# Autenticación LDAP

Resumen

Este documento pretende ser una guía para la configuración de un servidor LDAP (principalmente un servidor OpenLDAP) para la autenticación en FreeBSD. Esto es útil para situaciones en las que muchos servidores necesitan las mismas cuentas de usuario, por ejemplo, como reemplazo de NIS.

## Tabla de contenidos

1. Prólogo . . . 1
2. Configurando LDAP . . . 1
3. Configuración del Cliente . . . 6
4. Consideraciones de Seguridad . . . 11
Apéndice A: Consideraciones Útiles . . . 13
Apéndice B: Certificados OpenSSL para LDAP . . . 14

## 1. Prólogo

Este documento está destinado a proporcionar al lector una comprensión suficiente de LDAP para configurar un servidor LDAP. Este documento intentará proporcionar una explicación de net/nss_ldap y security/pam_ldap para usarlos con los servicios de la máquina del cliente para su uso con el servidor LDAP.

Cuando termine, el lector debería poder configurar e implementar un servidor FreeBSD que pueda alojar un directorio LDAP, y configurar e implementar un servidor FreeBSD que pueda autenticarse en un directorio LDAP.

Este artículo no pretende ser una explicación exhaustiva de las consideraciones de seguridad, robustez o mejores prácticas para configurar LDAP u otros de los servicios que se explican aquí. Aunque el autor tiene cuidado de hacer todo correctamente, no aborda los problemas de seguridad más allá del alcance general. Este artículo debe tenerse en cuenta para sentar las bases teóricas únicamente, y cualquier implementación real debe ir acompañado de un análisis cuidadoso de los requisitos.

## 2. Configurando LDAP

LDAP significa "Lightweight Directory Access Protocol" (Protocolo Ligero de Acceso a Directorio) y es un subconjunto del Protocolo de Acceso a Directorio X.500. Su especificación más reciente se encuentra en RFC4510. En esencia es una base de datos que espera recibir muchas más consultas que escrituras.

En los ejemplos de este documento se utilizará el servidor LDAP OpenLDAP; aunque los procedimientos deberían ser aplicables a los diferentes servidores, la mayor parte de la administración es específica de OpenLDAP. Hay varias versiones del servidor en la colección de ports, por ejemplo, net/openldap24-server. Los clientes necesitarán las librerías necesarias del paquete net/openldap24-client.

Hay (básicamente) dos áreas del servicio LDAP que necesitan configuración. Lo primero es configurar un servidor para recibir conexiones correctamente, y lo segundo es añadir entradas al directorio del servidor para que las herramientas de FreeBSD sepan como interactuar con él.

# 2.1. Configurar el Servidor para recibir Conexiones



Esta sección es específica de OpenLDAP. Si usas otro servidor, necesitarás consultar su propia documentación.

## 2.1.1. Instalando OpenLDAP

Primero, instala OpenLDAP:

*Ejemplo 1. Instalando OpenLDAP*

```
# cd /usr/ports/net/openldap24-server
# make install clean
```

Esto instala los binarios `slapd` y `slurpd`, junto con las librerías OpenLDAP necesarias.

## 2.1.2. Configurando OpenLDAP

Después necesitamos configurar OpenLDAP.

Es necesario que hagas obligatorio el uso de cifrado en tus conexiones al servidor LDAP; de lo contrario, las contraseñas de sus usuarios se transferirán en texto plano, lo que se considera inseguro. Las herramientas que utilizaremos admiten dos tipos muy similares de encriptación, SSL y TLS.

TLS significa "Seguridad en Capa de Transporte" (Transportation Layer Security). Los servicios que utilizan TLS suelen conectarse *a los mismos* puertos que los servicios que no utilizan TLS; por lo tanto un servidor SMTP que soporta TLS escuchará conexiones en el puerto 25 y un servidor LDAP escuchará conexiones en 389.

SSL significa "Capa de Sockets Seguros" (Secure Sockets Layer) y los servicios que implementan SSL *no* escuchan en los mismos puertos que sus equivalentes sin SSL. Por lo tanto SMTPS escucha en el puerto 465 (no en el 25), HTTPS escucha en el 443 y LDAPS en el 636.

