resto del sistema. A lo largo del presente documento describiremos cómo construir una jerarquía de servidores DNS virtuales en una misma máquina, utilizando un entorno basado en jails de tal forma que cada jaula represente una zona determinada dentro de la jerarquía DNS.

1. Introducción

Desde los primeros tiempos de Unix existe una llamada al sistema de nombre **chroot**. Su función es sencilla pero muy útil: cambiar la idea que tiene el proceso de cuál es el directorio raíz del sistema de ficheros. Dicha llamada se encarga de cambiar el número de inodo raíz que se proporciona al proceso que está bajo ese **chroot**, de tal forma que el proceso cree que está funcionando en un entorno donde todo lo que hay es el sistema de ficheros que cuelga del directorio elegido como raíz.

Sin embargo, una de las principales carencias de dicho sistema es la imposibilidad de *recrear* un sistema virtual completo y transparente de cara al usuario, es decir, con sus propia dirección IP, con su propia tabla de procesos y sus propios dispositivos virtuales. Bajo esta motivación nacen los denominados **jails** o *jaulas* bajo el sistema operativo **FreeBSD**.

Una de las ideas más interesantes a la hora de aplicar un sistema basado en **jails** es la seguridad en entornos de red. Pensemos que de cara al usuario se muestra una jerarquía de varias máquinas aparentemente independientes entre sí, por lo que si un *craker* logra explotar alguna vulnerabilidad de alguno de los demonios que tenemos a la escucha, tan sólo accederá un directorio *emph* real del servidor que hospeda todos los *jails*.

Por tanto, la idea del presente documento es la de construir una jerarquía completa de servidores DNS aprovechando la implementación de *jaulas* ya mencionada, todo ello bajo la misma máquina y con una seguridad que nos permita desenvolvernos en entornos suficientemente hostiles.

2. ¿Para qué me puede servir ésto?

Unas de las principales motivaciones que nos llevó a crear un sistema como éste, fue la posibilidad de disponer un servicio completo de DNS ideal para entornos en los que no interesa tener un servidor disponible en internet (pensemos en la seguridad), pero usable dentro de una intranet suficientemente grande en los que poder hacer balanceo de carga o algún sistema de *caching*.

3. Preparando nuestro sistema

Antes de comenzar debemos asegurarnos que tenemos todo lo necesario para la creación del entorno final:

- Sistema FreeBSD 4.0 o superior
- Disco Duro 1,5 Gb como mínimo
- Tarjeta de Red soportada por el sistema (casi todas lo están)
- Mucho café :-)

El primer paso es asegurarnos que disponemos de todas las fuentes del sistema. Para ello sincronizaremos nuestro árbol de *sources*. En nuestro caso utilizaremos la herramienta *cvsup* que instaremos desde el CD o desde los *ports* del sistema.

Una vez descargado editamos el archivo de configuración *supfile* necesario. En nuestro caso estamos utilizando la rama estable 5.2 de **FreeBSD** por lo que el fichero *supfile* quedaría de la siguiente forma:

```
*default host=cvsup.FreeBSD.org
*default base=/usr/local/etc/cvsup
*default prefix=/usr
*default release=cvs tag=RELENG_5_2
*default delete use-rel-suffix compress
src-all
ports-all tag=.
```

Ahora podemos sincronizar nuestro árbol de fuentes mediante el comando *cvsup /usr/local/etc/cvsup/supfile*. Podemos automatizar de forma robusta este proceso creando un pequeño script y añadiéndolo al *cron* para que se ejecute periódicamente. Dicho script tendría la siguiente pinta:

```
#!/bin/sh
echo Subject: `hostname` weekly cvsup run output
/usr/local/bin/cvsup -g -L 2 /usr/local/etc/cvsup/sup/supfile
```

4. Haciendo un sistema jail-friendly

Antes de comenzar a crear los diferentes **jails** debemos asegurarnos que nuestro sistema será compatible. Ya que todas las IPs de los diferentes subsistemas serán de una misma tarjeta de red, podríamos tener problemas con ciertos demonios que no sepan *atarse* a una IP en concreto causándole cierta confusión a la hora de ejecutarse.

