01/07/23

Claves públicas SSH en LDAP y registro de accesos Linux

Iban Ruiz de Galarreta Cadenas

UPNA - Pamplona

Contents

1. INTRODUCCIÓN ____ 3

2. OBJETIVOS ____ 4

1.1. Integración de Claves Públicas SSH en LDAP ____ 4

1.2. Registro de Accesos y Generación de Alertas ____ 4

3. IMPLEMENTACIÓN ____ 5

1. Despliegue de la infraestructura ____ 5

1.1. Servidor OpenLDAP ____ 5

1.2. Clientes Ubuntu ____ 8

1.3. Prueba de ejecución ____ 10

2. Claves públicas en LDAP **____ 11**

2.1. Creación de las claves públicas ____ 11

2.2. Configurar las calves públicas en el servidor LDAP ____ 12

2.3. Script para recibir las claves ____ 14

2.4. Ejecución del script ____ 16

3. Registro de accesos y alertas ____ 18

3.1. Servicio de registro centralizado ____ 18

3.2. Alertas de anomalías ____ 21

4. PRUEBAS ____ 24

4.1. Pruebas configuración inicial ____ 24

4.2. Pruebas claves SSH ____ 25

4.3. Prueba archivo de registro ____ 28

4.4. Prueba alertas ____ 29

5. CONCLUSIONES ____ 30

1. 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.

2. Objetivos

Los dos principales objetivos de mi proyecto son los siguientes:

1.1. 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.

1.2. 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.

3. Implementación

1. 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.

1.1. Servidor OpenLDAP

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

```yaml
openldap:
  image: osixia/openldap:latest
  container_name: openldap
  hostname: openldap
  environment:
    - LDAP_ORGANISATION=TFG iban
    - LDAP_DOMAIN=ibantfg.com
    - LDAP_ADMIN_USERNAME=admin
    - LDAP_ADMIN_PASSWORD=LDAPapTFG
    - LDAP_CONFIG_PASSWORD=LDAPcpTFG
    - "LDAP_BASE_DN=dc=ibantfg,dc=com"
    - LDAP_TLS_CRT_FILENAME=server.crt
    - LDAP_TLS_KEY_FILENAME=server.key
    - LDAP_TLS_CA_CRT_FILENAME=ibantfg.com.ca.crt
  volumes:
    - ./ldapConf/compartido:/root/compartido

  networks:
    - openldap_n
```

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.

1.2. 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:
    a. **"sudo apt update"**
    b. **"sudo apt install ssh"**
    c. **"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:
    a. Se introdujo la dirección de red del servidor LDAP en el campo correspondiente, como **"ldapi:///openldap/"**.
    b. Se configuró el dominio LDAP según lo establecido en el servidor, en este caso **"dc=ibantfg,dc=com"**.
    c. Se seleccionó la versión 3 de LDAP.
    d. 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:
    a. 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.
    b. 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:

```
client1:
  image: ibantxu12/uldapyssh
  container_name: client1
  hostname: cliente1
  ports:
    - 221:22
  networks:
    - openldap_n
  command: /usr/sbin/sshd -D
```

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.

1.3. 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:

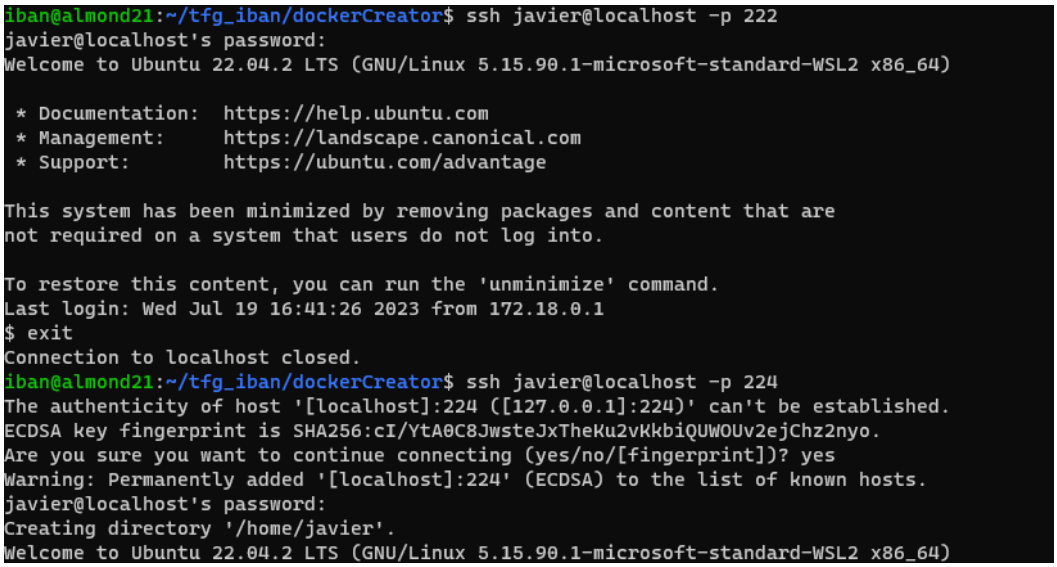

Imagen 3: Ejemplo ejecución

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.

