LECCIÓN 3.1 - PRÁCTICA 2

SSH continuación

Lo primero que se va a hacer es permitir el acceso sin contraseña. ¿Por qué hacerlo? Podemos tener contraseñas largas ,que sean trabajosas de escribir y puede resultar incómodo aministrar mas de una máquina tecleando constantemente. Además, adía de hoy, todo el mundo puede conseguir de forma astuta y fácil nuestras contraseñas sin más que siendo un poco observador. Además por comodidad, también por seguridad.

Para conseguir el acceso sin contraseña,si bien ssh utiliza cifrado simétrico, vamos a utilizar un cifrado asimétrico, teniendo una llave privada (que la tiene siempre el cliente) y una llave pública que copiaremos en nuestro servidor. Lo primero que vamos a hacer es generar esos dos archivos; esas dos claves:

Para generar estos dos archivos usamos el comando:

> ssh-keygen

1. Nos situamos en el localhost y tecleamos:

   > ssh-keygen (hacemos enter en el enter file... y no ponemos contraseña; enter solo)

   Ya tendríamos nuestros dos archivos.

2. Para ver las dos llaves:

   > ls -la .ssh/

   Llave privada: id_rsa

   Lave pública: id_rsa.pub

> Lo que pretendemos es enviar un paquete cifrado con nuestra llave privada y el receptor tendrá que leerlo descifrándolo con la llave pública.

3. Vamos ahora a copiar la llave pública en el servidor. Desde el cliente hacemos:

   > ssh-copy-id -p 22022 192.168.56.15

   Nos informa que ha copiado la llave pública en la máquina de destino.

4. Comprobamos iniciar sesión: (nos permitirá entrar sin contraseña):

   > ssh 192.168.56.15 -p 22022

5. Ahora vamos a desactivar el acceso por contraseña desde el servidor. Para ello, editamos el archivos de configuración y le decimos que no deje acceder por contraseña:

   > Ponemos no en "PasswordAuthentication"

6. Comprobamos a travé del cliente que ponemos seguir accediendo exitosamente sin contraseña.

7. Al intentar acceder desde otra terminal, nos dice que no tenemos permiso; no tiene más formas que la llave pública para autenticarse. La única forma posible para hacerlo es editando de nuevo el archivo de configuración. Entonces copiamos nuestra llave pública en el servidor igual que antes.

8. Vamos a ver ahora otra medida de seguridad extra que sería el dejar exclusivamente a unos únicos usuarios que puedan accerder. Para ello, indicamos dichos usuarios en el fichero de configuración en el apartado "AllowUsers".

    1. Creamos un usuario en el cliente con contraseña.

    2. Si intentamos acceder al servidor con nuestro usuario nuevo, vemos que no podemos acceder:

        ```
        ssh 192.168.56.15 -l user -p 22022
        ```

        Porque no tenemos activado el acceso por contraseña.

    3. Si volvemos a intentar acceder como el usuario nuevo, cambiando la autenticación por contraseña en yes, veremos que no nos deja; tenemos que crear el usuario en la máquina receptora; accederíamos a la máquina del servidor, pero del usuario nuevo que hemos creado.

    4. Si queremos que solo sea un usuario el que pueda acceder vamos al fichero sshd_config y modificamos el campo:

        ```
        AllowUsers nombre_user
        ```

LECCIÓN 3.2 (LAMP) - PRÁCTICA 2

Cuando hablamos de la pila LAMP, hablamos del concepto de "pila"; vamos apilando software uno encima de otro. LAMP viene de (Linux Apache MariaDB/MySQL PHP/Python).

Instalación LAMP

Instalamos Apache

Para ello, hacemos:

```
dnf search apache
```

```
sudo dnf install httpd
```

Una vez instalado, comprobamos que se ha hecho la instalación correctamente:

```
systemctl status httpd
```

Nos aparecerá deshabilitado. Lo habilitamos:

```
sudo systemctl enable httpd
```

Y lo arrancamos:

```
sudo systemctl start httpd
```

> systemctl status httpd (active)

Vamos a comprobar que efectivamente funciona, usando el comando curl (ver una URLo transferir una URL):

Haciendo

> curl localhost

Vemos como nuestro servidor http en el puerto 80 está funcionando y nos devuelve una página web.

Instalamos PHP

Para ello, hacemos:

> dnf search php

> sudo dnf install php

Para ver que está funcionando el intérprete de php hacemos:

> php -a

Instalamos MariaDB

Para ello, hacemos:

> dnf search mariadb

> sudo dnf install mariadb

Si hicieramos systemctl status mariadb nos sale como que no se encuentra el servicio. Esto de debe a que lo que hemos instalado es un cliente. Hay que instalar mariadb-server.

