Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Consejos para administrar CVS (11187 lectures)

Per **Domingo Fiesta Segura**, ***C2H5OH*** (http://www.krenel.net)

Creado el 20/04/2002 05:17 modificado el 20/04/2002 16:46

Aquí tenéis una versión completamente remodelada del anterior artículo. Cambio debido a que ciertos detalles del anterior eran incorrectos. En lugar de arreglar lo que había me resultaba más sencillo rehacerlo desde el principio.

Índice

1. Introducción

Hay mucha literatura existente sobre como administrar un *pserver*. De hecho, es la modalidad de servidor CVS más documentada que existe; sencillamente, porque es la más cómoda y sencilla de administrar. Pero el mismo *pserver* es inseguro por naturaleza. Mucha gente lo critica por la falta de encriptación en las transacciones.

Con este artículo ya no pretendo hacer la mejor guía del administrador de CVS (las hay mejores), sino una pequeña receta para manejar el *pserver* cómodamente.

Que lo disfrutéis.

---

2. Breve reseña para refrescar la memoria

A continuación un **breve** resumen sobre la administración del *pserver* en el CVS. Si queréis una guía más completa podéis mirar en la última sección de éste artículo.

En esta sección, todos los ficheros mencionados se encuentran en $CVSROOT/CVSROOT.

NOTA
Es recomendable que para editar los ficheros del directorio $CVSROOT/CVSROOT uséis el

propio cvs, es decir, hacer checkout de los ficheros y luego commit para guardar los cambios. Es el consejo que dan todos los manuales.

2.1. La gestión de usuarios

La principal razón para montar un *pserver* es la tacañería a la hora de dar cuentas de usuarios en una máquina (por ejemplo, nuestra máquina). El *pserver* tiene una gestión de usuarios independiente; ahora, es extremadamente simple. Para que la lista de usuarios del *pserver* sea **realmente** independiente tendremos que poner la siguiente línea en el fichero `config`.

```
plain
```

El fichero `passwd` del *pserver* tiene el siguiente formato:

```
plain
```

El tercer campo (`usuario_sistema`) es un usuario de sistema sobre el que `usuario` será mapeado. Así, todas las interacciones sobre el sistema de ficheros del servidor (*pserver*) se realizarán con los privilegios de `usuario_sistema`.

2.2. Formato de los ficheros de configuración

El resto de los ficheros que nos interesa tocar tienen el mismo formato:

```
plain
```

La expresión regular se compara contra la ruta, relativa a $CVSROOT, del directorio en el que estamos trabajando. Si la ruta coincide con la expresión regular entonces se ejecuta `ruta_hacia_ejecutable` al cual le podemos pasar cuantos `arugmentos` queramos.

En la parte de la expresión regular podemos poner *ALL*, que coincide **siempre**, o *DEFAULT*, que coincide **por defecto**.

2.3. El proceso del commit

Para entender mejor el significado de cada fichero de configuración es bueno saber el orden en el que son examinados. Al hacer un commit se examinan los siguientes ficheros, en orden:

1. commitinfo
2. verifymsg
3. **Aquí se realiza el commit**
4. loginfo

Si alguno de los programas ejecutados por `commitinfo` y `verifymsg` devuelve un valor diferente de cero el commit se aborta[1]. En la siguiente sección veremos algunos usos que le podemos dar a cada uno de éstos ficheros.

[1] Como el estándar UNIX de toda la vida.

3. Limitar el acceso

Si no buscamos hilar fino a la hora de restringir acceso en un repositorio. Nos podemos valer de los archivos `readers` y/o `writers`. Normalmente sólo hace falta uno de éstos.

Si ninguno de éstos ficheros existe en $CVSROOT/CVSROOT cualquier usuario CVS tiene acceso **total** (checkout y commit) al repositorio. Si existe `readers` los usuarios listados en él tienen acceso de **sólo-lectura** (checkout) y el resto tienen acceso total. Si existe `writers` los usuarios listados tienen acceso total y el resto tiene acceso de sólo-lectura. Si existen **ambos** ficheros, únicamente se tiene en cuenta el `writers`.

