Conceder acceso a un segundo administrador

En las prácticas anteriores vimos cómo crear un servidor virtual en AWS y cómo acceder a él usando las claves generadas en el propio entorno. Pero, ¿tenemos claro qué estamos haciendo exactamente? En esta práctica vamos a suponer que necesitamos conceder acceso a un segundo administrador a nuestra máquina, por ejemplo, al profesor para que evalúe nuestro trabajo.

Le daremos permisos de superusuario, crearemos un par de claves y le habilitaremos el acceso usando su clave privada.

Instalación del servidor Debian

Entra en el "Learner Lab" de AWS Academy y crea un servidor Debian con los mínimos recursos posibles. Bastará con los que te ofrezca la plataforma por defecto.

Puedes usar el par de claves generado en la primera práctica.

Crear un nuevo usuario

Una vez instalada nuestra Debian, tendremos un usuario `admin` creado automáticamente por AWS. Ese será nuestro usuario administrador.

Accede al servidor por SSH en modo comando con el usuario `admin`.

Pero supongamos que queremos que nuestro profesor pueda acceder a la máquina con permisos de administrador. Para ello le crearemos un usuario `profe` y le daremos permisos de `sudo` . Estos permisos nos permitirán que cualquier comando que ejecutemos en el terminal precedido de la palabra `sudo` se ejecute como *root*. De la misma forma, cualquier comando que ejecutemos con nuestro usuario sin `sudo`, será ejecutado con los permisos de nuestro usuario, por lo que nos protegemos de *liarla* con un comando que no toca como *root*.

Dicho esto, hay varias formas de proceder, veamos una sencilla. Se trata de modificar el archivo del sistema encargado de recoger estos permisos: `/etc/sudoers`.

Ya en nuestro servidor remoto, cambiaremos de nuestro usuario al usuario `admin` a `root`:

```
admin@ip-172-31-42-159:~$ sudo su -
```

Comprueba cómo el prompt se convierte en el de root:

```
root@ip-172-31-42-159:~#
```

Ahora, puedes crear un nuevo usuario usando el comando `adduser nuevo-usuario`. Reemplaza "nuevo-usuario" con el nombre que desees darle al nuevo usuario y responde a las distintas preguntas que te va haciendo.

Para crear el usuario `profe`:

```
root@ip-172-31-42-159:~# adduser profe
Adding user `profe' ...
Adding new group `profe' (1001) ...
Adding new user `profe' (1001) with group `profe (1001)' ...
Creating home directory `/home/profe' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for profe
Enter the new value, or press ENTER for the default
    Full Name []: Profesor
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] Y
Adding new user `profe' to supplemental / extra groups `users' ...
Adding user `profe' to group `users' ...
root@ip-172-31-42-159:~#
```

Luego, agrega al nuevo usuario al grupo "sudo" para otorgarle permisos de administrador:

```
root@ip-172-31-42-159:~# usermod -aG sudo profe
```

Para confirmar que el usuario se ha agregado correctamente al grupo "sudo", puedes verificar el archivo /etc/group:

```
root@ip-172-31-42-159:~# grep '^sudo:' /etc/group
sudo:x:27:admin,profe
```

Comprobamos cómo tanto `admin` como `profe` pertenecen ahora al grupo de sudoers.

Pero ahora todavía no podemos conectarnos a la máquina con el usuario profe, porque solo podemos conectarnos con par de claves y `profe` no las tiene. Vamos a generárselas.

Generación del par de claves.

En primer lugar nos crearemos el par de claves para `profe`, pública y privada, **desde del ordenador local o bien desde el terminal de Linux o desde PowerShell en Windows**, utiliza

el comando (**sin sudo**):

```
ssh-keygen -b 4096
```

Si dejáis las opciones por defecto, creará una clave privada `id_rsa` y una clave pública `id_rsa.pub` en el directorio `/home/nombreusuario/.ssh`. Compruébalo.

Os pedirá una contraseña para proteger el uso de la clave privada. Puesto que precisamente queremos agilizar el proceso de conexión por SSH para no introducir contraseñas, **debéis dejarla vacía**.

Una vez creado el par de claves, tal y como hemos visto en el apartado anterior, *el servidor SSH (Debian) debe poseer nuestra clave pública para que podamos autenticarnos con nuestra clave privada*, que como su nombre indica, sólo debemos poseer nosotros y por eso nos identifica unívocamente.