La razón por la que SSL utiliza un puerto diferente a TLS es porque una conexión TLS empieza como texto plano y cambia al tráfico cifrado después de la directiva `STARTTLS`. Las conexiones SSL se cifran desde el principio. Aparte de eso, no hay diferencias sustanciales entre ambos.



Ajustaremos OpenLDAP para que utilice TLS ya que SSL se considera obsoleto.

Una vez que hemos instalado OpenLDAP, los siguientes parámetros en /usr/local/etc/openldap/slapd.conf habilitarán el uso de TLS:

```
security ssf=128

TLSCertificateFile /path/to/your/cert.crt
TLSCertificateKeyFile /path/to/your/cert.key
TLSCACertificateFile /path/to/your/cacert.crt
```

En este caso, `ssf=128` indica a OpenLDAP que solicite una encriptación de 128 bits para todas las conexiones, tanto para búsquedas como para actualizaciones. Este parámetro se podría configurar según las necesidades de seguridad de tu sitio web, pero es raro que necesites rebajarlo ya que la mayoría de las librerías cliente de LDAP soportan encriptación fuerte.

Los ficheros cert.crt, cert.key, y cacert.crt son necesarios para que los clientes te autentiquen *a ti* como el servidor LDAP válido. Si sólo quieres ejecutar un servidor, puedes crear un certificado auto firmado con OpenSSL:

*Ejemplo 2. Generar una Clave RSA*

```
% openssl genrsa -out cert.key 1024
Generating RSA private key, 1024 bit long modulus
.....................++++++
...++++++
e is 65537 (0x10001)

% openssl req -new -key cert.key -out cert.csr
```

En este punto se te deberían preguntar algunos valores. Podrías introducir los valores que quisieras; sin embargo, es importante que el valor de "Common Name" sea el nombre de dominio del servidor LDAP completamente cualificado. En nuestro caso, y en los ejemplos, el servidor es *server.example.org*. Establecer este valor incorrectamente hará que los clientes no puedan conectar. Esto puede causar una gran frustración así que asegúrate de que sigues estos pasos con cuidado.

Finalmente, el certificado debe firmarse:

*Ejemplo 3. Autofirmar el certificado*

```
% openssl x509 -req -in cert.csr -days 365 -signkey cert.key -out cert.crt
Signature ok
subject=/C=AU/ST=Some-State/O=Internet Widgits Pty Ltd
Getting Private key
```

Esto creará un certificado auto firmado que puede ser usado para las directivas en slapd.conf, donde cert.crt y cacert.crt son el mismo fichero. Si vas a utilizar muchos servidores OpenLDA (para replicación vía `slurpd`) querrás echar un vistazo a Certificados OpenSSL para LDAP para generar una clave CA y usarla para firmar los certificados de servidor individuales.

Una vez hecho esto, escribe lo siguiente en /etc/rc.conf:

```
slapd_enable="YES"
```

Después ejecuta `/usr/local/etc/rc.d/slapd start`. Esto debería arrancar OpenLDAP. Confirma que está escuchando en el puerto 389 con

```
% sockstat -4 -p 389
ldap     slapd      3261  7  tcp4   *:389                 *:*
```

### 2.1.3. Configurar el Cliente

Instala el port net/openldap24-client para obtener las librerías de OpenLDAP. Las máquinas cliente siempre tendrán las librerías de OpenLDAP pues que eso es lo único que soportan security/pam_ldap y net/nss_ldap, al menos por el momento.

El fichero de configuración para las librerías de OpenLDAP es /usr/local/etc/openldap/ldap.conf. Edita este fichero para que contenga los siguientes valores:

```
base dc=example,dc=org
uri ldap://server.example.org/
ssl start_tls
tls_cacert /path/to/your/cacert.crt
```



Es importante que tus clientes tengan acceso a cacert.crt, de lo contrario no podrán conectarse.



Hay dos ficheros que se llaman ldap.conf. El primero es este fichero, que es para las librerías OpenLDAP y define cómo hablar con el servidor. El segundo es /usr/local/etc/ldap.conf y es para pam_ldap.

En este punto deberías ser capaz de ejecutar `ldapsearch -Z` en la maquina cliente; `-Z` significa "usa TLS". Si encuentras un error, entonces algo está mal configurado; seguramente sean tus certificados. Utiliza los comandos `s_client` y `s_server` de openssl(1) para asegurarte de que están correctamente configurados y firmados.

