

CAJA DE ARENA

Desconocido y Sin Ser de Confianza

Si eres como yo, te gustará probar nuevo software, porque es ahí donde se encuentra una de las ventajas más importantes del mundo del código abierto. Casi todo está a un corto `wget, ./configure; make; make install` de distancia y no es necesario pagar, registrarse, proporcionar información personal, esperar una semana para que llegue tu CD, etc, etc. Sin embargo, ¿cómo podemos estar seguros de que nuestro software no interferirá en nuestro sistema, sobrescribirá algo, o se comportará de mala manera?

¿O qué hacer si deseamos ejecutar un servidor web que sabemos que tiene antecedentes de problemas que permiten la ejecución de código remoto en el servidor web?

Banco de Pruebas

Una técnica de administración de sistemas y programación común es el uso de bancos de prueba, que son esencialmente áreas de software restringidas (o en algunos casos, un sistema operativo completo o grupo de sistemas) que se ejecutan donde no pueden interferir con sistemas de producción. Mediante la configuración de un área de comprobación amurallada podemos conocer si

algo ha ido mal, siendo menos probable que se originen problemas severos, como pueden ser que se vea afectado nuestro servidor de ficheros real o nuestro servidor web. Además, es más fácil observar y verificar el comportamiento del software porque hay menos jaleo dentro de nuestro banco de pruebas.

Esto conduce a los dos requerimientos principales de un banco de pruebas: Necesitamos que sea posible aislar el software y necesitamos que sea posible monitorizar lo que está haciendo y controlarlo.

Afortunadamente, hace unos cuantos años, un importante número de avances en el mundo de la computación han hecho posible que sea más fácil de cumplir el primero de estos requerimientos. CPUs más rápidas, discos duros más grandes, y memorias baratas, combinados con la expansión del software de virtualización, se traducen ahora en que casi todo el mundo con un ordenador reciente, de al menos 1-2 GHz y 512MB de RAM, puede ejecutar al menos un sistema operativo completo dentro de su sistema operativo habitual.

Desafortunadamente, muchos de esos productos no cumplen muy bien con el segundo requerimiento, bien porque requieren que el sistema operativo virtualizado (conocido también como cliente) sea modificado de manera significativa o bien por usar ficheros virtuales para mantener los contenidos del hard-drive del cliente.

Pruebas en un S.O. con VMware Server

Las buenas noticias son que el servidor VMware es libre para su descarga y uso. Las malas, que se trata de un producto de código cerrado. Nótese que no he cubierto todas las opciones posibles, tales como Bochs [1], Xen [2], User-Mode Linux [3], VirtualBox [4], KVM [5], OpenVZ [6], QEMU [7], etc.; debido a que simplemente supondría demasiado contenido para las páginas de este artículo.

Además, me gusta VMware Server [8] porque requiere solamente unos cuantos módulos del kernel (*vmnet, vmmon*) y puede ejecutar casi cualquier sistema operativo como un cliente sin modificaciones en el sistema operativo del cliente.

La instalación es relativamente fácil: Simplemente descargamos y desempaquetamos el fichero y ejecutamos el script *vmware-config.pl*. Después deberemos responder a unas cuantas preguntas rápidas y estaremos listos para ejecutarlo. El inconveniente principal de VMware Server es que usa ficheros de imagen basados en disco para el sistema operativo cliente, por lo que para examinar el “disco duro” del sistema virtualizado, necesitaremos detenerlo o suspenderlo y montar luego el disco imagen (Listado 1).

La ventaja es que podemos literalmente detener un sistema operativo en seco, examinar una docena de instantáneas de nuestra elección y reorganizar cuando lo hayamos hecho.

Pruebas en una Aplicación con chroot

A veces, sin embargo, aislar un sistema operativo completo es una exageración. ¿Qué ocurre si queremos compilar algún software e instalarlo sin que afecte a nuestro sistema actual o permitirnos la opción de eliminarlo fácilmente? Aunque parezca mentira, este fue el mismo desafío que Bill Joy se planteó mientras trabajaba en BSD en los años 90. Su solución fue crear el llamado programa de utilidad y sistema chroot.

Con chroot debemos recordar algo importante: chroot **no** se diseñó como mecanismo de seguridad. En lugar de eso, fue diseñado para hacer que la instalación y comprobación de software fuera más fácil y segura. Un proceso o usuario con privilegios root puede salirse de un entorno chroot y causar daños al sistema operativo subyacente. Sin embargo, esto puede mitigarse en gran parte ejecutando todo el software dentro de chroot como usuario no-root y eliminando cualesquiera binarios

setuid potencialmente inseguros que se ejecutan como root o con otros privilegios elevados.

Compilación de un Entorno chroot

En sistemas basados en Debian DPKG y RPM, crear un entorno chroot resulta relativamente fácil. Alguna gente me acusará de ser alguien centrado en RPM, y podrían estar en lo cierto. Comencé con Slackware 1.0, aunque cambié después de ver Red Hat 3.0.3, y desde entonces he usado Red Hat y CentOS.

