01/07/23

**UPNA - Pamplona**

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

*Iban Ruiz de Galarreta Cadenas*

08

**Fall**

Contents

[1. Introducción 3](#_Toc144140373)

[1. Tecnologías elegidas 3](#_Toc144140374)

[1.1. LDAP 3](#_Toc144140375)

[1.2. Docker Compose 5](#_Toc144140376)

[1.3. SSH 6](#_Toc144140377)

[2. Objetivos 8](#_Toc144140378)

[2.1. Integración de Claves Públicas SSH en LDAP 8](#_Toc144140380)

[2.2. Registro de Accesos y Generación de Alertas 8](#_Toc144140381)

[3. Implementación 9](#_Toc144140382)

[1. Despliegue de la infraestructura 9](#_Toc144140383)

[1.1. Servidor OpenLDAP 9](#_Toc144140384)

[1.2. Clientes Ubuntu 12](#_Toc144140385)

[1.3. Prueba de ejecución 14](#_Toc144140386)

[*2.* *Claves públicas en LDAP* 15](#_Toc144140387)

[2.1. Creación de las claves públicas 15](#_Toc144140388)

[2.2. Configurar las calves públicas en el servidor LDAP 16](#_Toc144140389)

[2.3. Script para recibir las claves 18](#_Toc144140390)

[2.4. Ejecución del script 20](#_Toc144140391)

[3. Registro de accesos y alertas 22](#_Toc144140392)

[3.1. Servicio de registro centralizado 22](#_Toc144140393)

[3.2. Alertas de anomalías 25](#_Toc144140394)

[4. Pruebas 28](#_Toc144140395)

[4.1. Pruebas configuración inicial 28](#_Toc144140397)

[4.2. Pruebas claves SSH 29](#_Toc144140398)

[4.3. Prueba archivo de registro 32](#_Toc144140399)

[4.4. Prueba alertas 33](#_Toc144140400)

[5. Conclusiones 34](#_Toc144140401)

[6. Referencias 36](#_Toc144140402)

# 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.

No obstante, el alcance de este proyecto no se detiene aquí. Una vez establecida la infraestructura y la gestión de claves, se ha añadido una capa adicional de seguridad y monitoreo: la creación de un sistema de registro de accesos y alertas. A través de la configuración del módulo PAM, se ha habilitado la generación de registros detallados de cada acceso a los servidores Linux. Estos registros contienen información crucial como la fecha, el usuario y el servidor de destino. Otra aportación radica en la capacidad del sistema para generar alertas en tiempo real ante actividades inusuales o potencialmente peligrosas.

## Tecnologías elegidas

### LDAP

LDAP (Lightweight Directory Access Protocol) es uno de los principales protocolos de autenticación que se desarrolló por los servicios de directorio. LDAP históricamente se ha usado como una base de datos de información, como por ejemplo usuarios, sus atributos, privilegios… Esta información se utiliza posteriormente para permitir la autenticación a los sistemas tales como una aplicación o servidor.

La autenticación entre un cliente y un servidor se realizaría de la siguiente manera. un cliente envía una solicitud para requerir información almacenada dentro de una base de datos LDAP junto con las credenciales del usuario a un servidor LDAP. El servidor LDAP autentifica las credenciales enviadas por los usuarios contra la identidad principal del usuario, la cual se almacena en la base de datos LDAP. Si las credenciales enviadas por el usuario coinciden con las credenciales asociadas con la identidad principal del usuario que se encuentra almacenada dentro de la base de datos LDAP, se le permite al usuario el acceso y recibe la información requerida (atributos, membresías de grupo y otra información). Si las credenciales enviadas no coinciden, se le niega el acceso a la base de datos LDAP.

En este proyecto, se ha empleado el protocolo LDAP para lograr una centralización efectiva en la autenticación y la administración de claves públicas en entornos de servidores Linux. Esta centralización ha demostrado ser una herramienta valiosa para gestionar de manera organizada y segura las identidades y las credenciales de acceso en toda la infraestructura de red. Se ha configurado un servidor OpenLDAP que almacena y organiza la información de los usuarios y grupos en una estructura jerárquica.

El uso del protocolo LDAP ha permitido implementar una solución escalable y ordenada para manejar las claves públicas SSH de los usuarios. Estas claves públicas se almacenan de manera eficiente como atributos en las entradas de los usuarios en el servidor LDAP, lo que posibilita una autenticación segura y eficaz a través del protocolo SSH.

Adicionalmente, se ha aprovechado las capacidades del protocolo LDAP para establecer un sistema de registro de accesos y alertas en tiempo real. Esta capa adicional de seguridad ha permitido monitorizar de cerca la actividad en los servidores, generando alertas inmediatas ante cualquier comportamiento sospechoso o potencialmente riesgoso.