## 2.2. Entradas en la base de datos

La autenticación en un directorio LDAP se logra generalmente al intentar vincularse al directorio como el usuario que se conecta. Esto se realiza mediante el establecimiento de un enlace "simple"

en el directorio con el nombre de usuario proporcionado. Si hay una entrada con el `uid` igual al nombre de usuario y el atributo `userPassword` de la entrada coincide con la contraseña proporcionada, el enlace tiene éxito.

Lo primero que tenemos que hacer es averiguar en qué parte del directorio estarán nuestros usuarios.

La entrada base de nuestra base de datos es `dc=example,dc=org`. La mayoría de los clientes esperan una localización para los usuarios que sea algo como `ou=people,base` así que es lo que se usará aquí. Sin embargo, ten en cuenta que esto es configurable.

Así que la entrada ldif para la unidad organizacional `people` se parecerá a:

```
dn: ou=people,dc=example,dc=org
objectClass: top
objectClass: organizationalUnit
ou: people
```

Todos los usuarios se crearán como subentradas de esta unidad organizativa.

Se podría pensar en la clase de objeto a la que pertenecerán sus usuarios. Por defecto, la mayoría de las herramientas utilizarán `people`, lo cual está bien si simplemente quieres proporcionar entradas para la autenticación. Sin embargo, si también vas a almacenar información de usuario en la base de datos LDAP, probablemente quieras usar `inetOrgPerson`, el cual dispone de muchos atributos útiles. En cualquier caso, los esquemas relevantes deben introducirse en el archivo slapd.conf.

Para este ejemplo utilizaremos la clase de objeto `person`. Si usas `inetOrgPerson`, los pasos son básicamente iguales, con la excepción de que se requiere el atributo `sn`.

Para añadir un usuario de pruebas llamado `tuser`, el ldif sería:

```
dn: uid=tuser,ou=people,dc=example,dc=org
objectClass: person
objectClass: posixAccount
objectClass: shadowAccount
objectClass: top
uidNumber: 10000
gidNumber: 10000
homeDirectory: /home/tuser
loginShell: /bin/csh
uid: tuser
cn: tuser
```

Yo empiezo los UIDs de mis usuarios de LDAP en el 10000 para evitar conflictos con las cuentas del sistema; puedes establecer el número que desees aquí, siempre que sea inferior a 65536.

También necesitamos entradas grupales. Son tan configurables como las entradas de usuario, pero

usaremos los valores predeterminados que se muestran a continuación:

```
dn: ou=people,dc=example,dc=org
objectClass: top
objectClass: organizationalUnit
ou: people

dn: cn=tuser,ou=groups,dc=example,dc=org
objectClass: posixGroup
objectClass: top
gidNumber: 10000
cn: tuser
```

Para introducir estos en tu base de datos, puedes utilizar `slapadd` o `ldapadd` en un fichero que contenga esas entradas. De forma alternativa, puedes utilizar sysutils/ldapvi.

La utilidad `ldapsearch` en la máquina del cliente debería devolver estas entradas. Si es así, la base de datos está configurada correctamente para ser utilizada como un servidor de autenticación LDAP.

# 3. Configuración del Cliente

El cliente ya debería tener las librerías de OpenLDAP de Configurar el Cliente, pero si estás instalando varias máquinas cliente, necesitarás instalar net/openldap24-client en cada una de ellas.

FreeBSD requiere de la instalación de dos ports para autenticarse en un servidor LDAP, security/pam_ldap y net/nss_ldap.

## 3.1. Autenticación

security/pam_ldap se configura en el fichero /usr/local/etc/ldap.conf.



Este fichero es *diferente* del fichero de configuración de las librerías de OpenLDAP, /usr/local/etc/openldap/ldap.conf; sin embargo, tiene muchas de las mismas opciones; de hecho es un superconjunto de ese fichero. En lo que queda de sección, referencias a ldap.conf se refieren a /usr/local/etc/ldap.conf.

Por lo tanto, queremos copiar todos nuestros parámetros de configuración originales de openldap/ldap.conf al nuevo ldap.conf. Una vez hecho esto, le indicaremos a security/pam_ldap qué buscar en el servidor de directorio.