2. Claves públicas en LDAP

2.1. 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.

```
iban@almond21:~/tfg_iban/dockerCreator/ldapConf/compartido$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/iban/.ssh/id_rsa): ./javier
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ./javier
Your public key has been saved in ./javier.pub
The key fingerprint is:
SHA256:wLvKpWkAHhnTRbLwQrWY47olnuL9Y7tqCv9GKIYoXw0 iban@almond21
The key's randomart image is:
+---[RSA 3072]----+
| ooooo           |
|.o=.+.           |
| =++  o          |
|ooo E  o         |
|+o.. o. S        |
|*oo o ..         |
|*o.+  o          |
|+=+ooB           |
|+=+*X++          |
+----[SHA256]-----+
```

Imagen 4: Creación de claves

2.2. 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:

```
root@openldap:/# ldapadd -x -D cn=admin,dc=ibantfg,dc=com -w LDAPapTFG -f /root/compartido/u.ldif
adding new entry "uid=javier,ou=users,dc=ibantfg,dc=com"
ldap_add: Object class violation (65)
        additional info: attribute 'sshPublicKey' not allowed
```

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"**.

```
dn: cn={2}nis,cn=schema,cn=config
changetype: modify
add: olcAttributeTypes
olcAttributeTypes: ( 1.3.6.1.4.1.9999.1.1.1 NAME 'customAttribute'
    DESC 'Custom Attribute'
    EQUALITY caseIgnoreMatch
    SUBSTR caseIgnoreSubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

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.

```
iban@almond21:~/tfg_iban/dockerCreator$ docker exec -it openldap bash
root@openldap:/# ldapadd -x -D cn=admin,dc=ibantfg,dc=com -w LDAPapTFG -f /root/compartido/pu.ldif
adding new entry "ou=users,dc=ibantfg,dc=com"

adding new entry "uid=nuevoUsuario,ou=users,dc=ibantfg,dc=com"
ldap_add: Undefined attribute type (17)
        additional info: customAttribute: attribute type undefined

root@openldap:/#
```

Imagen 7: Antes de crearlo

```
root@openldap:/# ldapadd -x -D cn=admin,dc=ibantfg,dc=com -w LDAPapTFG -f /root/compartido/pu.ldif
adding new entry "uid=nuevoUsuario,ou=users,dc=ibantfg,dc=com"
ldap_add: Object class violation (65)
        additional info: attribute 'customAttribute' not allowed
```

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".

```
# Usuario Javier
dn: uid=javier,ou=users,dc=ibantfg,dc=com
objectClass: top
objectClass: posixAccount
objectClass: inetOrgPerson
cn: Javier
sn: Javier
uid: javier
uidNumber: 1001
gidNumber: 1001
homeDirectory: /home/javier
userPassword: javierPass
mail: javier@ibantfg.com
description: ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDwR99sbF3O
```

Imagen 9: Clave añadida

2.3. 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:

```
LDAP_SERVER="ldap://openldap"
BASE_DN="dc=ibantfg,dc=com"
LDAP_USER="admin"
LDAP_PASSWORD="LDAPapTFG"

LDAP_QUERY="(&(objectClass=posixAccount))"
```

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.

```
ldapsearch -x -H "$LDAP_SERVER" -b "$BASE_DN" -D "cn=$LDAP_USER,$BASE_DN" -w "$LDAP_PASSWORD" "$LDAP_QUERY" | while IFS= read -r line; do
    if [[ $line =~ ^dn:\ (.*) ]]; then
        user_dn="${BASH_REMATCH[1]}"
        username=$(echo "$user_dn" | cut -d ',' -f 1 | cut -d '=' -f 2)
        groups=$(ldapsearch -x -H "$LDAP_SERVER" -b "$BASE_DN" -D "cn=$LDAP_USER,$BASE_DN" -w "$LDAP_PASSWORD" "(memberUid=$username)" "cn
        user_home="/home/$username"

        if [ ! -d "$user_home" ]; then
            mkdir -p "$user_home"
            chown "$username:$groups" "$user_home"
            chmod 700 "$user_home"
        fi
```

Imagen 11: Bucle y carpetas

El 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.