Comprobamos estado de nuevo. Estará deshabilitado, lo habitlitamos e iniciamos (todo con systemctl).

Para comprobar si nos deja acceder a mysql:

> sudo mysql -u root (con root si nos deja)

A continuación viene un paso muy importante: tenemos que instalar

> sudo mysql_secure_installation (si no pones sudo no te deja)

Si volvemos a intentar acceder a mysql como antes no nos deja. Tenemos que hacer:

> mysql -u root -p

A continuación, comprobemos si Apache es capaz de servirnos. Para ello, nos vamos a una terminal remota del anfitrión y hacemos:

> curl 192.168.56.10

> ping 192.168.56.10 (comprobamos si hace ping)

Vemos que no hay conexión. Esto es porque el cortafuegos (firewall) está impidiendo que tengamos acceso al puerto. Hacemos entonces:

> sudo firewall-cmd --add-port=80/tcp

> sudo firewall-cmd --add-port=80/tcp --permanent

> sudo firewall-cmd --reload

Hacemos

> ssh quintinmr@192.168.56.10 -p 22022

Una vez dentro,

> less /etc/httpd/conf/httpd.conf

> cd /var/www/html

> sudo vi index.php

Copiamos en el archivo el código:

```
    <?php
    $enlace = mysqli_connect("127.0.0.1", "mi_usuario", "mi_contraseña",
"mi_bd");

    if (!$enlace) {
        echo "Error: No se pudo conectar a MySQL." . PHP_EOL;
        echo "errno de depuración: " . mysqli_connect_errno() . PHP_EOL;
        echo "error de depuración: " . mysqli_connect_error() . PHP_EOL;
        exit;
    }

    echo "Éxito: Se realizó una conexión apropiada a MySQL! La base de
datos mi_bd es genial." . PHP_EOL;
    echo "Información del host: " . mysqli_get_host_info($enlace) .
PHP_EOL;

    mysqli_close($enlace);
    ?>
```

A continuación, nos conectamos como usuario root al mysqpl:

> mysql -u root -p

Y hacemos:

> MariaDB[(none)]> CREATE DATABASE mi_bd;

Con esto ya tenemos la posibilidad de que funcione nuestro script. Probar en un navegador: 192.168.56.10 y 192.168.56.10/index.php. En este último aparece el código del .php tal cual!!!. Lo que tenemos que hacer

indicarle que interprete el archivo: editamos el archivo de confiración de httpd.conf y modificamos:

```
DirectoryIndex ---> Añadimos al final *.php
```

Recargamos el sistema.

```
systemctl restart httpd
```

Y hacemos:

```
php -a /var/www/html/index.php
```

Si nos sale un error, será porque no tenemos el paquete

```
sudo dnf install php-mysqli
```

Comprobar otra vez con php -a si hay error.

A partir de aquí, hacer:

```
getsebool -a | grep httpd
```

```
sudo setsebool -P httpd_can_network_conect_db on
```

LAMP en Ubuntu

1. Instalar apache2

   ```
   sudo apt install apache2
   ```

2. Arrancar apache2:

   ```
   sudo systemctl start apache2
   ```

3. Probamos a poner la url en un navegador y no nos saldrá nada. Tenemos que hacer:

   ```
   sudo ufw allow 80
   ```

4. Instalar mariadb-server y mariadb-client.

5. Arrancamos mariadb:

   ```
   sudo systemctl start maridb
   ```

6. Probamos a entrar al intérprete de Mariadb:

   ```
   sudo mysql -u root -p
   ```

7. Ejecutamos:

   ```
   sudo mysql_secure_installation
   ```

8. Instalamos php:

   ```
   sudo apt install php
   ```

9. Instalamos php:

   ```
   sudo apt install php
   ```

   Probamos el intérprete de php:

   ```
   php -a
   ```

10. Nos conectamos remotamente desde el ordenadoor anfitrión (como tenemos que copiar y eso, es más facil desde nuestra terminal).

11. Creamos el archivo index.php:

    ```
    cd /var/www/html
    ```

    ```
    sudo vi index.php
    ```

    ```
    rellenamos el archivo.
    ```

12. Nos vamos al intérprete de mariadb:

    ```
    sudo mysql -u root -p
    ```

13. Creamos y rellenamos la base de datos (código de swad).

14. Comprobamos lo que sale en el navegador al poner:

    ```
    192.168.56.15/index.php
    ```

15. Si sale mal, modificar el Directory index del archivo /etc/apache2/conf/apache.conf, poniendo *.php

16. Hacer:

    ```
    systemctl restart apache2
    ```