Estamos identificando nuestros usuarios mediante el atributo `uid`. Para configurarlo (aunque es el valor por defecto), establece la directiva `pam_login_attribute` en ldap.conf:

*Ejemplo 4. Estableciendo* `pam_login_attribute`

```
pam_login_attribute uid
```

Con esto ya establecido, security/pam_ldap buscará el valor `uid=username` en todo el directorio LDAP bajo `base`. Si encuentra una sola entrada, intentará vincular a ese usuario con la contraseña que se le ha pasado. Se vincula correctamente, entonces permitirá el acceso. En cualquier otro caso fallará.

Los usuarios cuyo shell no esté en /etc/shells no podrán iniciar sesión. Esto es muy importante cuando se configura Bash como la shell de usuario en el servidor LDAP. Bash no está incluido en la instalación estándar de FreeBSD. Cuando se instala desde un paquete o port, se encuentra en el directorio /usr/local/bin/bash. Comprueba que la ruta a la shell en el servidor esté configurada correctamente:

```
% getent passwd username
```

Hay dos opciones cuando en la salida se muestra `/bin/bash` en la última columna. La primera es cambiar en el servidor LDAP la entrada del usuario para que apunte a /usr/local/bin/bash. La segunda es crear un enlace simbólico en la máquina LDAP cliente de forma que se pueda encontrar Bash en el lugar correcto:

```
# ln -s /usr/local/bin/bash /bin/bash
```

Asegúrate de que /etc/shells contiene las entradas tanto para `/usr/local/bin/bash` como para `/bin/bash`. El usuario ya será capaz de logearse en el sistema utilizando Bash como shell.

### 3.1.1. PAM

PAM, que significa "Pluggable Authentication Modules", es el método por el cual FreeBSD autentica la mayoría de sus sesiones. Para decirle a FreeBSD que queremos usar un servidor LDAP, tendremos que añadir una línea al archivo PAM apropiado.

La mayoría de las veces el fichero PAM apropiado es /etc/pam.d/sshd, si quieres usar SSH (recuerda establecer las opciones correspondientes en /etc/ssh/sshd_config, de lo contrario SSH no usará PAM).

Para usar PAM para la autenticación, añade la línea

```
auth  sufficient  /usr/local/lib/pam_ldap.so  no_warn
```

El lugar exacto en el que aparece esta línea en el fichero y las opciones que aparecen en la cuarta columna determinan el comportamiento exacto del mecanismo de autenticación; lee pam(d)

Con esta configuración deberías ser capaz de autenticar un usuario contra un directorio LDAP. PAM

realizará un vínculo con tus credenciales, y si tiene éxito le dirá a SSH que permita el acceso.

Sin embargo, no es buena idea permitir que *cada* usuario del directorio pueda acceder a *todos* las máquinas clientes. Con la configuración actual, todo lo que necesita un usuario para iniciar sesión en una máquina es una entrada LDAP. Afortunadamente, hay algunas formas de restringir el acceso de los usuarios.

ldap.conf admite la directiva `pam_groupdn`; cada cuenta que se conecta a esta máquina debe ser miembro del grupo especificado aquí. Por ejemplo, si tienes

```
pam_groupdn cn=servername,ou=accessgroups,dc=example,dc=org
```

en ldap.conf, solo los miembros de este grupo podrán iniciar sesión. Sin embargo hay algunas cosas a tener en cuenta.

Los miembros de este grupo se especifican en uno o más atributos `memberUid` y cada atributo debe tener el nombre completamente unívoco del miembro. Entonces `memberUid: someuser` no funcionará; debe ser:

```
memberUid: uid=someuser,ou=people,dc=example,dc=org
```

Además, esta directiva no se verifica en PAM durante la autenticación, se verifica durante la administración de la cuenta, por lo que necesitarás añadir más configuraciones en tus archivos de PAM en la sección de `account`. Esto, a su vez, requerirá que *cada* usuario se incluya en el grupo, lo cual no es necesariamente lo que queremos. Para evitar bloquear usuarios que no están en LDAP, debes habilitar el atributo `ignore_unknown_user`. Finalmente, debes configurar la opción `ignore_authinfo_unavail` para que el usuario no quede bloqueado en todos los ordenadores cuando el servidor LDAP no esté disponible.