Además, Debian ha documentado el proceso de compilación del entorno chroot adecuadamente, por lo que no necesito repetirlo de nuevo aquí [9].

Para crear un entorno chroot completo, son necesarios algunos elementos básicos:

- un sistema operativo con algunas cosas básicas, como */dev* y *proc* (de manera que cosas como *ps* funcionarán);
- programas y librerías necesarios para ejecutar el software que deseamos comprobar; y
- opcionalmente, una manera fácil de instalar o actualizar software en chroot, lo cual es especialmente

importante si deseamos usar chroot como un entorno de producción para compartimentar software).

Paso 1: Sistema de Fichero Básico

Aquí uso */chroot* como el directorio base de chroot. Como root, ejecuto:

```
# mkdir /chroot
# mkdir /chroot/proc
# mkdir /chroot/dev
# mount -t proc \
proc/ chroot/proc
# /sbin/MAKEDEV generic -D \
/chroot/dev -d /chroot/dev
```

Paso 2: Preparar chroot para Uso de yum

La instalación del paquete *release* (por ejemplo, *centos-release*, *redhat-release*) en chroot permitirá funcionar a *yum*:

```
# rpm -Uvh - - nodeps \
- - root=/chroot/ \
centos-release-5-1.0.e15. \
centos.1.x86_64.rpm
```

Paso 3: Instalar en chroot

El RPM también instala el software en el chroot, aunque *yum* manejará dependencias y hará que las cosas sean mucho más simples:

```
# yum - - installroot=/chroot/ \
install bash yum vim-minimal
```

Como mínimo, recomiendo una shell (bash), *yum* para instalar software y el editor vim para modificar ficheros en el chroot.

Paso 4: Ficheros de Configuración de Red

Si deseamos acceder a la red desde dentro de chroot necesitaremos un fichero *resolv.conf* (permite aplicaciones donde nuestros servidores DNS tienen que encontrarse) y:

```
# mkdir /chroot/etc/
# mkdir /chroot/etc/sysconfig
# cp /etc/resolv.conf \
/chroot/etc/
# cp /etc/sysconfig/network \
/chroot/etc/sysconfig/
```

"Logging" en un chroot

En este punto, podremos acceder al chroot con un comando como *\$ chroot*

/chroot/ bash, que nos llevará hasta el directorio *chroot* y ejecutará bash desde dentro de él.

Tal y como he mencionado, chroot no es un método inherentemente seguro para aislar aplicaciones. Si nos registramos en chroot como usuario privilegiado, por ejemplo, root, y si eliminamos los binarios *setuid* y *setgid* que se ejecutan con privilegios elevados, podemos asegurar que nada se ejecuta como root dentro del entorno chroot:

```
# find/ -type f -perm +6000
```

Conclusión

La práctica de los bancos de pruebas es ahora más fácil que nunca y sus beneficios jamás han sido tan importantes. El aislamiento de aplicaciones del sistema operativo subyacente con errores, o permitir que un administrador instale un programa sin que afecte al sistema puede ahorrarnos tanto tiempo como dinero. Como en todo, la prevención y previsión pueden reducir a largo plazo significativamente la cantidad de trabajo necesario para mantener y arreglar un sistema, y un banco de pruebas ofrece una práctica herramienta para conseguirlo. ■

Listado 1: Montaje de la Imagen del Disco

```
01 # vmware-mount.pl -p
Centos.vmdk
02
03 Nr Star Size Type Id
System
04 - - - - -
-----
05 1 63 208782 BIOS 83
Linux
06 2 208845 530145 BIOS 82
Linux swap
07 3 738990 20225835 BIOS 83
Linux
08
09 # vmware-mount.pl Centos.vmdk
3 /mnt/vmware/
10
11 # df
12 Filesystem iK-blocks Used
Available Use% Mounted on
13 /dev/nb0 9796164 2548372
6742148 28% /mnt/vmware
```

RECURSOS

- [1] Bochs: <http://bochs.sourceforge.net/>
- [2] Xen: <http://www.xensource.com/>
- [3] Linux User-Mode: <http://user-mode-linux.sourceforge.net/>
- [4] VirtualBox: <http://www.virtualbox.org/>
- [5] KVM: <http://kvm.qumranet.com/kvmwiki>
- [6] OpenVZ: <http://openvz.org/>
- [7] QEMU: <http://fabrice.bellard.free.fr/qmu/>
- [8] VMware Server: <http://www.vmware.com/products/server/>
- [9] Instrucciones chroot Debian: <http://www.debian.org/doc/manuals/reference/chips.en.html#s-chroot>
- [10] FreeVPS: <http://www.freevps.com/>
- [11] Linux-VServer: <http://linux-vserver.org/>
- [12] AppArmor: <http://www.novell.com/linux/security/apparmor/>
- [13] SELinux: <http://www.nsa.gov.selinux/>