Éste es el caso del demonio *inetd*. Para solucionarlo tan sólo tenemos que editar nuestro */etc/rc.conf* agregando lo siguiente:

```
inetd_enable="YES"
inetd_flags="-wW -a 192.168.0.4"
```

Donde 192.168.0.4 es la IP del sistema que hospedará todos los **jails**. Además de *inetd*, *sendmail* podría darnos problemas por el mismo motivo. Para solucionarlo podemos editar su configuración en */etc/mail/sendmail.cf*. o bien desactivarlo desde el fichero */etc/rc.conf* mediante:

```
sendmail_enable="NO"
```

Por último, en caso de utilizar NFS, deberíamos recompilar ciertos servicios tales como *rpcbind*, *nfsd* y *mountd*

5. Construyendo jails

Una vez tengamos nuestro sistema totalmente adaptado, nos dispondremos a crear los diferentes *jails*. En nuestro caso usaremos */usr/jails* como directorio donde se albergarán todos los subsistemas. A continuación se describe cómo configurar una jaula. En primer lugar creamos toda la estructura de directorios y además compilaremos y copiaremos los binarios y librerías necesarias para arrancar un sistema.

```
D=/usr/jails/bind1
cd /usr/src
mkdir -p $D
make world DESTDIR=$D
cd etc
make distribution DESTDIR=$D
```

Una vez completado este proceso podemos ver el resultado:

```
root@jailhost:/usr/jails/bind1# ls -l
total 37
drwxr-xr-x  2 root  wheel  1024 10 abr 01:24 bin
drwxr-xr-x  5 root  wheel   512 10 abr 01:27 boot
dr-xr-xr-x  4 root  wheel   512  5 may 15:07 dev
drwxr-xr-x 17 root  wheel  2048 10 abr 03:49 etc
lrwxrwxrwx  1 root  wheel    9 10 abr 03:49 home -> /usr/home
```

```

drwxr-xr-x  2 root  wheel  1024 10 abr 01:26 lib
drwxr-xr-x  2 root  wheel   512 10 abr 01:24 libexec
drwxr-xr-x  2 root  wheel   512 10 abr 01:22 mnt
dr-xr-xr-x  1 root  wheel    0  6 may 00:28 proc
drwxr-xr-x  2 root  wheel  2560 10 abr 01:25 rescue
drwxr-xr-x  2 root  wheel   512 10 abr 02:50 root
drwxr-xr-x  2 root  wheel  2048 10 abr 01:26 sbin
drwxr-xr-x  2 root  wheel   512 10 abr 02:09 stand
lrwxr-xr-x  1 root  wheel   11 10 abr 01:22 sys -> usr/src/sys
drwxrwxrwt  3 root  wheel   512  5 may 15:07 tmp
drwxr-xr-x 14 root  wheel   512 10 abr 03:49 usr
drwxr-xr-x 20 root  wheel   512 10 abr 01:22 var

```

Como podemos observar se ha creado una estructura de directorios similar a la de un sistema real.

A continuación creamos el kernel:

```
ln -sf /dev/null kernel
```

Y por último creamos los sistemas *dev* y *proc* :

```
mount_devfs devfs /usr/jails/bind1/dev
mount -t procfs proc /usr/jails/bind1/proc
```

6. Configurando el jail

En primer lugar asignaremos una dirección IP al subsistema. La forma adecuada es mediante IP Aliasing de la única tarjeta de red que posee la máquina:

```
ifconfig xl0 inet alias 192.168.0.21/32
```

Donde *xl0* es el interfaz de nuestra tarjeta de red (3Com en este caso).

Cuando tengamos la IP asignada y todo el sistema creado pasaremos a configurarlo. Para ello nos serviremos del comando *jail*. Para comenzar una sesión de shell con el subsistema ejecutaremos:

```
jail /usr/jails/bind1 bindhost1 192.168.0.21 /bin/sh
```

Asumiendo que no ha habido errores, nos encontraremos con un interprete de shell en que, básicamente, tendremos que configurar lo siguiente:

- Establecer un password para root. Como todo sistema recién creado la contraseña está vacía.