En resumen, el protocolo LDAP ha sido un pilar fundamental en este proyecto, permitiendo centralizar la autenticación, gestionar claves y supervisar los accesos en servidores Linux. La estructura jerárquica y la versatilidad para almacenar atributos han sido explotadas de manera efectiva para crear una solución completa y segura en la administración de servidores dentro de una red.

### Docker Compose

Docker es una plataforma de código abierto que permite a los desarrolladores crear, desplegar, ejecutar y gestionar contenedores, que son componentes estandarizados y ejecutables que combinan el código fuente de aplicación con las dependencias y las bibliotecas del sistema operativo (SO) necesarias para ejecutar dicho código en cualquier entorno.

Los contenedores simplifican el desarrollo y la entrega de las aplicaciones distribuidas y han ido ganando popularidad con la transición de las organizaciones hacia entornos multicloud híbridos y de desarrollo nativos en cloud. Los desarrolladores pueden crear contenedores sin Docker trabajando directamente con las funciones integradas en Linux y otros sistemas operativos, pero la contenerización que realiza Docker es más rápida, fácil y segura.

Mas en concreto Docker Compose, es una herramienta para definir y ejecutar aplicaciones Docker multicontenedor. Permite definir los servicios, las redes y los volúmenes de la aplicación en un único archivo, y poner en marcha la aplicación con un único comando.

En el proyecto, se ha empleado Docker Compose para lograr la orquestación eficiente de contenedores y servicios. Mediante la definición de un archivo de configuración en formato YAML, se han establecido los diferentes servicios, redes y volúmenes requeridos para el funcionamiento conjunto de las aplicaciones en contenedores.

La utilización de Docker Compose ha permitido simplificar el despliegue y la administración de múltiples contenedores interdependientes. Mediante una única instrucción, se pueden iniciar y detener todos los servicios relacionados, lo que agiliza significativamente el proceso de desarrollo, pruebas y puesta en producción.

Además, Docker Compose ha facilitado la configuración de redes personalizadas para los contenedores, lo que ha posibilitado la comunicación segura y eficiente entre los diferentes servicios. Esto ha contribuido a la creación de un entorno aislado y controlado para las aplicaciones, optimizando la seguridad y el rendimiento.

En resumen, Docker Compose ha sido una herramienta esencial en este proyecto al permitir la definición, el despliegue y la gestión simplificada de múltiples contenedores interconectados. Su capacidad para configurar redes y volúmenes ha mejorado la seguridad y la eficiencia en el funcionamiento de las aplicaciones en contenedores.

### SSH

SSH es un protocolo de administración remota que le permite a los usuarios controlar y modificar sus servidores remotos a través de Internet a través de un mecanismo de autenticación. Proporciona un mecanismo para autenticar un usuario remoto, transferir entradas desde el cliente al host y retransmitir la salida de vuelta al cliente. Esta tecnología utiliza técnicas criptográficas para garantizar que todas las comunicaciones hacia y desde el servidor remoto sucedan de manera encriptada.

SSH opera en el puerto TCP 22 de forma predeterminada (aunque esto se puede cambiar si es necesario). El host (servidor) escucha en el puerto 22 (o cualquier otro puerto SSH asignado) para las conexiones entrantes. Organiza la conexión segura mediante la autenticación del cliente y la apertura del entorno de shell correcto si la verificación tiene éxito.

En el proyecto, se ha integrado el protocolo SSH (Secure Shell) como parte esencial de la infraestructura para establecer conexiones seguras y encriptadas en redes. Mediante la generación y gestión de pares de claves públicas y privadas, se ha habilitado un método de autenticación robusto que permite el acceso remoto confiable a servidores Linux.

En el proceso de desarrollo, se llevaron a cabo creaciones de pares de claves públicas y privadas. La clave privada se mantiene resguardada con cifrado en la máquina de origen, garantizando así la protección de la información. En contraste, la clave pública se distribuye en los servidores remotos de destino. Este enfoque no solo simplifica el proceso de inicio de sesión, sino que también refuerza la seguridad al eliminar la necesidad de utilizar contraseñas.

Además, se efectuaron configuraciones en los archivos de ajustes del servicio SSH, permitiendo la delimitación precisa de accesos para usuarios y grupos específicos. Esto concede la capacidad de establecer políticas de seguridad particulares, otorgando un control detallado sobre las autorizaciones de acceso y las operaciones que pueden llevar a cabo una vez autenticados.

En resumen, la integración del protocolo SSH en el proyecto ha desempeñado una función fundamental al proporcionar un medio seguro y eficaz para administrar servidores Linux de forma remota.