```
pubkey=$(ldapsearch -x -H "$LDAP_SERVER" -b "$user_dn" -D "cn=$LDAP_USER,$BASE_DN" -w "$LDAP_PASSWORD" | awk
pubkey="ssh-rsa "$(echo "$pubkey" | sed 's/description: ssh-rsa //'| tr -d '[:space:]')

if [ -n "$pubkey" ]; then
    mkdir -p "$user_home/.ssh"
    chown "$username:$groups" "$user_home/.ssh"
    echo "$pubkey" > "$user_home/.ssh/authorized_keys"
    chown "$username:$groups" "$user_home/.ssh/authorized_keys"
    chmod 600 "$user_home/.ssh/authorized_keys"
    chmod 700 "$user_home/.ssh"
fi
```

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"**.

2.4. 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.

```
client1:
  image: ibantxu12/uldapyssh
  container_name: client1
  hostname: cliente1
  ports:
    - 221:22
  volumes:
    - ./ldapConf/scriptsClientes:/usr/local/bin/scripts
    - ./ldapConf/pam:/etc/pam.d/sshd
```

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.

```
# PAM configuration for the Secure Shell service
auth sufficient pam_exec.so /usr/local/bin/scripts/getSSHKey.sh
```

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.

```
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of cr
#
# m h  dom mon dow   command

* * * * * /usr/local/bin/scripts/getSSHKey.sh
```

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.

```
#!/bin/bash
/usr/sbin/sshd -D &
/usr/sbin/cron -f
```

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.

3. Registro de accesos y alertas

3.1. 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"**.

```
dn: cn=config
changetype: modify
replace: olcLogLevel
olcLogLevel: trace
```

Imagen 17: Audit_config.ldif

Luego se ejecutó el siguiente comando en el servidor LDAP.

```
root@openldap:~/compartido# ldapmodify -x -D "cn=admin,cn=config" -w LDAPcpTFG -f audit_config.ldif
modifying entry "cn=config"
```

Imagen 18: Comando auditoria

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

```
USERNAME="$USER"
CLIENT_IP="$(who | awk '{print $6}')"
LDAP_SERVER="openldap"

# Archivo LDIF para el evento de auditoría
LDIF_FILE="/tmp/ldap_audit.ldif"

# Crear el archivo LDIF
echo "dn: cn=admin,dc=ibantfg,dc=com" > "$LDIF_FILE"
echo "changetype: add" >> "$LDIF_FILE"
echo "objectClass: auditEvent" >> "$LDIF_FILE"
echo "eventType: sshAuthentication" >> "$LDIF_FILE"
echo "eventName: SSH Authentication" >> "$LDIF_FILE"
echo "eventDescription: User $USERNAME authenticated from $CLIENT_IP" >> "$LDIF_FILE"

# Enviar el evento de auditoría al servidor LDAP
ldapmodify -x -D " cn=admin,dc=ibantfg,dc=com" -w "LDAPapTFG" -H "ldap://$LDAP_SERVER" -f "$LDIF_FILE"

# Eliminar el archivo LDIF
rm "$LDIF_FILE"
```

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.

```
root@cliente1:/# /usr/local/bin/scripts/sendLog.sh
adding new entry "cn=admin,dc=ibantfg,dc=com"
ldap_add: Invalid syntax (21)
        additional info: objectClass: value #0 invalid per syntax
```

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.

```yaml
ssh_log:
  image: alpine:latest
  container_name: ssh_log
  command: sh -c "while true; do nc -l -p 514 >> /var/log/ssh.log; done"
  volumes:
    - ./ldapConf/ssh.log:/var/log/ssh.log
  networks:
    - openldap_n
```

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.

```
LOG_SERVER="ssh_log"
LOG_PORT=514


CURRENT_TIME=$(date +"%Y-%m-%d %H:%M:%S")
USERNAME="$PAM_USER"
SERVER_NAME=$(hostname)

if [ "$PAM_TYPE" == "open_session" ]; then
    ACTION="conectado a"
else
    ACTION="desconectado de"
fi

echo "$CURRENT_TIME - $USERNAME se ha $ACTION $SERVER_NAME" | nc -w1 -q1 "$LOG_SERVER" "$LOG_PORT"
```

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.