- Crear el fichero `/etc/fstab` vacío, evitando así posibles warnings.
- Editar `/etc/rc.conf` y deshabilitar `portmapper`
- configurar `/etc/resolv.conf` para la resolución de nombres
- Añadir usuarios al subsistema mediante `adduser`
- Establecer la hora (opcional)

Además es interesante instalar un demonio `ssh` para poder conectarnos al *jail* desde la máquina que los hospeda para futuras configuraciones. En nuestro caso, además, instalamos `bind9` y lo habilitamos para que se inicie de forma automática cuando el subsistema arranque. Para ello editamos, dentro del jail, el fichero `/etc/rc.conf`:

```
named_enable = "YES"
```

Una vez tengamos todo configurado, salimos del jail mediante el comando `exit`.

Podemos repetir el proceso de creación y configuración del jail tantas veces como subsistemas queramos tener.

7. Ejecutando el jail

Llegados a este punto tan sólo nos queda lanzar el sistema que acabamos de configurar. Para ellos nos volveremos a servidor del comando *jail*:

```
jail /usr/jails/bind1 bindhost1 192.168.0.21 /bin/sh /etc/rc
```

Tras una serie de warnings (totalmente normales) vemos como nuestro subsistema inicia los demonios, su interfaz de red y todos los servicios que normalmente requiere una máquina real. La salida debería ser, dependiendo de lo que hayamos instalado, similar a ésta:

```
Loading configuration files.
Entropy harvesting:sysctl: kern.random.sys.harvest.interrupt: Operation not permitted
interruptsysctl: kern.random.sys.harvest.ethernet: Operation not permitted
ethernet:sysctl: kern.random.sys.harvest.point_to_point: Operation not permitted
point_to_point.
Fast boot: skipping disk checks.
mount: /: unknown special file or file system
ifconfig: ioctl (SIOCDIFADDR): permission denied
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
Additional routing options:.
hw.bus.devctl_disable: 1 -> 1
Mounting NFS file systems:.
ln: /dev/log: Operation not permitted
```

```
Starting syslogd.
syslogd: child pid 14009 exited with return code 1
Starting named.
ELF ldconfig path: /lib /usr/lib /usr/lib/compat
a.out ldconfig path: /usr/lib/aout /usr/lib/compat/aout
Starting local daemons:.
Updating motd.
Configuring syscons: blanktime.
Starting sshd.
Initial i386 initialization:.
Additional ABI support:.
Starting cron.
Local package initialization:.
Additional TCP options:.
Starting background file system checks in 60 seconds.
```

lunes, 2 de mayo de 2004, 22:57:52 GMT

Si hemos configurado varios jails, ahora es el momento de lanzarlos todos. Podemos ver su estado mediante el comando *jls*

JID	IP Address	Hostname	Path
5	192.168.0.25	bind5	/usr/jails/bind5
4	192.168.0.24	bind4	/usr/jails/bind4
3	192.168.0.23	bind3	/usr/jails/bind3
2	192.168.0.22	bind2	/usr/jails/bind2
1	192.168.0.21	bind1	/usr/jails/bind1

8. Configurando la jerarquía DNS

Tenemos que tener claro qué zona de la jerarquía DNS queremos que sirva cada uno de los jails creados. En nuestro caso hemos optado por la siguiente distribución:

```
bind1 -> .
bind2 -> .com
bind3 -> dominio.com
bind4 -> subdominio.dominio.com
```

Así pues procedemos a configurar el fichero `named.conf` de cada demonio *named*. Para el caso del raíz el fichero quedaría algo parecido a:

```
options {
    directory "/etc/namedb";
    pid-file "/var/run/named/pid";

    listen-on {
        192.168.0.22/32;
    };
};

zone "."{
    type master;
    file "db.root";
};
```

