Autentificación SSH con llave privada (private key)

Autor: Sergio González D. **(sergio.gonzalez.duran@gmail.com)**

Ya no es nada raro que un centro de cómputo o en un site se encuentren varios sistemas Linux actuando como servidores de archivos, de correo, por supuesto web, vpn, como ftp, etc. Como administrador de sistemas es muy probable que entre ellos copies archivos, configuraciones, los respaldes entre ellos, etc. En este artículo de LinuxTotal te presento como evitar el usar contraseñas comunes entre estos servidores autentificándolos con llaves privadas, y a través de **ssh** para acceso via terminal y **scp** o **rsync** para copias de archivos entre servidores.

La principal razón para usar llaves privadas con ssh y scp es automatizar el respaldo o copias de archivos entre servidores Linux. El equipo A enviará toda la carpeta "/var/log" al equipo B todas las noches a las 23:00 horas. Si lo hacemos con **scp** sin llave privada, no sería posible ya que tendríamos que indicar la contraseña cada vez que iniciara el respaldo. También podemos usar llaves privadas con **ssh** y la opción *c* para ejecutar comandos remotos en otros equipos sin necesidad de indicar la contraseña. Haciendo posible la creación de scripts automatizados para monitorear otros servidores sin intervención de contraseñas y de manera segura.

Por otro lado, también es posible crear los certificados o llaves privados y seguir usando contraseñas sobre la llave, esto obviamente incrementa enormemente la seguridad entre los servidores, pero no es útil para tareas automatizadas. Si entre dos servidores no necesitas automatización pero si mucha seguridad (por ejemplo entre un servidor vpn y un cliente laptop que lo configure), esta solución es ideal, llaves privadas con contraseñas. Veamos como utilizar ambos casos.

Creando las llaves, sin contraseña de frase o passphrase

En el equipo "cliente" generaremos las llaves como el usuario que se conectará al servidor. Ejemplo, usuario 'sergio' en el equipo 'cliente', esto lo haremos utilizando el comando **ssh-keygen**, y las opciones *-t dsa* que indica el tipo dsa (Digital Signature Algorithm) como medio de encriptación, y la opción *-b 1024* que indica 1024 bits, aunque podemos omitirla ya que los certificados del tipo dsa por defecto siempre son de 1024 bits.

```
00 $> ssh-keygen -t dsa -b 1024
01 Generating public/private dsa key pair.
02 Enter file in which to save the key (/home/sergio/.ssh/id_dsa):
03 Enter passphrase (empty for no passphrase):
04 Enter same passphrase again:
05 Your identification has been saved in /home/sergio/.ssh/id_dsa.
06 Your public key has been saved in /home/sergio/.ssh/id_dsa.pub.
07 The key fingerprint is:
08 e7:0e:2e:d6:aa:90:6e:9b:ac:ad:7f:6f:1d:23:50:28 sergio@cliente
```

Numeré la salida para entender lo que sucede, en la línea '02', el programa propone el directorio donde guardará el par de llaves que se crearán la llave pública y privada, y este por defecto es el directorio oculto '.ssh', en el HOME del usuario. Es posible indicar otro directorio si asi se desea.

En las líneas '03' y '04' se pide un 'passphrase' que es como una contraseña solo que es posible indicar espacios con la intención mas bien de que se use una frase a modo de contraseña. En este primer ejemplo usaremos el certificado generado para tareas automatizadas, como por ejemplos respaldos, asi que no es necesario indicarlas y las dejamos en blanco.

Las líneas '05' y '06' nos indican que se generó correctamente la llave privada 'id_dsa' y la llave pública 'id_dsa.pub'

Y por último la línea '08' es el fingerprint de nuestro archivo público, veamos:

```
$> cd ~/.ssh
$> more id_dsa.pub
ssh-dss AAAAB3NzaC1kc3MAAACBAN6SEI4Qqzb23pJYRXIAtPmGJHln5hFdthFq43ef+ifR29v2IknXCFwefKK8j
mjhFQu1jNXOgF0PAZTfivRVFn6Q9FRsyXU9s+fx+xQiW3mf3y4IX654O82SLGl7Vhh5UsvG8r8d8pV6R+L22sV7Gk
Cap4xr/0gFArHmFwAxfQAAAIEAmVYjPYAdQ9DCNWP+J44xDDn+03anWgyoZqSPPs23djyVQ756U4VitM0GiIQQ89H
Hxo4Y5skKihnPMtB+bFNbP/2SmGdPz1AOmb7tvRrTkj5VLtXeDeB3ulowUKarwiBVVvAvxtxmozoZHOADWqrEPizx
1oXguj+OhVB7mlsVhhkq53OxKKJbZqsl9hkOiSxaLUfQBNu6Ae441ekIObqolWNCBIvCO3uQYOozyzNGBhqHE7FVc
S7daieIKNmFer2hO/SBxzepMrWAiIUnUsP5irmYspkjGlQxP+hxw= sergio@cliente
```

mmmm, esto no parece ser el 'fingerprint' previamente mostrado, pero si lo es (nótese en la última línea 'sergio@cliente' que indica que es el certificado público de 'sergio' en el equipo 'cliente'), asi que si deseas ver su huella digital original y comprobar que no se ha alterado en ningún caracter, entonces usamos de nuevo el comando **`ssh-keygen`** con las opciones *l* y *f* que muestra el fingerprint a partir del archivo indicado.

```
$> ssh-keygen -l -f id_dsa.pub
1024 e7:0e:2e:d6:aa:90:6e:9b:ac:ad:7f:6f:1d:23:50:28 id_dsa.pub
```