Tu pam.d/sshd podría parecerse a esto:

*Ejemplo 5. Ejemplo de pam.d/sshd*

```
auth            required        pam_nologin.so          no_warn
auth            sufficient      pam_opie.so             no_warn no_fake_prompts
auth            requisite       pam_opieaccess.so       no_warn allow_local
auth            sufficient      /usr/local/lib/pam_ldap.so      no_warn
auth            required        pam_unix.so             no_warn try_first_pass

account         required        pam_login_access.so
account         required        /usr/local/lib/pam_ldap.so      no_warn
ignore_authinfo_unavail ignore_unknown_user
```



Como estamos añadiendo estas líneas específicamente a pam.d/sshd, esto solo tendrá efecto en las sesiones SSH. Los usuarios de LDAP no podrán iniciar sesión por consola. Para cambiar este comportamiento, examina los otros archivos en

> /etc/pam.d y modifícalos como corresponda.

## 3.2. Name Service Switch

NSS es el servicio que mapea atributos a nombres. Por ejemplo, si un fichero es propiedad del usuario `1001`, una aplicación preguntará a NSS por el nombre de `1001` y podría obtener `bob` o `ted` o el cualquiera que sea el nombre del usuario.

Ahora que tenemos nuestra información en LDAP, necesitamos decirle a NSS que mire ahí cuando se le hagan preguntas.

Est es lo que hace el port net/nss_ldap. Utiliza el mismo archivo de configuración que security/pam_ldap, y no debería necesitar ningún parámetro adicional después de su instalación. En cambio, solo quedaría editar el archivo /etc/nsswitch.conf para aprovechar el directorio. Simplemente cambia las siguientes líneas:

```
group: compat
passwd: compat
```

por

```
group: files ldap
passwd: files ldap
```

Esto te permitirá asignar nombres de usuario a UIDs y UIDs a nombres de usuario.

¡Felicidades! Ahora deberías tener la autenticación de LDAP en funcionamiento.

## 3.3. Advertencias

Desafortunadamente, en el momento de escribir esto FreeBSD no soportaba cambiar las contraseñas de usuario con passwd(1). Como resultado, la mayoría de los administradores tienen que implementar una solución por ellos mismos. Aquí proporciono algunos ejemplos. Observa que si escribes tu propio script de cambio de contraseñas deberías tener en cuenta algunas consideraciones de seguridad; lee Almacenamiento de Contraseña

*Ejemplo 6. Shell Script para Cambiar Contraseñas*

```
#!/bin/sh

stty -echo
read -p "Old Password: " oldp; echo
read -p "New Password: " np1; echo
read -p "Retype New Password: " np2; echo
stty echo
```

```
if [ "$np1" != "$np2" ]; then
  echo "Passwords do not match."
  exit 1
fi

ldappasswd -D uid="$USER",ou=people,dc=example,dc=org \
  -w "$oldp" \
  -a "$oldp" \
  -s "$np1"
```



Este script apenas verifica errores, pero lo más importante es el poco cuidado con el que almacena sus contraseñas. Si haces algo como esto, establece al menos el valor de `security.bsd.see_other_uids`:

```
# sysctl security.bsd.see_other_uids=0
```

Se puede utilizar un enfoque más flexible (y probablemente más seguro) escribiendo un programa personalizado o incluso una interfaz web. Lo siguiente es parte de una librería de Ruby que puede cambiar las contraseñas LDAP. Se puede usar por línea de comandos y en la web.

*Ejemplo 7. Script en Ruby para Cambiar las Contraseñas*

```
require 'ldap'
require 'base64'
require 'digest'
require 'password' # ruby-password

ldap_server = "ldap.example.org"
luser = "uid=#{ENV['USER']},ou=people,dc=example,dc=org"

# get the new password, check it, and create a salted hash from it
def get_password
  pwd1 = Password.get("New Password: ")
  pwd2 = Password.get("Retype New Password: ")

  raise if pwd1 != pwd2
  pwd1.check # check password strength

  salt = rand.to_s.gsub(/0\./, '')
  pass = pwd1.to_s
  hash =
"{SSHA}"+Base64.encode64(Digest::SHA1.digest("#{pass}#{salt}")+salt).chomp!
  return hash
end

oldp = Password.get("Old Password: ")
newp = get_password
```

```
# We'll just replace it.  That we can bind proves that we either know
# the old password or are an admin.

replace = LDAP::Mod.new(LDAP::LDAP_MOD_REPLACE | LDAP::LDAP_MOD_BVALUES,
                        "userPassword",
                        [newp])

conn = LDAP::SSLConn.new(ldap_server, 389, true)
conn.set_option(LDAP::LDAP_OPT_PROTOCOL_VERSION, 3)
conn.bind(luser, oldp)
conn.modify(luser, [replace])
```

Aunque no se garantiza que esté a salvo de agujeros de seguridad (la contraseña se guarda en memoria, por ejemplo), esto es más limpio y más flexible que un simple script `sh`.