Nótese que tanto `readers` como `writers` **NO** contienen usuarios de sistema, sino del *pserver*. Si únicamente utilizamos éste mecanismo para limitar el acceso al repositorio no necesitamos mapear los usuarios CVS sobre usuarios del sistema.

3.1. Limitación de acceso por grupos de usuarios

Ahora pasamos a mostrar algunas posibilidades para restringir los accesos de forma más *personalizada*. Para esta sección asumo que se tienen creadas tres cuentas de usuario en la máquina: `cvsadmin`, `cvsanon`, `cvsdevel`. Donde las dos últimas no pueden loggearse en el sistema (sólo las usaremos para mapear usuarios CVS). Además, los siguientes grupos: `cvspublic`, `cvsprivate`. Donde `cvsadmin` y `cvsdevel` pertenecen a ambos y `cvsanon` únicamente pertenece a `cvspublic`.

Reconozco que habrá mejores maneras de organizarlo, pero para nuestros ejemplos lo consideraremos así. El resto depende de vuestra imaginación.

3.2. *Checkouts*

Si queremos limitar los checkouts en términos de público y privado (como vamos a mostrar) haremos uso de las cuentas de sistema `cvsanon` y `cvsdevel`. Suponemos que nuestro `passwd` tiene la siguiente forma:

```
plain
```

Ahora supongamos que nuestro $CVSROOT tiene la siguiente forma:

```
plain
```

Con todo montado así. Un usuario anónimo únicamente podrá hacer checkout del proyecto *gnu_pepe_manolo*. Los usuarios `pepe` y `manolo` podrán hacer checkout de todo **excepto** del proyecto *privador* que pertenece al los usuarios **locales** pertenecientes al grupo `torpedete`. Esto nos permite utilizar el mismo repositorio para tareas locales y remotas (con restricciones).

¿Dónde está el truco? En los permisos. El proceso de checkout necesita la creación de ficheros de bloqueo (*lockfiles*) por el tema de la exclusión mutua; para evitar checkouts y commits inconsistentes (en caso de accesos concurrentes). Por lo tanto se necesita permiso de **escritura** en el **directorio** del que se quiere hacer checkout. Si no se tiene dicho permiso, el checkout no se puede llevar a cabo.

Por eso necesitábamos mapear los usuarios CVS con usuarios de sistema. Así, cuando `pepe` se ha autentificado (`cvs login`) para cada operación que realiza el sistema (servidor) lo considera como `cvsdevel`. Ojo, el *pserver* lo sigue viendo como `pepe`.

El único problema que se puede presentar es el caso de que `pepe` y `manolo` tengan que hacer una práctica (cada uno) para la misma asignatura. Con esta configuración del repositorio `pepe` puede hacer checkout de la práctica de `manolo` y viceversa. Pero esto se puede solucionar con los permisos del sistema, añadiendo más usuarios para ser mapeados.

3.3. *Commits*

La limitación de commits es más sencilla de implementar en el sentido de que no implica al sistema donde corre el *pserver*. Para hacer commits lo que más nos interesa es implementar ACLs (*Access Control Lists*); las cuales se pueden implementar a través de scripts.

El método es sencillo: simplemente consiste en *crear* (o utilizar) algún programa u script que recibe varios parámetros (entre ellos el usuario y el directorio afectado) y devuelva cero o distinto de cero según unos requisitos (por ejemplo, una lista de control de acceso). Y todo esto gracias al fichero `commitinfo`.