# Objetivos

Los dos principales objetivos de mi proyecto son los siguientes:



### Integración de Claves Públicas SSH en LDAP

La primera parte del proyecto se enfoca en la configuración de un servidor LDAP y la inclusión de claves públicas SSH en dicho servidor. A través del módulo PAM, se implementará un sistema que automatice la extracción de las claves públicas de los usuarios y las almacene en el servidor LDAP. Se explorarán diferentes enfoques para lograr esta integración y se analizarán las ventajas de utilizar un atributo específico o una alternativa como "description" para gestionar estas claves.

### Registro de Accesos y Generación de Alertas

La segunda fase del proyecto se centra en el registro y monitorización de los accesos a los servidores Linux. Mediante la configuración del módulo PAM, se llevará a cabo la creación de un sistema que registre los accesos de los usuarios a los servidores y genere alertas en tiempo real en caso de actividades sospechosas o no autorizadas. Para lograr esto, se establecerá un proceso que analice y registre los detalles de cada acceso, como la fecha, el usuario y el servidor al que se accede.

En conjunto, este proyecto busca optimizar la administración y seguridad de los servidores Linux en una red al centralizar la gestión de claves públicas y el registro de accesos. La implementación de estas funciones a través del módulo PAM brindará una solución integral para mantener un control eficaz sobre las conexiones a los sistemas, detectar posibles amenazas y tomar medidas preventivas para salvaguardar la integridad y la confidencialidad de la red.

# Implementación

## Despliegue de la infraestructura

El presente apartado tiene como objetivo describir el proceso de configuración de la infraestructura necesaria para el despliegue de la solución. Para llevar a cabo esta tarea, se ha empleado la herramienta **Docker**, en particular, la herramienta **Docker Compose**. Con el propósito de establecer el entorno requerido, se ha utilizado la imagen **"osixia/openldap:latest"** para el servidor LDAP, y la imagen **"ubuntu:latest"** ha sido utilizada para construir una imagen personalizada denominada **"ibantxu12/uldapyssh"**. Toda la configuración detallada de las máquinas ha sido gestionada y especificada en el archivo **"docker-compose.yaml"**. Adicionalmente, se ha implementado la conveniencia de simplificar el proceso de creación y eliminación de las máquinas mediante la creación de dos scripts: **"./start-app.sh"** y **"./stop-app.sh"**.

A continuación, se procederá a presentar de manera exhaustiva la secuencia de pasos involucrados en la creación y configuración de las máquinas necesarias para este proyecto.

### Servidor OpenLDAP

La configuración empleada para la puesta en marcha del servidor LDAP se detalla a continuación:

A screen shot of a computer

Description automatically generated

Imagen 1 : Docker compose LDAP

Se optó por utilizar la imagen proporcionada **"osixia/openldap:latest"** debido a que ya contenía una instalación funcional del servidor OpenLDAP. Esta elección permitió evitar la tediosa tarea de instalación y configuración desde cero.

Dentro del archivo de composición de Docker **"docker-compose.yaml"**, se establecieron los siguientes parámetros para el servidor LDAP:

* **container\_name** y **hostname**: Ambos valores se fijaron con el mismo nombre, con la finalidad de posibilitar la fácil identificación de la máquina tanto interna como externamente.
* **volumes**: Se especificó la ruta del directorio donde se compartirían archivos. En particular, esta ruta sería útil para almacenar archivos de configuración de usuarios, evitando tener que reconfigurar cada vez que las máquinas se cierran. En un futuro, es posible que se deban añadir nuevos volúmenes para certificados u otros ajustes relacionados con PAM.
* **networks**: Se designó la red en la cual estarían ubicadas todas las máquinas. Esta red, denominada **"openldap-n"**, fue definida más adelante en la configuración.
* **environment**: Se establecieron varias variables de entorno que, aunque en la actualidad no son imperativas, podrían ser de utilidad en futuras situaciones. Entre estas variables, dos son particularmente relevantes: **"LDAP\_DOMAIN"** y **"LDAP\_BASE\_DN"**, las cuales deben coincidir. Asimismo, se proporcionó información de inicio de sesión para el administrador, es decir, **"LDAP\_ADMIN\_USERNAME"** y **"LDAP\_ADMIN\_PASSWORD"**.

La creación de usuarios ha sido automatizada mediante el uso del script **"./start-app.sh"**. En este script, se incluyó el comando **"ldapadd -x -D cn=admin,dc=ibantfg,dc=com -W -f user.ldif"** para la adición de usuarios y los grupos. Esto presenta una gran ventaja, ya que modificar, añadir o eliminar usuarios se vuelve una tarea sencilla y centralizada, simplemente al editar estos archivos de configuración.

Por defecto, el script de creación de usuarios establece la creación de los siguientes usuarios: Javier, Mark, Lorea, Maite, Olatz, Mikel y Mario. Además, también se configuran los grupos 1, 2 y 3.