# 4. Consideraciones de Seguridad

Ahora que tus máquinas (y posiblemente otros servicios) se están autenticando contra su servidor LDAP, este servidor tiene que estar protegido, así como /etc/master.passwd estaría en un servidor normal, y posiblemente aún más puesto que un servidor LDAP corrupto o comprometido rompería todos los servicios del cliente.

Recuerda, esta sección no es exhaustiva. Debes revisar continuamente tu configuración y procedimientos para mejorarlos.

## 4.1. Establecer Atributos de Solo Lectura

Varios atributos en LDAP deberían ser de sólo lectura. Si el usuario pudiera escribirlos, por ejemplo, un usuario podría cambiar su `uidNumber` a `0` ¡y obtener acceso `root`!

Para empezar, el atributo `userPassword` no debe ser legible por todo el mundo. Por defecto, cualquiera que pueda conectarse al servidor LDAP puede leer este atributo. Para deshabilitar esto, usa la siguiente configuración en el archivo slapd.conf:

*Ejemplo 8. Ocultar Contraseñas*

```
access to dn.subtree="ou=people,dc=example,dc=org"
  attrs=userPassword
  by self write
  by anonymous auth
  by * none

access to *
  by self write
  by * read
```

Esto evitará que se pueda leer el atributo `userPassword`, a la vez que seguirá permitiendo a los usuarios cambiar sus propias contraseñas.

Además, querrás evitar que los usuarios cambien algunos de sus atributos. De forma predeterminada, los usuarios pueden cambiar cualquier atributo (excepto aquellos en los que los esquemas LDAP mismos niegan cambios), como `uidNumber`. Para cerrar este agujero, modifica lo anterior a

*Ejemplo 9. Atributos de Solo Lectura*

```
access to dn.subtree="ou=people,dc=example,dc=org"
  attrs=userPassword
  by self write
  by anonymous auth
  by * none

access to attrs=homeDirectory,uidNumber,gidNumber
  by * read

access to *
  by self write
  by * read
```

Esto evitará que los usuarios puedan hacerse pasar por otros usuarios.

## 4.2. Definición de la Cuenta `root`

Habitualmente la cuenta `root` o la cuenta del gestor para el servicio de LDAP estará definida en el fichero de configuración. Por ejemplo, OpenLDAP soporta esto y funciona, pero puede dar lugar a problemas si slapd.conf se ve comprometido. Sería mejor usar esto sólo para entrar en LDAP y después definir ahí una cuenta `root`.

Es incluso mejor definir cuentas que tengan permisos limitados y omitir completamente la cuenta `root`. Por ejemplo, los usuarios que pueden crear o eliminar cuentas de usuario se añaden a un grupo, pero ellos mismos no pueden cambiar la pertenencia a este grupo. Esta política de seguridad ayudaría a mitigar los efectos de una contraseña que se haya podido filtrar.

### 4.2.1. Crear un Grupo de Mantenimiento

Supongamos que quieres que tu departamento de TI pueda cambiar los directorios home de los usuarios, pero no quieres que todos puedan añadir o eliminar usuarios. La forma de hacerlo es agregar un grupo para estos administradores:

*Ejemplo 10. Crear un Grupo de Mantenimiento*

```
dn: cn=homemanagement,dc=example,dc=org
objectClass: top
```

```
objectClass: posixGroup
cn: homemanagement
gidNumber: 121 # required for posixGroup
memberUid: uid=tuser,ou=people,dc=example,dc=org
memberUid: uid=user2,ou=people,dc=example,dc=org
```

Y luego cambia los atributos de los permisos en slapd.conf:

*Ejemplo 11. ACLs para el Grupo de Administración del Directorio Home*

```
access to dn.subtree="ou=people,dc=example,dc=org"
  attr=homeDirectory
  by dn="cn=homemanagement,dc=example,dc=org"
  dnattr=memberUid write
```

Ahora el usuario `tuser` y el `user2` pueden cambiar los directorios home del otro.

