Creación de jaulas *chroot* para el mantenimiento de aplicaciones Web mediante sistemas de control de versiones (*SCV*) distribuidos sobre Debian GNU/Linux 7

Eugenia Bahit[i]

Resumen

Durante el laboratorio se crearon *jaulas chroot* mediante *JailKit*, asociadas a grupos de usuarios pertenecientes a un mismo equipo de desarrollo. Dentro de cada jaula se creó un repositorio de trabajo con *Bazaar*, el cual se estableció como ruta del *DocumentRoot* en los *VirtualHost* correspondientes. Habiendo simulado las diferentes conexiones de los miembros de los distintos equipos, se lograron mantener las diversas aplicaciones Web actualizadas, independientes unas de otras y con total abstracción del sistema circundante.

Palabras clave:

jail, jaula, enjaular, chroot, jailkit, debian, bazaar, ssh, apache, virtual host, scv

Introducción

Los servidores destinados al hospedajes de aplicaciones Web, suelen ser generalmente accedidos por el administrador del sistema y un profesional del equipo de desarrollo.

Cuando múltiples proyectos son hospedados en el mismo servidor, la cantidad de usuarios con acceso al sistema suele ser equivalente -y a veces superior- a la cantidad de proyectos alojados en el equipo.

Por lo general, el transporte de los archivos -desde los ordenadores locales de desarrollo-, suele hacerse a través de clientes de transferencia de archivos que emplean protocolos como FTP y en otros casos, se recurre a la transferencia directa a través de SSH. Ambos protocolos (FTP y SSH) han sido desarrollados con fines específicos -diferentes a los mencionados- y si bien resuelven la necesidad a niveles pragmáticos, como consecuencia, generan conflictos derivados planteando así, de forma cíclica, problemas que a largo plazo, terminan siendo irresolubles tanto para la seguridad del sistema y su entorno circundante como para la evolución y mantenimiento preciso de las aplicaciones alojadas.

Las **jaulas *chroot*** (*change root*) sobre sistemas GNU/Linux constituyen una forma de mantener el aislamiento de los usuarios del sistema, generando entornos virtuales de la raíz del mismo. El proceso de enjaulado consiste en copiar los binarios -y directorios dependientes- que serán habilitados a cada usuario, dentro de un directorio (jaula *chroot*) consignado a dicho propósito, simulando que el mismo constituye la raíz del sistema. A cada *jaula chroot* le es asignado uno o más usuarios de forma tal que cuando los mismos acceden al sistema, no pueden conocer ni investigar el árbol de archivos y directorios real, sino tan solo, la estructura creada dentro la jaula.

Los ***sistemas de control de versiones (SCV) distribuidos***, han sido desarrollados específicamente para mantener un control de cambios preciso y certero sobre el *Software* desarrollado.

Con el presente estudio, mediante el análisis lógico de la necesidad planteada de mantener en constante actualización los archivos de múltiples aplicaciones Web desde diversos puestos de trabajo, se buscó hallar un mecanismo de acción que resuelva el requerimiento de forma fehaciente, evitando la generación de conflictos subyacentes encontrados en las alternativas pragmáticas *«ut supra»* mencionadas.

El laboratorio realizado pretende sentar precedente empírico para la resolución lógica y efectiva del requerimiento, mediante el uso concomitante de jaulas *chroot* y un sistema de control de versiones distribuido.

Materiales y Métodos

Equipo empleado

Para realizar el laboratorio se ha empleado un ***Servidor Virtual Dedicado (VPS)*** con 512 MB de RAM, IP fija dedicada y disco SSD de 20 GB con microprocesador de 1 núcleo. El VPS se instaló con la versión **7.0** del sistema operativo **Debian GNU/Linux** para arquitecturas de 32 bits.

Paquetes y herramientas instaladas

- Servidor Web: *Apache 2.2*
- Enjaulado *chroot*: *JailKit*[1] *2.17*
- SCV distribuido: *Bazaar*[2] *2.6*
- Paquetes adicionales: *GNU C Compiler (gcc)* y *build-essential*[3] (para compilar e instalar *JailKit*); *Vim* (como editor de textos)

1 http://olivier.sessink.nl/jailkit/jailkit-2.17.tar.gz
2 http://bazaar.canonical.com
3 https://packages.debian.org/es/wheezy/build-essential

Procedimiento

1. Instalación de paquetes necesarios mediante *apt*:

```
apt-get install gcc build-essential vim apache2 bzr
```

2. Obtención, compilación e instalación de *JailKit*:

```
wget http://olivier.sessink.nl/jailkit/jailkit-2.17.tar.gz
tar -xzvf jailkit-2.17.tar.gz && cd jailkit-2.17
./configure
make && make install
```

3. Creación del contenedor de proyectos (directorio en el que se alojarán las aplicaciones web)

```
mkdir -p /srv/projects/jails
```

4. Creación de las jaulas para cada proyecto

```
mkdir -p /srv/projects/jails/{proyecto1,proyecto2,proyecto3}/www
```

Solo a los fines prácticos del laboratorio, dentro de cada una de las carpetas de los proyectos, se creó una carpeta denominada `www` *la cual ha sido utilizada como ruta para el* `DocumentRoot` *de cada uno de los respectivos* `VirtualHost` *de Apache.*

No se crearon estructuras de directorios complejas para las aplicaciones Web puesto que las mismas no hubieran podido influir en el resultado del estudio. Así mismo, como único contenido de los archivos VirtualHost, solo se indicaron las directivas ServerName y DocumentRoot sin ningún otro agregado.