```
@include common-password

session optional pam_exec.so /usr/local/bin/scripts/sendLog.sh
```

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:

```
1   2023-08-21 16:31:16 - maite se ha conectado a cliente2
2   2023-08-21 16:31:19 - maite se ha desconectado a cliente2
3   2023-08-21 16:31:58 - maite se ha conectado a cliente4
4   2023-08-21 16:32:58 - javier se ha conectado a cliente3
5   2023-08-21 16:33:07 - maite se ha desconectado de cliente4
6   2023-08-21 16:33:11 - javier se ha desconectado de cliente3
```

Imagen 22: Ejemplo de logs

3.2. 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.

```
LOG_FILE="/var/log/login_attempts.log"
LOG_SERVER="ssh_log"
LOG_PORT=514


if [[ "$PAM_TYPE" == "open_session" && "$PAM_USER" != "root" && "$PAM_SERVICE" == "sshd" && "$PAM_RHOST" != "" ]]; then
    # Incrementar contador de intentos de inicio de sesión fallidos para el usuario
    echo "$(date '+%Y-%m-%d %H:%M:%S') - User: $PAM_USER, Host: $PAM_RHOST" >> "$LOG_FILE"

    current_time=$(date '+%Y-%m-%d %H:%M')
    current_ip="$PAM_RHOST"
    count=0
    while read -r line; do
        timestamp=$(echo "$line" | awk '{print $1, $2}')
        timestamp=$(echo "$timestamp" | cut -d' ' -f1-2 | rev | cut -c 4- | rev)
        ip=$(echo "$line" | awk '{print $NF}')

        if [[ "$timestamp" == "$current_time" && "$ip" == "$current_ip" ]]; then
            ((count++))
        fi
    done < "$LOG_FILE"

    if [[ "$count" -ge 3 ]]; then
        echo "dentro"
        echo "$(date '+%Y-%m-%d %H:%M:%S') - ATENCION!!!! El host $PAM_RHOST a intentado conectarse 3 veces en el mismo
    fi
fi
```

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"**.

```
session required pam_exec.so /usr/local/bin/scripts/loginAttempts.sh

# Standard Un*x session setup and teardown.
@include common-session
```

Imagen 24: Configuración PAM loginAttempts

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

```
2023-08-21 21:48:32 - javier se ha desconectado de cliente1
2023-08-21 21:48:35 - javier se ha conectado a cliente1
2023-08-21 21:48:38 - javier se ha desconectado de cliente1
2023-08-21 21:48:39 - ATENCION!!!! El host 172.29.0.1 a intentado conectarse 3 veces en el mismo minuto con el usuario javier
```

Imagen 25: Registro de alertas

4. Pruebas

4.1. 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:

```
iban@almond21:~/tfg_iban$ ssh mario@localhost -p 221
mario@localhost's password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.90.1-microsoft-standard-WSL2 x86_64)
```

Imagen 26: Mario 221

```
iban@almond21:~/tfg_iban$ ssh javier@localhost -p 221
javier@localhost's password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.90.1-microsoft-standard-WSL2 x86_64)
```

Imagen 27: Javier 221

```
iban@almond21:~/tfg_iban$ ssh mikel@localhost -p 222
mikel@localhost's password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.90.1-microsoft-standard-WSL2 x86_64)
```

Imagen 28: Mikel 222

```
iban@almond21:~/tfg_iban$ ssh mario@localhost -p 222
mario@localhost's password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.90.1-microsoft-standard-WSL2 x86_64)
```

Imagen 29: Mario 222

```
iban@almond21:~/tfg_iban$ ssh javier@localhost -p 224
javier@localhost's password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.90.1-microsoft-standard-WSL2 x86_64)
```

Imagen 30: Javier 224

4.2. 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:

```
iban@almond21:~/tfg_iban/dockerCreator$ ssh -i keys/lorea lorea@localhost -p 221
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.90.1-microsoft-standard-WSL2 x86_64)
```

Imagen 31: Key Lorea 221

```
$ hostname
cliente1
$ ls /home
javier  jlopez  lorea  maite  mario  mark  mikel  olatz
```

Imagen 32: Key cliente1

```
iban@almond21:~/tfg_iban/dockerCreator$ ssh -i keys/mark mark@localhost -p 222
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.90.1-microsoft-standard-WSL2 x86_64)
```

Imagen 33: Key Mark 222

```
$ hostname
cliente2
$ ls /home
javier  jlopez  lorea  maite  mario  mark  mikel  olatz
```

Imagen 34: Key cliente2

```
iban@almond21:~/tfg_iban/dockerCreator$ ssh -i keys/olatz olatz@localhost -p 223
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.90.1-microsoft-standard-WSL2 x86_64)
```

Imagen 35: Key Olatz 223