Lo montaríamos de la siguiente manera: Tenemos un programita dándole un nombre y una ruta va y lo comprueba en una lista asociativa devolviendo cero o distinto de cero según algún criterio. Para hacerlo funcionar basta con que editemos el fichero `commitinfo` de la siguiente forma:

```
plain
```

El `$USER` que me acabo de sacar de la manga[2] se expande por el nombre del **usuario CVS**, en nuestro ejemplo cualquiera de los posibles: `anonimo`, `pepe` o `manolo`. Así si, por ejemplo, es `manolo` el que hace el commit, `ruta_a_mi_programita` recibe los parámetros `manolo` y la ruta hasta el fichero que se pretende insertar. Por lo tanto, aquí no nos sirve de nada el mapeado de usuarios.

[2] Debo confesar que el truco del `$USER` lo descubrí un poco de forma empírica.

4. Y queda camino por delante

Como véis, administrar un *pserver* da juego a ideas muy creativas y variadas. Desde enviar mails a un grupo de desarrolladores tras cada commit hasta sincronizar varios repositoros de backup. También se pueden validar los mensajes de log a través del fichero `verifymsg`. Lo mejor es probar o, si eres vago, buscar algo ya hecho ;-)

4.1. Más seguridad

Si lo que te preocupa es la seguridad has de saber que es posible montar un servidor CVS sobre ssh[3] sin excesivas complicaciones. De lo que tendréis que prescindir en este caso es de la tacañería a la hora de crear cuentas de sistema, pues es necesario para poder utilizar ssh.

Montar el servidor así es algo que no he probado todavía pero sé que lo necesitaré pronto, así que me estoy documentando. A continuación os doy algunos enlaces para que os informeis acerca del tema:

- Los consejos de SourceForge.(13)
- Guía rápida de CVS sobre SSH.(14)
- Un HOWTO más breve todavía.(15)

4.2. Otras fuentes de información

- SourceForge(16), sección **F**.
- Un tutorial(17) de CVS en castellano de mucha calidad.
- Un fabuloso libro publicado y también disponible por la red(18).
- El manual oficial de CVS, también conocido como Cederqvist manual.(19)

[3] También se puede montar en una red *kerberos* pero no es algo fácil de hacer

Generado: lunes, 23-agosto-2004 2:40

Lista de enlaces de este artículo:

1. http://bulma.net/body.phtml?nIdNoticia=1286&nIdPage=1#sec_1
2. http://bulma.net/body.phtml?nIdNoticia=1286&nIdPage=2#sec_2
3. http://bulma.net/body.phtml?nIdNoticia=1286&nIdPage=2#sec_2_1
4. http://bulma.net/body.phtml?nIdNoticia=1286&nIdPage=2#sec_2_2
5. http://bulma.net/body.phtml?nIdNoticia=1286&nIdPage=2#sec_2_3
6. http://bulma.net/body.phtml?nIdNoticia=1286&nIdPage=3#sec_3
7. http://bulma.net/body.phtml?nIdNoticia=1286&nIdPage=3#sec_3_1
8. http://bulma.net/body.phtml?nIdNoticia=1286&nIdPage=3#sec_3_2
9. http://bulma.net/body.phtml?nIdNoticia=1286&nIdPage=3#sec_3_3
10. http://bulma.net/body.phtml?nIdNoticia=1286&nIdPage=4#sec_4
11. http://bulma.net/body.phtml?nIdNoticia=1286&nIdPage=4#sec_4_1
12. http://bulma.net/body.phtml?nIdNoticia=1286&nIdPage=4#sec_4_2
13. https://sourceforge.net/docman/display_doc.php?docid=761&group_id=1#top
14. http://rch2.mine.nu/cvs_over_ssh.php
15. http://cvs.ecoinformatics.org/HOWTO-cvs-over-ssh.html
16. http://sourceforge.net/docman/?group_id=1

17. http://congreso.hispalinux.es/ponencias/iarenaza/cvs-como.html
18. http://cvsbook.red-bean.com/cvsbook.html
19. http://www.cvshome.org/docs/manual/

E-mail del autor: etanol _ARROBA_ krenel.net