En caso de requerir cambios en la lista de usuarios o grupos, tan solo será necesario modificar los archivos pertinentes en el script. Este modularidad facilita enormemente la administración y gestión de usuarios en el servidor LDAP.

Una vez que la infraestructura se encuentra en funcionamiento, es posible verificar la existencia y los detalles de los usuarios en el servidor LDAP. Esto se logra a través del comando **"docker exec -it openldap bash"** para acceder al servidor LDAP y, una vez dentro, ejecutar el comando **"slapcat"** para visualizar la información de los usuarios almacenada en el servidor. Esto permite una verificación sencilla y directa de la configuración y los usuarios creados en el servidor LDAP.

### Clientes Ubuntu

Para la creación de los clientes, se siguió un proceso estructurado que garantiza la instalación adecuada de las herramientas necesarias y la configuración de los servicios para lograr la conexión con el servidor LDAP. A continuación, se detalla paso a paso el proceso de creación de los clientes:

1. Se inició utilizando una imagen base de **"Ubuntu:latest"** en Docker.
2. Se accedió a la imagen de Ubuntu utilizando el comando **"docker exec -it ubuntu bash"**.
3. Se realizó la instalación de los componentes requeridos mediante los siguientes comandos:
   1. **"sudo apt update"**
   2. **"sudo apt install ssh"**
   3. **"sudo apt-get install libnss-ldap libpam-ldap ldap-utils -y"**
4. Después de ejecutar el último comando, se abrió un menú de configuración para el cliente LDAP. En este menú se realizaron las siguientes configuraciones:
   1. Se introdujo la dirección de red del servidor LDAP en el campo correspondiente, como **"ldapi:///openldap/"**.
   2. Se configuró el dominio LDAP según lo establecido en el servidor, en este caso **"dc=ibantfg,dc=com"**.
   3. Se seleccionó la versión 3 de LDAP.
   4. Se proporcionaron las credenciales del administrador del servidor LDAP, es decir, **"cn=admin,dc=ibantfg,dc=com"** y su contraseña **"LDAPapTFG"**.
5. Una vez instalado OpenLDAP, se procedió a la configuración manual de tres archivos importantes en el cliente:
   1. Se modificó el archivo **"/etc/nsswitch.conf"** para cambiar **"systemd"** por **"ldap"** en las líneas 7, 8 y 9. Esto asegura que el sistema utilice LDAP para la resolución de nombres.
   2. Se editó el archivo **"/etc/pam.d/common-session"** y se añadió al final del archivo la línea **"session optional pam\_mkhomedir.so skel=/etc/skel umask=077"**. Esta línea permite crear automáticamente el directorio de inicio del usuario en caso de que no exista.
6. Con todas estas configuraciones, la máquina cliente quedó preparada y lista para conectarse al servidor LDAP. Para facilitar el acceso y compartir la máquina, se subió la imagen resultante al repositorio de Docker con el nombre **"ibantxu12/uldapyssh"**.

En resumen, el proceso de creación de clientes se basó en la instalación y configuración de los componentes necesarios para la autenticación y la conexión con el servidor LDAP. Cada máquina cliente resultante está lista para conectarse a la infraestructura, y su configuración se simplifica gracias a la imagen compartida en el repositorio de Docker. Esta es la configuración para cada cliente:

A screen shot of a computer

Description automatically generated

Imagen 2: Docker compose Clientes

Se utilizó la imagen personalizada **"ibantxu12/uldapyssh"** para las máquinas clientes. Cada cliente tiene el mismo nombre para **"container\_name"** y **"hostname"** para facilitar la identificación. Se conectaron a la red **"openldap\_n"** para la comunicación con el servidor LDAP. Se mapeó el puerto 22 de cada cliente a puertos individuales en la máquina física (por ejemplo, puerto 221 para el primer cliente, 222 para el segundo…). Se añadió el comando **"/usr/sbin/sshd -D"** para mantener activo el servidor SSH en las máquinas cliente y evitar su cierre automático. Esto asegura que las máquinas estén disponibles para recibir conexiones SSH centralizadas y seguras.

### Prueba de ejecución

Para verificar la operatividad de la red, puedes conectarte desde cualquier usuario a cualquiera de las máquinas mediante el comando “**ssh <usuario>@localhost -p <puerto>"** donde **<usuario>** es un usuario del servidor LDAP, y **<puerto>** es el número asignado a la máquina a la que deseas conectarte (por ejemplo, entre 221 y 225). Los nombres de usuario disponibles son: javier, mark, lorea, maite, olatz, mikel y mario. Las contraseñas corresponden a “**<usuario>Pass”**. Aquí tienes un ejemplo de cómo se realizaría:

A screenshot of a computer screen

Description automatically generated

Imagen 3: Ejemplo ejecución

Esto te permitirá acceder a las máquinas de forma centralizada y segura mediante SSH, utilizando las credenciales de los usuarios almacenados en el servidor LDAP.