Copia la clave pública desde el ordenador local al servidor Debian

Este proceso de copia se puede realizar fácilmente con el comando `ssh-copy-id` si nuestro servidor aceptara login por contraseña.

```
$ ssh-copy-id usuario@ip_servidor
```

Pero como nuestro servidor solo acepta login por claves ssh, entonces deberemos hacerlo manualmente. Así entenderemos mejor lo que ocurre realmente. Sigue estos pasos:

1. Desde el terminal local

Copia la clave pública generada al portapapeles. Si tienes Windows, utiliza Notepad. Y si tienes Linux puedes utilizar este comando.

```
$ cat ~/.ssh/id_rsa.pub | pbcopy
```

Para instalar pbcopy puedes realizar lo siguiente:

```
sudo apt update && sudo apt install xclip -y
```

Y crear un alias para usarlo:

```
# Añadir en ~/.bashrc o ~/.zshrc
alias pbcopy='xclip -selection clipboard'
alias pbpaste='xclip -selection clipboard -o'
```

2. En el servidor remoto Debian

Conéctate al servidor Debian como `admin`. Ya sabes cómo.

Cada usuario guarda las claves públicas en el archivo denominado **~/.ssh/authorized_keys**. Puedes ver la clave pública del usuario `admin` correspondiente a la clave privada con la que se conecta con:

```
$ cat /home/admin/.ssh/authorized_keys
```

Pero ahora necesitamos crear una nueva clave pública para `profe`, así que desde el usuario `admin` cambiaremos al usuario `profe`

```
admin@ip-172-31-42-159:~$ su profe
profe@ip-172-31-42-159:~$
```

Cambiaremos a la carpeta raiz de profe:

```
profe@ip-172-31-42-159:~$ cd
profe@ip-172-31-42-159:~$ pwd
/home/profe
```

Ahora creamos la carpeta .ssh y creamos y editamos authorized_keys

```
profe@ip-172-31-42-159:~$ mkdir .ssh
profe@ip-172-31-42-159:~$ cd .ssh/
profe@ip-172-31-42-159:~/.ssh$ nano authorized_keys
```

Ahora pegamos el contenido del portapapeles que copiamos anteriormente con CTRL+U y cerramos guardando con CTRL+X.

Deberemos cambiar los permisos del fichero authorized_keys.

```
profe@ip-172-31-42-159:~/.ssh$ chmod 600 ~/.ssh/authorized_keys
```

Reinicia el servicio SSH para aplicar los cambios:

```
profe@ip-172-31-42-159:~/.ssh$ sudo service ssh restart
```

Comprobación de acceso al servidor con ese nuevo usuario

Ahora ya podemos conectarnos **desde nuestro equipo local** a la máquina remota usando la clave privada que generamos para `profe`.

```
$ ssh -i ~/.ssh/id_rsa profe@direccion-ip-publica-servidor-debian
```

Para finalizar, si vas a generar y usar varias claves para distintas finalidades podría ser interesante asignarles nombres descriptivos según las mismas.

> *Lógicamente, ahora deberíamos proporcionarle al profesor la clave privada para que se pudiera conectarse al servidor. Y deberíamos hacerlo desde un entorno securizado para evitar que terceros pudieran capturarla. Sin embargo, este no sería el proceso habitual. Lo hemos hecho así para aprender el funcionamiento. Pero en un entorno real sería el profesor el que generaría el par de claves y facilitaría al alumno la clave pública. De esta forma el profesor mantendría su clave privada y podría acceder a la máquina del alumno sin haber expuesto en ningún momento su clave privada.*

Para Windows

Este módulo está diseñado desde un cliente Linux conectándose al servidor Linux, por lo que el cliente SSH está integrado en el propio terminal. Para Windows existen multitud de alternativas como cliente SSH, desde utilizar el propio WSL2 (Windows Subsystem Linux) de forma similar a lo que aquí se describe, hasta utilizar cualquier otro de los varios clientes disponibles

Por ejemplo, si utilizáis Putty, deberéis seguir los pasos que detallan en este tutorial para configurar las claves.

En caso de utilizar otro cliente, buscad la forma de hacerlo pues diferirá en cada caso.