01/07/23

**UPNA - Pamplona**

**Calves públicas SSH en LDAP y registro de accesos Linux**

*Iban Ruiz de Galarreta Cadenas*

08

**Fall**

TABLA DE CONTENIDOS

[Ev. 1. Ut 1: CARACTERIZACIÓN DE SISTEMAS OPERATIVOS 3](#_Toc491007477)

[2. Sistema informático 3](#_Toc491007478)

[3. Clasificación de los sistemas operativos 4](#_Toc491007479)

[4. Los gestores de arranque 5](#_Toc491007480)

[5. Gestión del procesador 6](#_Toc491007481)

[Índice 6](#_Toc491007482)

# Introducción

En este proyecto, se ha desarrollado un sistema que combina el uso del módulo de autenticación PAM de Linux con un servidor LDAP para lograr una centralización de conexiones en servidores Linux de una red. La infraestructura se ha desplegado utilizando Docker y Docker Compose, aprovechando las imágenes existentes y personalizadas para configurar un servidor OpenLDAP y clientes Ubuntu. La implementación ha incluido la creación y configuración de claves públicas en LDAP, así como la automatización de la recepción y almacenamiento de estas claves en los clientes.

# Objetivos

# Implementación

## Despliegue de la infraestructura

En este apartado, voy a explicar cómo se ha preparado la infraestructura. Para todo el despliegue, se ha utilizado **Docker**, más concretamente **Docker Compose**. Se ha empleado la imagen **"osixia/openldap:latest"** para el servidor LDAP y la imagen **"ubuntu:latest"** para crear una imagen personalizada llamada **"ibantxu12/uldapyssh"**. Toda la configuración de las máquinas se ha programado en el archivo **"docker-compose.yaml"** y creado dos scripts: **"./start-app.sh"** y **"./stop-app.sh"**, para facilitar la creación y eliminación de las máquinas.

A continuación, explicaré detalladamente la creación y configuración de las maquinas:

### Servidor OpenLDAP

Esta es la configuración utilizada para el servidor LDAP:

A screen shot of a computer

Description automatically generated

Imagen 1 : Docker compose LDAP

Se ha elegido esta **imagen** poque ya disponía del servidor openldap instalado y me permitía no tener que instalar y configurar todo el servidor.

Para el **container\_name** y el **hostname** se ha decidido usar el mismo nombre, para poder identificarlos bien dentro y fuera de las maquinas.

El apartado de **volumes** guardada la ruta donde se van a compartir archivos, como por ejemplo los que configuran los usuarios y no haya que configurarlo cada vez que cierras las máquinas. Es posible que en este apartado en el futuro necesite añadirle nuevos volúmenes, como los certificados o los ajustes de PAM.

El apartado **networks** selecciona la red donde van a estar todas las maquinas, la cual se ha creado más abajo y es **“openldap-n”**.

Y, por último, en el apartado **environment** se han configurado varias variables que ahora mismo no son muy necesarias, pero puede que en un futuro me sea útil, como toda la parte del certificado. La información que sí que es importante es **“LDAP\_DOMAIN”** y **“LDAP\_BASE\_DN”** que tienen que coincidir y la información de login del admin, es decir **“LDAP\_ADMIN\_USERNAME”** y **“LDAP\_ADMIN\_PASSWORD”**.

Los usuarios se crean automáticamente en el **“start-app.sh”**, como se ve, tenemos el comando **"ldapadd -x -D cn=admin,dc=ibantfg,dc=com -W -f user.ldif>"** y otro para los grupos. Si queremos, cambiar, añadir o eliminar los usuarios solo tendremos que modificar estos archivos, por defecto crea los usuarios: : Javier, Mark, Lorea, Maite, Olatz, Mikel, Mario. Y los grupos: 1, 2 y 3. Podemos conectarnos al servidor LDAP con el comando **"docker exec -it openldap bash"** y así comprobar los usuarios con el comando **“slapcat”**.

### Clientes Ubuntu

Para crear los clientes se ha elaborado una imagen con las herramientas necesarias de la siguiente manera:

Primero se ha arrancado una imagen normal de “Ubuntu:latest”, se ha accedido a ella **"docker exec -it ubuntu bash"** e instalado todo lo necesario: **“sudo apt update”**, **“sudo apt install ssh”** y **“sudo apt-get install libnss-ldap libpam-ldap ldap-utils -y”**. después del último comando se abrirá un menú para configurar el cliente LDAP. En el primer apartado se ha escrito la dirección de red del servidor **“ldapi:///openldap/”**, en el siguiente lo que se configuró en el servidor LDAP **“dc=ibantfg,dc=com”**. Después se ha seleccionado la versión 3 de LDAP. Por último, el usuario y la contraseña del admin, es decir: **“cn=admin,dc=ibantfg,dc=com”** y **“LDAPapTFG”**.