El fichero `db.root` al que remitimos la zona `.`, tiene el siguiente aspecto:

```
.      IN      SOA      RAIZ.ROOTSERVER.COM.  anavarro.RAIZ.ROOTSERVER.COM. (
                                1994040142 ; Serial
                                7200      ; refresh every 2 hours
                                7200      ; retry every 2 hours
                                12096000 ; expire in twenty weeks
                                604800 ) ; minimum ttl

.      IN      NS       NS1.TGV.COM.
.      IN      NS       NS2.TGV.COM.
COM.   IN      NS       COMSERVER.COM.

COMSERVER.COM.      IN      A        192.168.0.22
ROOTSERVER.COM.     IN      A        192.168.0.21
RAIZ.ROOTSERVER.COM. IN      A        192.168.0.21
NS1.TGV.COM.        IN      A        212.128.4.4
NS2.TGV.COM.        IN      A        212.128.4.5
```

Pasamos a configurar el nodo que servirá la zona `.com`. Su fichero de configuración es similar al del jail que sirve el raíz, por lo que sólo indicamos la diferencia más notable:

```

zone "com."{
    type master;
    file "db.com";
};

```

El fichero db.com tiene la siguiente estructura:

```

COM.      IN      SOA      NODO1.COMSERVER.COM.  anavarro.NODO1.COMSERVER.COM. (
                                1994040142 ; Serial
                                7200    ; refresh every 2 hours
                                7200    ; retry every 2 hours
                                12096000 ; expire in twenty weeks
                                604800 ) ; minimum ttl

COM.      IN      NS      NS1.TGV.COM.
COM.      IN      NS      NS2.TGV.COM.
DOMINIO.COM.  IN      NS      DOMINIOSERVER.COM.

DOMINIOSERVER.COM.  IN      A      192.168.0.23
COMSERVER.COM.      IN      A      192.168.0.22
NS1.TGV.COM.        IN      A      212.128.4.4
NS2.TGV.COM.        IN      A      212.128.4.5

```

El jail encargado de servir dominio.com tiene el siguiente archivo de configuración:

```

zone "dominio.com."{
    type master;
    file "db.dominio";
};

```

Y su fichero db.dominio asociado quedaría de la siguiente forma:

```

DOMINIO.COM.  IN      SOA      NODO1.DOMSERVER.COM.  anavarro.NODO1.DOMSERVER.COM. (
                                1994040142 ; Serial
                                7200    ; refresh every 2 hours
                                7200    ; retry every 2 hours
                                12096000 ; expire in twenty weeks
                                604800 ) ; minimum ttl

DOMINIO.COM.  IN      NS      NS1.TGV.COM.
DOMINIO.COM.  IN      NS      NS2.TGV.COM.
SUBDOMINIO.COM.  IN      NS      SUBDOMINIOSERVER.COM.

SUBDOMINIOSERVER.COM.  IN      A      192.168.0.24
DOMSERVER.COM.      IN      A      192.168.0.23
NS1.TGV.COM.        IN      A      212.128.4.4
NS2.TGV.COM.        IN      A      212.128.4.5

```

Para finalizar, el jail que servirá la zona subdominio.dominio.com tendrá el siguiente *named.conf*

```
zone "subdominio.dominio.com." {
    type master;
    file "db.subdominio";
};
```

Y su db.sudominio quedaría así:

```
SUBDOMINIO.COM.      IN      SOA      NODO1.SUBDOMSERVER.COM.  anavarro.NODO1.SUBDOMSERVER.COM. (
    1994040142 ; Serial
    7200      ; refresh every 2 hours
    7200      ; retry every 2 hours
    12096000 ; expire in twenty weeks
    604800 ) ; minimum ttl
```

```
SUBDOMINIO.COM.      IN      NS       NS1.TGV.COM.
SUBDOMINIO.COM.      IN      NS       NS2.TGV.COM.
```

```
SUBDOMSERVER.COM.    IN      A        192.168.0.24
NS1.TGV.COM.         IN      A        212.128.4.4
NS2.TGV.COM.         IN      A        212.128.4.5
MAQUINA1             IN      A        212.128.1.1
MAQUINA2             IN      A        193.147.71.55
```

Referencias

- [1] Bind9 Administrator Reference Manual
OReilly
- [2] FreeBSD Handbook
<http://www.freebsd.org/>
- [3] Eldemonio.org
<http://www.eldemonio.org/>
- [4] Creating a Fake RootServer
<http://www.process.com/techsupport/multinet/787/4.html>