5. Creación de los grupos de trabajo

Se crearon tres grupos de usuarios, denominados *equipo1*, *equipo2* y *equipo3* respectivamente.

```
groupadd <equipo>
```

6. Creación de usuarios

Se crearon cuatro usuarios para cada par de grupo/proyecto:

```
useradd -g <equipo> -md /home/<usuario> -s /bin/bash <usuario>
```

Posteriormente se estableció una contraseña de acceso diferente a cada uno de los usuarios:

```
passwd <usuario>
```

7. Copia de recursos necesarios en cada una de las jaulas

Para poder disponer del intérprete de *Python* en cada una de las jaulas (necesario para el funcionamiento de *Bazaar*), así como de sus binarios, dependencias e incluso, páginas *man*, se agregó una sección `[python]` al archivo de configuración[4] de *JailKit* con todas las rutas obtenidas a través del comando `whereis python`.

Se agregaron una a una, para cada proyecto, las secciones `python`, `ssh`, `jk_lsh`, `editors`, `basicshell` y `extendedshell`:

```
jk_init -vj /srv/projects/jails/<proyecto>/ <seccion>
```

Bazaar se copió en cada una de las carpetas, mediante la opción `jk_cp` de *JailKit*:

```
jk_cp /srv/projects/jails/<proyecto>/ bzr
```

8. Asignación recursiva de los grupos creados a cada uno de los directorios del *DocumentRoot* de los proyectos

```
chown -R :<equipo> /srv/projects/jails/<proyecto>/www
```

Dado que por defecto el grupo no contaba con permisos de escritura y éstos serían necesarios para que los miembros del equipo pudiesen escribir en la carpeta del proyecto consignado, los mismos han sido asignados de forma manual:

```
chmod 775 -R /srv/projects/jails/{proyecto1,proyecto2,proyecto3}/www
```

9. Asignación de usuarios a cada una de las jaulas

```
jk_jailuser -m -j /srv/projects/jails/<proyecto>/ <usuario>
```

4 /etc/jailkit/jk_init.ini

10. Prueba

Al intentar conectar a uno de los nuevos usuarios mediante SSH, la conexión se cerraba inmediatamente después de aportar la contraseña consignada al usuario. Se editó el archivo `passwd` de la jaula del usuario, modificando la *shell* que *JailKit* establece por defecto, cambiando la ruta `/usr/sbin/jk_lsh` por la de la *shell* de *GNU Bash*, `/bin/bash` resolviendo así el problema.

Sin embargo, tras lograr la conexión mediante SSH, el *promt* de la *shell* mostraba la leyenda *«I have no name!»* en lugar del nombre del usuario.

El problema fue erradicado tras copiar la librería de resolución de nombres del sistema, dentro de la jaula del usuario:

```
jk_cp <jail-path> /lib/i386-linux-gnu/libnss_compat.so.2
```

11. Creación de ramas y repositorios

Conectados ya como usuarios, fue posible convertir la carpeta `www` de cada proyecto, en un repositorio y asignarlo a la vez como rama principal, tras presentar el usuario a *Bazaar*:

```
bzr whoami "$USER <$USER@$HOSTNAME>"
bzr init-repo /www
cd /www
bzr init
```

Una vez creadas las ramas y repositorios, se realizaron los *branch* desde cada uno de los ordenadores locales de los miembros del equipo, habiendo logrado operar de forma absolutamente normal y sin inconvenientes inesperados.

Resultados

El enjaulado *chroot* fue exitoso al 100%. Los miembros de un grupo no pudieron acceder más allá de la carpeta raíz asignada al proyecto que les fue consignado.

Los integrantes de cada grupo de desarrollo tuvieron a su disposición los comandos y herramientas fundamentales para trabajar en el equipo remoto con total libertad, sin afectar por ello la seguridad del servidor.

El trabajo mediante el sistema de control de versiones permitió un control preciso y cuidado de la aplicación desarrollada y evitó por completo la sobre-escritura y/o el borrado accidental de archivos así como la falta de actualización de los mismos.

Discusión

La incapacidad de los usuarios “enjaulados” de ejecutar comandos del sistema que requieran permisos de administrador, puede suponer un problema considerable a la hora de mantener al día aplicaciones Web que requieran un *gateway* entre la aplicación y el servidor Web, como podría ser el caso de las aplicaciones Web desarrolladas en *Python*, ya que el *gateway* requiere del reinicio del servidor Web en cada cambio del código fuente, para mantener al día la aplicación.

Finalizado el laboratorio principal, los escasos intentos por poner a disposición de los usuarios enjaulados servicios como *Apache* y *MySQL* han sido infructuosos. No obstante, se contempla como tema de investigación futura.

A fin de reforzar la seguridad del sistema, sería factible -sin lugar a dudas-, la habilitación de acceso mediante autentificación RSA en lugar del uso de contraseñas, a fin de restringir no solo la conexión desde equipos autorizados, sino además, frustraría cualquier tipo de ataque por fuerza bruta.

i

Eugenia Bahit – LAECI (Laboratorio de Altos Estudios en Ciencias Informáticas)

483 Green Lanes. London, N13 4BS (UK)