## *Claves públicas en LDAP*

### Creación de las claves públicas

Se han generado claves públicas y privadas para los usuarios de ejemplo, y se han guardado en la carpeta compartida con LDAP. Esto se logró utilizando el comando **"ssh-keygen -t rsa"**, y las claves generadas no tienen contraseñas asociadas.

Al ejecutar este comando, se generarán las claves pública y privada para cada usuario, y se almacenarán en la carpeta correspondiente para su uso posterior en la autenticación SSH.

A screenshot of a computer

Description automatically generated

Imagen 4: Creación de claves

### Configurar las calves públicas en el servidor LDAP

En un principio, se consideró la posibilidad de utilizar el atributo **"sshPublicKey"** para almacenar las claves públicas en LDAP. Sin embargo, se descubrió que este atributo no estaba habilitado en la versión de OpenLDAP utilizada, lo cual se evidenció mediante el siguiente mensaje de error:

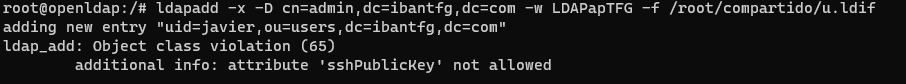


Imagen 5: Error sshPublicKey

Dado que el problema persistía a pesar de los intentos de modificar el esquema, se tomó la decisión de introducir un nuevo atributo. Se procedió a crear un archivo .ldif con la intención de añadir este atributo mediante el comando **"ldapmodify -Y EXTERNAL -H ldapi:/// -f /root/compartido/pu.ldif"**.

A screenshot of a computer program

Description automatically generated

Imagen 6: Nuevo atributo

A pesar de este enfoque, el problema persistía y seguía apareciendo el mensaje "not allowed". Las imágenes 7 y 8 muestran el atributo antes y después de su creación. Aunque se intentó agregar el atributo en diferentes ubicaciones del esquema del LDAP, no se lograron resultados positivos en el proceso.

A screen shot of a computer

Description automatically generated

Imagen 7: Antes de crearlo

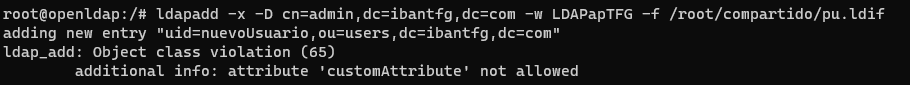


Imagen 8: Después de crearlo

Dado que no se pudo lograr la adición del nuevo atributo, se tomó la decisión de utilizar un atributo existente, y el más adecuado resultó ser "description". Para lograr esto, se incluyó una línea adicional en la creación de usuarios que agregaba la clave pública al atributo "description".

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 desarrollado un script que se ejecuta en cada cliente, encargado de agregar las claves a la carpeta correspondiente. En la siguiente sección, se detallará cómo se ejecuta este script. A continuación, explicaré el funcionamiento del script:

A screen shot of a computer

Description automatically generated

Imagen 10: Variables globales

Inicialmente, se definen variables locales en el script, coincidiendo con las mismas configuraciones establecidas en el servidor. Para reforzar la seguridad, sería recomendable crear un usuario en el servidor LDAP con permisos adecuados, evitando así el uso del usuario 'admin'. La última variable indica la ubicación de los usuarios en la estructura, lo cual se ha previsto en caso de que se establezcan usuarios de distintas categorías.

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 script continúa con un bucle que itera a través de los registros en LDAP, aprovechando la instrucción de la primera línea de la imagen 11. Dentro de este bucle, se extraen uno a uno los usuarios y el primer grupo al que están asignados. Luego, en el bloque IF que sigue, se procede a crear la carpeta del usuario en caso de que aún no exista y se le asignan los permisos necesarios.

Imagen 12: Claves publicas

Luego, el script extrae la clave pública del servidor LDAP y, dentro del bloque IF, crea las carpetas necesarias con los permisos correspondientes para almacenar la clave pública en el servidor. Es importante destacar que la clave pública sobrescribe el archivo existente, lo que significa que cada usuario solo puede tener una clave pública almacenada.

Al ejecutar este script, se generan todas las carpetas de los usuarios en el servidor y se añaden las claves públicas almacenadas en LDAP. Esto habilita la posibilidad de iniciar sesión en el servidor utilizando la clave pública, como se muestra en el ejemplo: **"ssh -i keys/maite maite@localhost -p 221"**.

### Ejecución del script

Para automatizar la ejecución del script cuando sea necesario, se ha utilizado el módulo PAM. Se ha configurado una regla que ejecuta este script cada vez que un cliente intenta conectarse a través de SSH. Para lograr esto, se ha realizado una modificación en el archivo **"/etc/pam.d/sshd"** para que los cambios se apliquen. Además, se ha ajustado el archivo de configuración de las máquinas para incluir el script necesario y tenerlo disponible.