Una vez instalado openldap se ha configurado manualmente tres archivos, uno es **“/etc/nsswitch.conf”**, en el cual se ha cambiado systemd por ldap en las líneas 7,8 y 9. El otro es **“/etc/pam.d/common-session”** al que habrá que añadir al final del archivo la línea **“session optional       pam\_mkhomedir.so skel=/etc/skel umask=077”**.

Con esto la maquina ya estaba preparada y se podía conectar a un usuario del servidor LDAP. Esa máquina se ha subido al repositorio de Docker con el nombre **“ibantxu12/uldapyssh”** para que sea accesible desde cualquier equipo. Y esta es la configuración para cada cliente:

A screen shot of a computer

Description automatically generated

Imagen 2: Docker compose Clientes

Como ya he dicho uso la imagen **“ibantxu12/uldapyssh”** que se ha creado, se ha puesto el mismo nombre de **container\_name** y de **hostname**. Se ha añadido a la red **openldap\_n**. Luego, se ha mapeado el puerto 22 en el puerto 221 de mi maquina física, pero solo para el cliente 1, el cliente 2 mapeare el puerto 222, para el 3 el 223… Y por último se ha añadido el comando **“/usr/sbin/sshd -D”** para que el servidor ssh esté funcionando y para que la maquina no se cierre automáticamente.

### Prueba de ejecución

Para comprobar el correcto funcionamiento de la red podemos conectarnos que desde cualquier usuario a cualquiera de las maquinas. Con el comando “**ssh <usuario>@localhost -p <puerto>"** donde **<usuario>** es un usuario cualquiera del servidor LDAP y **<puerto>** es el puerto de la maquina a la que conectarse ( número entre 221 y 225). Los usuarios son javier, mark, lorea, maite, olatz, mikel y mario. Y las contraseñas **“<usuario>Pass”**. Aquí tenemos un ejemplo:

A screenshot of a computer screen

Description automatically generated

Imagen 3: Ejemplo ejecución

## *Claves públicas en LDAP*

### Creación de las claves públicas

Se han creado todas las claves publicas de los usuarios de ejemplo (la clave publica y la privada) dentro de la carpeta compartida con LDAP. Esto se ha realizado mediante el comando **“ssh-keygen -t rsa”**. A estos archivos no se les ha puesto contraseña. Este es un ejemplo de la creación:

A screenshot of a computer

Description automatically generated

Imagen 4: Creación de claves

### Configurar las calves públicas en el servidor LDAP

Para almacenar las claves públicas en LDAP primero se pensó en utilizar el atributo “sshPublicKey” pero resulto no estar habilitado en esta versión de OpenLDAP, como podemos observar en el siguiente error:

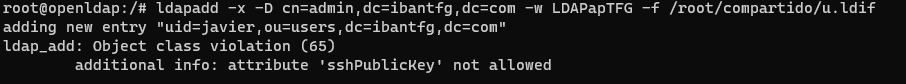


Imagen 5: Error sshPublicKey

Como el error persistía aun que se intentase modificar el esquema se decidió añadir un nuevo atributo, se creó un archivo .ldif que añadiese un atributo gracias al comando **“ldapmodify -Y EXTERNAL -H ldapi:/// -f /root/compartido/pu.ldif”**.

A screenshot of a computer program

Description automatically generated

Imagen 6: Nuevo atributo

Con esto se obtenía lo mismo, “not allowed”, en las dos siguientes imágenes (7 y 8) podemos ver el atributo antes y después de crearse. Se probo a añadirlo en distintos puntos del esquema del LDAP, pero sin ningún resultado.

A screen shot of a computer

Description automatically generated

Imagen 7: Antes de crearlo

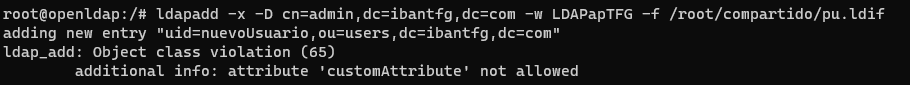


Imagen 8: Después de crearlo

Como no se conseguía agregar el atributo se opto por usar un atributo ya creado, el mejor candidato era el atributo “description”. Para conseguir esto se ha añadido una línea a la creación de los usuarios llamada “description” con la clave pública.

A computer screen with white text

Description automatically generated

Imagen 9: Clave añadida

### Script para recibir las claves

Para que los clientes reciban las claves se ha creado un script que se ejecuta en el cliente, encargado de añadir las claves a la carpeta correspondiente. En el siguiente apartado se explicará cómo se ejecuta este script. En este explicare el funcionamiento del script:

A screen shot of a computer

Description automatically generated

Imagen 10: Variables globales

Primero se han establecido las variables locales, que son las mismas que se han configurado en el servidor. Para aumentar la seguridad se debería crear un usuario en el servidor LDAP con los permisos justos para no usar el usuario admin. La última variable es la ubicación de los usuarios en la estructura, se ha creado por si se establecen usuarios de otras clases.