```
$ hostname
cliente3
$ ls /home
javier  jlopez  lorea  maite  mario  mark  mikel  olatz
```

Imagen 36: Key cliente3

```
iban@almond21:~/tfg_iban/dockerCreator$ ssh -i keys/mikel mikel@localhost -p 224
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.90.1-microsoft-standard-WSL2 x86_64)
```

Imagen 37: Key Mikel 224

```
$ hostname
cliente4
$ ls /home
javier  jlopez  lorea  maite  mario  mark  mikel  olatz
```

Imagen 38: Key cliente4

```
iban@almond21:~/tfg_iban/dockerCreator$ ssh -i keys/maite maite@localhost -p 225
The authenticity of host '[localhost]:225 ([127.0.0.1]:225)' can't be established.
ECDSA key fingerprint is SHA256:cI/YtA0C8JwsteJxTheKu2vKkbiQUWOUv2ejChz2nyo.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[localhost]:225' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.90.1-microsoft-standard-WSL2 x86_64)
```

Imagen 39: Key Maite 225

```
$ hostname
cliente5
$ ls /home
javier  jlopez  lorea  maite  mario  mark  mikel  olatz
```

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.

```
dn: uid=sonia,ou=users,dc=ibantfg,dc=com
objectClass: top
objectClass: posixAccount
objectClass: inetOrgPerson
cn: Sonia
sn: Sonia
uid: sonia
uidNumber: 1008
gidNumber: 1003
homeDirectory: /home/sonia
userPassword: soniaPass
mail: sonia@ibantfg.com
description: ssh-rsa AAAAB3NzaC1yc2EAAAADAQABA
```

Imagen 41: LDIF añadir Sonia

```
dn: uid=maite,ou=users,dc=ibantfg,dc=com
changetype: modify
replace: description
description: ssh-rsa AAAAB3NzaC1yc2EAAAADAQAB
```

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"**.

```
iban@almond21:~/tfg_iban/dockerCreator$ ssh -i keys/lorea sonia@localhost -p 222
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.90.1-microsoft-standard-WSL2 x86_64)
```

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".

```
iban@almond21:~/tfg_iban/dockerCreator$ ssh -i keys/maite maite@localhost -p 223
maite@localhost's password:

iban@almond21:~/tfg_iban/dockerCreator$ ssh -i keys/javier maite@localhost -p 223
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.90.1-microsoft-standard-WSL2 x86_64)
```

Imagen 44: Modificación Maite

4.3. 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:

```
2023-08-22 09:52:11 - mario se ha conectado a cliente1
2023-08-22 09:52:48 - mario se ha desconectado de cliente1
2023-08-22 09:52:59 - javier se ha conectado a cliente1
2023-08-22 09:53:19 - javier se ha desconectado de cliente1
2023-08-22 09:53:40 - mikel se ha conectado a cliente2
2023-08-22 09:53:57 - mikel se ha desconectado de cliente2
2023-08-22 09:54:09 - mario se ha conectado a cliente2
2023-08-22 09:54:55 - mario se ha desconectado de cliente2
2023-08-22 09:55:08 - javier se ha conectado a cliente4
2023-08-22 10:03:42 - javier se ha desconectado de cliente4
2023-08-22 10:04:58 - lorea se ha conectado a cliente1
2023-08-22 10:08:00 - lorea se ha desconectado de cliente1
2023-08-22 10:08:17 - mark se ha conectado a cliente2
2023-08-22 10:10:04 - mark se ha desconectado de cliente2
2023-08-22 10:10:35 - olatz se ha conectado a cliente3
2023-08-22 10:11:34 - olatz se ha desconectado de cliente3
2023-08-22 10:11:57 - mikel se ha conectado a cliente4
2023-08-22 10:13:40 - mikel se ha desconectado de cliente4
2023-08-22 10:13:58 - maite se ha conectado a cliente5
2023-08-22 10:42:37 - maite se ha desconectado de cliente5
```

Imagen 45: Archivo logs

4.4. 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:

```
2023-08-22 11:27:12 - sonia se ha conectado a cliente2
2023-08-22 11:27:16 - sonia se ha desconectado de cliente2
2023-08-22 11:27:18 - sonia se ha conectado a cliente2
2023-08-22 11:27:20 - sonia se ha desconectado de cliente2
2023-08-22 11:27:22 - ATENCION!!!! El host 192.168.0.1 a intentado conectarse 3 veces en el mismo minuto con el usuario sonia
2023-08-22 11:27:23 - sonia se ha conectado a cliente2
```

Imagen 46: Archivo logs con alerta

5. 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.