A screen shot of a computer

Description automatically generated

Imagen 13: Volúmenes clientes

Se ha incorporado el archivo "volumes" que contiene la carpeta de scripts y la configuración de PAM al archivo de configuración. Para asegurar el correcto funcionamiento, solo es necesario agregar la siguiente línea al inicio del archivo de configuración.

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.

En un principio, esta solución parecía efectiva: solicitaba una contraseña al intentar iniciar sesión, pero resultaba que cualquier palabra permitía el acceso. Sin embargo, la realidad era más compleja. Esto funcionaba debido a que el uso de la directiva "sufficient" en PAM permitía iniciar sesión en caso de que el script se ejecutara, independientemente de la contraseña o clave proporcionada. Tras explorar diversas alternativas, ninguna de ellas resultó exitosa, ya que ni PAM ni la configuración nativa de SSH ejecutaban scripts antes de que se iniciara la sesión. En consecuencia, la única solución viable consistía en programar la ejecución periódica del script. Para lograr esto, se implementó el uso de **"cron"**, como se refleja en el archivo de configuración.

**A screen shot of a computer program

Description automatically generated**

Imagen 15: Configuración cron

Esta configuración implica que el script se ejecutará cada minuto mediante "cron". Deseamos ejecutar los servicios "cron" y "sshd" de manera simultánea. Sin embargo, dado que Docker Compose solo admite la ejecución de un solo comando a la vez, se ha implementado una solución que involucra la creación de un archivo compartido con los clientes. Este archivo ejecuta ambos servicios de manera simultánea.

A screen shot of a computer

Description automatically generated

Imagen 16: Script servicios

Se ha agregado la ejecución del script en el campo **"command"** del archivo YAML de configuración de Docker con la siguiente línea: **"command: ["/usr/local/bin/scripts/start\_services.sh"]"**. Esto asegurará que el script se ejecute y nos permitirá acceder utilizando la clave SSH.

## Registro de accesos y alertas

### Servicio de registro centralizado

Se ideó utilizar el servicio integrado de OpenLDAP llamado **"auditEvent"** para centralizar los registros en los servidores. Para habilitar esta función, se ha creado el archivo **"audit\_config.ldif"**.

A screen shot of a computer code

Description automatically generated

Imagen 17: Audit\_config.ldif

Luego se ejecutó el siguiente comando en el servidor LDAP.



Imagen 18: Comando auditoria

Después, se creó un script que generaba el archivo de registro y lo enviaba al servidor LDAP.

A computer screen shot of text

Description automatically generated

Imagen 17: Script logs

Cuando se probó, se encontró el error que se muestra en la imagen siguiente. Esto se debe a que una vez más el registro **"auditEvent"** no se encuentra en el LDAP y, como ya se discutió anteriormente, no podemos agregar registros.

A screen shot of a computer program

Description automatically generated

Imagen 18: Error auditoria

Por esta razón, se decidió crear otro servicio encargado de recibir los registros de los clientes. Aunque normalmente se añadiría este servicio al servidor LDAP, para simplificar la configuración en Docker se ha creado una nueva máquina dedicada a recibir los registros de los logs.

A computer screen shot of a black background

Description automatically generated

Imagen 19: Maquina de logs

Esta máquina utiliza Alpine debido a la presencia de NetCat en su instalación por defecto. Su función es escuchar en el puerto **514** y guardar los registros en el archivo **"/var/log/ssh.log"**. Este archivo puede ser accedido a través de un volumen. Por otro lado, se ha creado un script en los clientes para enviar la información a este servidor mediante NetCat. En este caso, fue necesario instalar NetCat en la imagen Docker personalizada de Ubuntu que se utilizó en los clientes.

A computer screen with text

Description automatically generated

Imagen 20: Script de logs

El archivo en cuestión simplemente envía información de las conexiones o desconexiones al servidor, incluyendo el nombre del usuario, la fecha y el nombre del servidor al que se ha accedido.

A black screen with white text

Description automatically generated

Imagen 21: Configuración PAM de logs

Luego, se ha agregado la última línea del archivo de configuración PAM de SSH tal como se ha visto en el apartado anterior, para que el script se ejecute con cada conexión al servidor. A continuación, se presenta un ejemplo del resultado obtenido en el archivo de registro:

A screenshot of a computer

Description automatically generated

Imagen 22: Ejemplo de logs

### Alertas de anomalías

La intención principal con las alertas era programar, con la ayuda de PAM, algunas de las alertas más utilizadas para analizar las conexiones SSH, que incluyen:

* Detectar intentos repetidos de inicio de sesión fallidos.
* Identificar cambios en archivos de configuración importantes.
* Alertar sobre el acceso a recursos restringidos por parte de usuarios no autorizados.
* Monitorizar actividad inusual en registros o archivos del sistema.
* Notificar sobre el consumo excesivo de recursos por parte de un usuario o proceso.