A computer screen with text on it

Description automatically generatedImagen 11: Bucle y carpetas

A computer screen shot of a computer code

Description automatically generatedEl resto del script es un bucle que recorre los registros de LDAP gracias a la primera línea de la imagen 11. Dentro del bucle, en las primeras líneas se van sacando uno por uno los usuarios y el primer grupo al que pertenecen. Después, en el IF de abajo creamos la carpeta del usuario si no esta creada y le añadimos los permisos necesarios.

Imagen 12: Claves publicas

Por último, saca del LDAP la clave publica almacenada en el servidor, y dentro del IF se crean las carpetas necesarias, con sus respectivos permisos para almacenar la clave publica en el servidor. Como podemos ver, la clave publica machaca el archivo, solo podemos tener una clave privada por usuario.

Ejecutando este script se crearían todas las carpetas de los usuarios en ese servidor y se añadiría las claves publicas almacenadas en ldap, y ahora podríamos logarnos en el servidor con la clave, por ejemplo **“ssh -i keys/maite maite@localhost -p 221”**.

### Ejecución del script

Para que el script se ejecute automáticamente cuando sea necesario entra en juego PAM. Se ha creado una regla que ejecuta este script cada vez que un cliente intenta conectarse por SSH. Esto lo hemos logrado modificando el archivo **“/etc/pam.d/sshd”** para que estos cambios se guarden y para tener el script a mano hemos modificado el archivo de configuración de las maquinas.

A screen shot of a computer

Description automatically generated

Imagen 13: Volúmenes clientes

Como se puede observar se ha añadido el archivo volúmenes con la carpeta scripts y la configuración de PAM. Al archivo de configuración basta con añadirle la siguiente línea al principio del archivo para que todo funcione correctamente.

A screen shot of a computer screen

Description automatically generated

Imagen 14: Configuración PAM

**“auth”** Indica que estamos creando una regla de autenticación. **“sufficient”** Indica que esta regla es suficiente para permitir la autenticación. Si se cumple esta regla, no se verificarán más reglas después de ella. Es importante escribir esto en lugar de “required”. **“pam\_exec.so”** Es el módulo PAM que se está utilizando. En este caso, permite ejecutar un comando externo durante la autenticación. Y después tenemos el script que queremos ejecutar.

De primeras parecía que esta solución funcionaba, pedía contraseña, pero poniendo cualquier palabra se conseguía iniciar sesión, pero nada mas lejos de la realidad. Esto funcionaba porque “sufficient” inicia sesión si el script se ejecuta, es decir, que daba igual la clave que usases o la contraseña que pusieras que se iba a iniciar sesión. Después de investigar posibles soluciones ninguna de ellas funcionaba, porque tanto PAM como la propia configuración de SSH no ejecuta ningún script hasta que se inicie la sesión. Por lo tanto, la única opción posible era programar la ejecución del script periódicamente. Para esto he utilizado **“cron”**, este es archivo de configuración.

**A screen shot of a computer program

Description automatically generated**

Imagen 15: Configuración cron

Esto quiere decir que se va a ejecutar el script cada minuto. Para que cron se ejecute hay que iniciar el servicio a la vez que el de SSHD, como Docker compose solo permite ejecutar un comando a la vez se ha tenido que crear un archivo compartido con los clientes que ejecuta los dos servicios.

A screen shot of a computer

Description automatically generated

Imagen 16: Script servicios

En el **“command”** del yaml se ha añadido la ejecución de este script **“command: ["/usr/local/bin/scripts/start\_services.sh"]”**. Con esto ya se ejecutara el script y podremos acceder con la clave SSH.

## Registro de accesos y alertas

### Servicio de logs centralizado

Para tener centralizados los registros a los servidores se pensó en usar el propio servicio de OpenLDAP “auditEvent” para habilitar esto se ha creado el archivo “audit\_config.ldif”.

A screen shot of a computer code

Description automatically generated

Imagen 17: Audit\_config.ldif

Y ejecutado el siguiente comando en el servidor LDAP.



Imagen 18: Comando auditoria

Después se ha creado un script que creaba el archivo de log y lo mandaba al servidor LDAP.

A computer screen shot of text

Description automatically generated

Imagen 17: Script logs

Cuando se ha probado se a obtenido el error que obtenemos en la siguiente imagen. Esto se debe a que nuevamente el registro “auditEvent” no se encuentra en el LDAP y como ya hemos comentado anteriormente no podemos agregar registros.

A screen shot of a computer program

Description automatically generated

Imagen 18: Error auditoria

Por eso se ha optado por crear otro servicio encargado de recibir los logs de los clientes. Normalmente se añadiría este servicio al servidor LDAP, pero por facilitar la configuración de Docker se ha creado una maquina nueva encargada de recibir los logs.

# Pruebas

# Conclusiones

La implementación exitosa de un sistema de registro de accesos y generación de alertas en tu infraestructura de servidores Linux centralizados proporcionará una mayor seguridad y control sobre las conexiones y actividades en la red.