Pues asi de sencillo, ahora habrá que copiar el certificado o llave pública al equipo 'servidor', asi que primero crearemos en la cuenta del mismo usuario en el 'servidor' la misma carpeta '.ssh'

```
(en el equipo 'servidor')
sergio@servidor $> pwd
/home/sergio
sergio@servidor $> mkdir .ssh; chmod 700 .ssh
sergio@servidor $> cd .ssh
sergio@servidor $> touch authorized_keys; chmod 600 authorized_keys
```

Creamos el directorio '.ssh' con permisos 700 solo para el usuario, y dentro de este creamos el archivo vacio (de

momento) 'authorized_keys' y con permisos restringidos también de 600. (más sobre permisos aqui). En este archivo es donde pondremos el contenido de 'id_dsa.pub' y puede contener todas las llaves públicas que se deseen. Es decir, un renglón por equipo al que estemos autorizando.

Ahora copiamos desde el 'cliente' el contenido de la llave pública a la carpeta '.ssh' en el 'servidor':

```
sergio@cliente $> cd ~/.ssh
sergio@cliente $> cat id_dsa.pub | ssh sergio@servidor "cat - >> ~/.ssh/authorized_keys"

(o también podrías copiar el archivo al servidor y luego agregarlo a authorized_keys)

sergio@cliente $> cd ~/.ssh
sergio@cliente $> scp id_dsa.pub servidor:/tmp/.
sergio@servidor password:
(y después en el servidor)
sergio@servidor $> cd /tmp
sergio@servidor $> cat id_dsa.pub >> /home/sergio/.ssh/authorized_keys
sergio@servidor $> rm id_dsa.pub
```

De un modo u otro ahora en el 'servidor' en la cuenta de usuario 'sergio' tenemos autorizado por medio de un certificado digital (llave pública) al mismo usuario a que se loguee o use herramientas como **`scp`** sin necesidad de contraseña:

```
[sergio@cliente ~]$ ssh servidor
Last login: Tue Mar 18 19:50:07 2008 from 192.168.1.9
[sergio@servidor ~]$
```

O usando **`scp`**:

```
[sergio@cliente ~]$ scp archivo.txt servidor:.
archivo.txt                                           100%  409     0.4KB/s   00:01
(el 'archivo.txt' se copia al servidor y no se preguntó por una contraseña)
```

Usando llaves públicas con passphrase (`ssh-agent` y `ssh-add`)

(Se hace lo mismo al momento de generar el certificado:**`ssh-keygen -t dsa`**, solo que ahora se indica una frase contraseña.)

Si se esta fuera de la intranet, cliente y servidor en distintos puntos de Internet, y no haremos uso de tareas automatizadas, sino que más bien nos conectaremos al servidor con propósitos de administración remota, lo mejor será indicar una contraseña o frase de contraseña muy larga (15 o más caracteres, mayúsculas, minúsculas, números y símbolos) a la llave pública. De esta manera será sumamente dificil ser hackeados por este método, ya que no solo el hacker tendrá que saber la contraseña sino que tendrá que tener un certificado público válido en el servidor para que pueda ser autentificado. (Claro suponiendo que el servidor nunca haya sido comprometido y este completamente actualizado y con la mejor seguridad posible).

Solo que ahora tendremos el inconveniente de estar indicando la contraseña cada vez que copiemos algo al servidor. Entonces, hagamos uso de más herramientas del paquete openssh, **`ssh-agent`** y **`ssh-add`**. Primero veamos un ejemplo sin el uso de estos comandos lo incómodo que resulta trabajar al estar indicando la contraseña de la llave.

```
[sergio@cliente ~]$ scp archivo1.txt servidor:.
Enter passphrase for key '/home/sergio/.ssh/id_dsa':     (se ingresa la contraseña)
archivo1.txt                                              100%   10KB  10.2KB/s   00
[sergio@cliente ~]$

(unos minutos después deseamos copiar otro archivo)

[sergio@cliente ~]$ scp archivo2.txt servidor:.
Enter passphrase for key '/home/sergio/.ssh/id_dsa':     (se ingresa la contraseña de nuevo)
archivo2.txt                                              100%   15KB  10.7KB/s   00
[sergio@cliente ~]$
```

Resulta bastante incómodo y más si la frase de contraseña es complicada (como debe ser), asi que usaremos **`ssh-agent`** y **`ssh-add`** para crear en la sesión temporal un 'llavero' con la llave almacenada temporalmente solo por esta sesión, asi que solo la tendremos que indicar una sola vez:

```
[sergio@cliente ~]$ ssh-agent /bin/bash         (se inicia un subshell a partir del actual con ssh-ag
[sergio@cliente ~]$ ssh-add                     (añade la identidad dsa al agente)
Enter passphrase for /home/sergon/.ssh/id_dsa:  (se indica la frase contraseña una sola vez)
Identity added: /home/sergio/.ssh/id_dsa (/home/sergio/.ssh/id_dsa)
[sergio@cliente ~]$ scp archivo1.txt servidor:.
archivo1.txt                                          100%   10KB  10.2KB/s   00:
[sergio@cliente ~]$ scp archivo2.txt servidor:.
archivo2.txt                                          100%   15KB  10.7KB/s   00:
[sergio@cliente ~]$
```

Como podrá observarse, solo basta indicar la frase contraseña una sola vez y por lo que dure la sesión no necesitas indicarla de nuevo, permiténdote trabjar con tu servidor remoto de manera muy segura.