De entre estas situaciones, la única en la que podría parecer posible utilizar PAM es para detectar intentos fallidos de inicio de sesión. Sin embargo, PAM solo actúa en el caso de conexiones exitosas y no aborda los intentos fallidos. Por este motivo, se ha desarrollado un script que envía una alerta al servidor de logs si un usuario se conecta tres veces en el mismo minuto. A primera vista, esta solución podría parecer eficaz, aunque en realidad sería más conveniente abordar esta cuestión directamente en el servidor LDAP al aprovechar la auditoría que ya hemos configurado previamente. A pesar de esta opción, se ha optado por programar el script con el fin de proporcionar otro ejemplo de uso de PAM. Este enfoque podría resultar útil en situaciones donde no se requiere un registro detallado de cada inicio de sesión. Para lograr este propósito, se ha implementado el siguiente script.

A screen shot of a computer

Description automatically generated

Imagen 23: Script alertas

Este script genera un archivo que registra los accesos de los usuarios y luego verifica que no haya habido dos conexiones adicionales en el mismo minuto. Posteriormente, se ha incorporado la siguiente línea a la configuración PAM de SSH. Es importante ubicar esta línea justo encima de la directiva **"@include common-session"**.

A screen shot of a computer

Description automatically generated

Imagen 24: Configuración PAM loginAttempts

Esto envía una alerta al servidor de registros, como se puede observar en la siguiente imagen.

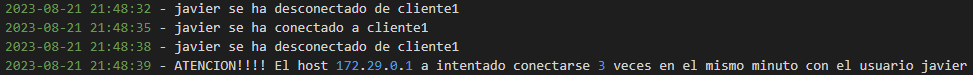


Imagen 25: Registro de alertas

# Pruebas



### Pruebas configuración inicial

Vamos a comenzar evaluando la configuración inicial. A continuación, se presentan las pruebas realizadas con diferentes usuarios de LDAP que intentan iniciar sesión con sus contraseñas en las diversas máquinas:

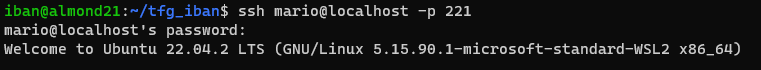


Imagen 26: Mario 221

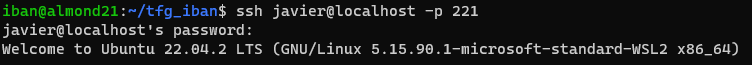


Imagen 27: Javier 221

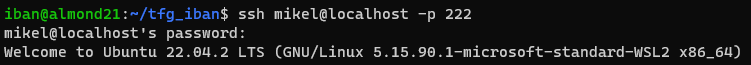


Imagen 28: Mikel 222

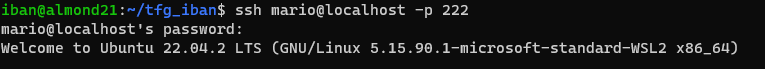


Imagen 29: Mario 222

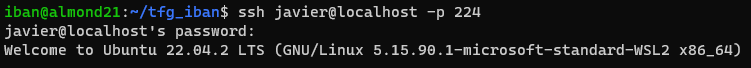


Imagen 30: Javier 224

### Pruebas claves SSH

En primer lugar, procederemos a realizar pruebas de conexión a cada servidor a través de SSH utilizando claves privadas y diferentes usuarios. Luego, verificaremos la presencia de las carpetas de usuarios en todos los servidores:



Imagen 31: Key Lorea 221

A black background with white text

Description automatically generated

Imagen 32: Key cliente1



Imagen 33: Key Mark 222

A black background with white text

Description automatically generated

Imagen 34: Key cliente2



Imagen 35: Key Olatz 223

A black background with white text

Description automatically generated

Imagen 36: Key cliente3



Imagen 37: Key Mikel 224

A black background with white text

Description automatically generated

Imagen 38: Key cliente4

A computer screen shot of white text

Description automatically generated

Imagen 39: Key Maite 225

A black background with white text

Description automatically generated

Imagen 40: Key cliente5

Hemos concluido las pruebas exhaustivas que nos aseguran el funcionamiento óptimo de todos los usuarios y clientes, confirmando que el servidor LDAP, el script, PAM y Cron están operando adecuadamente. Sin embargo, para una validación final, procederemos a realizar dos pruebas adicionales. Primero, consideraremos la posibilidad de modificar o añadir un usuario en la estructura de LDAP para evaluar si estos cambios se reflejan en los clientes. Para llevar a cabo esto, agregaremos el usuario Sonia. Además, exploraremos la actualización de las credenciales de usuario al asignarle a Maite la clave de Javier. Estos pasos complementarán nuestras pruebas y nos proporcionarán una certeza adicional sobre el funcionamiento integral de todo el sistema.

Se han elaborado estos dos archivos con el propósito de implementar las adiciones y ajustes mencionados.

A computer screen shot of a black screen

Description automatically generated

Imagen 41: LDIF añadir Sonia

A black screen with white text

Description automatically generated

Imagen 42: LDIF modificar Maite

En aras de simplificar el proceso, se ha configurado la clave de Sonia con la misma que posee Lorea. Para llevar a cabo las modificaciones, se emplearán los comandos **"ldapadd -x -D "cn=admin,dc=ibantfg,dc=com" -W -f modificacionSonia.ldif"** y **" ldapmodify -x -D "cn=admin,dc=ibantfg,dc=com" -W -f modificacionMaite.ldif"**.



Imagen 43: Nuevo usuario

Evidentemente, el recién incorporado usuario ha sido creado sin inconvenientes, y el script ha logrado su integración efectiva en los servidores. Adicionalmente, en la imagen subsiguiente se percibe cómo el usuario "Maite" ahora únicamente responde a la clave correspondiente a "Javier".

A black screen with white text

Description automatically generated

Imagen 44: Modificación Maite

### Prueba archivo de registro

Para esta prueba, simplemente procedemos a acceder a los diversos servidores mediante los distintos clientes y observamos cómo se van agregando las entradas al archivo de registro, tal como hemos estado llevando a cabo en las pruebas anteriores. El archivo de registro presenta ahora la siguiente configuración:

A screen shot of a computer

Description automatically generated

Imagen 45: Archivo logs

### Prueba alertas

Una vez más, para poner a prueba las alertas, nuestro enfoque consiste en generar una alerta específica y luego observar cómo se añade al archivo de registro. Este sería el resultado:

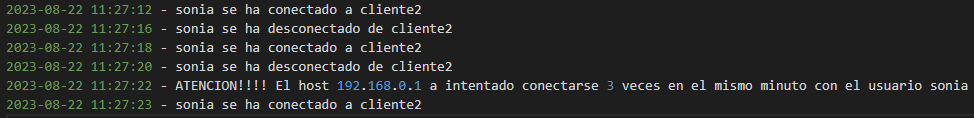


Imagen 46: Archivo logs con alerta

# Conclusiones

En resumen, este proyecto ha dado lugar a una solución integral y altamente efectiva para la administración y seguridad de los servidores Linux en una red. A través de la unión estratégica del módulo PAM y el servidor LDAP, junto con las funcionalidades de registro y alerta implementadas, se ha construido una infraestructura robusta y versátil que satisface múltiples necesidades de gestión y protección.

La centralización de conexiones a través de la combinación de PAM y LDAP ha permitido una administración más eficiente de las claves públicas SSH, mejorando la accesibilidad y la seguridad en todo el entorno. La automatización del proceso de recepción y almacenamiento de claves en los clientes agiliza considerablemente la gestión de identidades, asegurando la consistencia y la disponibilidad de las claves en toda la red.

La introducción de un sistema de registro de accesos y alertas representa un hito crucial en la seguridad de la infraestructura. Al habilitar la generación de registros detallados de cada acceso, se ha proporcionado a los administradores una visibilidad completa de la actividad en los servidores. Las alertas en tiempo real por actividades sospechosas añaden una capa adicional de protección, permitiendo una respuesta proactiva ante posibles amenazas y comportamientos anómalos.

En conjunto, esta implementación no solo optimiza la gestión de conexiones y claves en servidores Linux, sino que también eleva la seguridad a un nivel superior. La combinación de centralización, automatización, monitoreo y alertas forma un sistema sólido que no solo simplifica las operaciones diarias, sino que también dota a los administradores de herramientas efectivas para prevenir y mitigar riesgos de seguridad. Este proyecto se alza como una contribución significativa al campo de la administración de sistemas Linux, reforzando la capacidad de las organizaciones para salvaguardar la integridad y la confidencialidad de sus redes.

Se puede observar como no se ha conseguido exactamente lo que ser esperaba por diversos problemas, pero el resultado satisface los objetivos y todo funciona correctamente.

# Referencias

Información sobre tecnologías:

<https://jumpcloud.com/es/blog/what-is-ldap-authentication>

<https://www.ibm.com/es-es/topics/docker>

<https://www.dongee.com/tutoriales/que-es-docker-compose/>

<https://www.hostinger.es/tutoriales/que-es-ssh>

GitHub del proyecto:

<https://github.com/ibantxu12/tfg_iban>