En este ejemplo hemos concedido un subconjunto de poderes administrativos a algunos usuarios sin darles poder en otros dominios. La idea es que pronto ninguna cuenta de usuario tenga el poder de la cuenta de `root`, pero cada poder que tenga root lo tiene como mínimo algún otro usuario. Entonces la cuenta `root` se hace innecesaria y se puede eliminar.

## 4.3. Almacenamiento de Contraseña

OpenLDAP almacenará por defecto el valor del atributo `userPssword` de la misma forma que cualquier otro dato: en plano. La mayoría de las veces está codificado en base 64 lo que proporciona suficiente protección para evitar que un administrador honesto conozca tu contraseña, pero poco más.

Por lo tanto, es buena idea almacenar las contraseñas en un formato más seguro, como SSHA (salted SHA). Esto lo hace cualquier programa que uses para cambiar las contraseñas de los usuarios.

# Apéndice A: Consideraciones Útiles

Hay otros programas que pueden ser útiles, especialmente si tienes muchos usuarios y no quieres configurarlo todo manualmente.

security/pam_mkhomedir es un módulo de PAM que siempre funciona; su propósito es crear directorios home para los usuarios que no los tienen. Si tienes docenas de servidores cliente y cientos de usuarios, es mucho más fácil usarlo y configurar un directorio tipo plantilla para cada directorio home.

sysutils/cpu es una utilidad tipo pw(8) que se puede usar para gestionar usuarios en el directorio LDAP. Puedes llamarlo directamente o envolverlo en un script. Puede gestionar tanto TLS (con el

flag `-x`) como SSL (directamente).

sysutils/ldapvi es una utilidad de gran ayuda para editar valores LDAP en una sintaxis similar a LDIF. El directorio (o subsección del directorio) se muestra en el editor elegido por la variable de entorno `EDITOR`. Esto facilita la realización de cambios de directorio a gran escala sin escribir una herramienta personalizada.

security/openssh-portable tienen la capacidad de contactar con un servidor LDAP para verificar claves SSH. Esto es realmente útil si tienes muchos servidores y no quieres copiar tus claves públicas a todos ellos.

# Apéndice B: Certificados OpenSSL para LDAP

Si alojas dos o más servidores LDAP, probablemente no quieras utilizar certificados autofirmados, ya que cada cliente deberá estar configurado para funcionar con cada certificado. Si bien esto es posible, no es tan simple como crear tu propia autoridad de certificación y firmar con ella los certificados de tus servidores.

Los pasos se muestran aquí tal cual, sin ninguna intención de explicar lo que hacen - se puede encontrar más información en openssl(1) y amigos.

Para crear una autoridad de certificación, simplemente necesitamos un certificado autofirmado y una clave. De nuevo, las instrucciones son

*Ejemplo 12. Crear un Certificado*

```
% openssl genrsa -out root.key 1024
% openssl req -new -key root.key -out root.csr
% openssl x509 -req -days 1024 -in root.csr -signkey root.key -out root.crt
```

Estos serán tu clave CA y certificado root. Probablemente quieras cifrar la clave y almacenarla en un lugar freso y seco; cualquier persona con acceso a ella puede hacerse pasar por uno de tus servidores LDAP.

A continuación, utilizando los dos pasos anteriores, crea la clave ldap-server-one.key y la solicitud de firma de certificado ldap-server-one.csr. Una vez que firmes la solicitud con la clave root.key, podrás usar ldap-server-one.* en tus servidores LDAP.



No olvides utilizar un fully qualified domain name (nombre de dominio completamente cualificado) para el atributo "common name" al generar la solicitud de firma del certificado; de lo contrario, los clientes rechazarán la conexión y esto puede ser muy difícil de diagnosticar.

Para firmar la clave utiliza `-CA` y `_CAkey` en lugar de `-signkey`:

*Ejemplo 13. Firmar como Autoridad Certificadora*

```
% openssl x509 -req -days 1024 \
-in ldap-server-one.csr -CA root.crt -CAkey root.key \
-out ldap-server-one.crt
```

El archivo resultante será el certificado que puedes utilizar en sus servidores LDAP.

Por último, para que los clientes confíen en todos tus servidores, distribuye root.crt (el *certificado*, ¡no la clave!) a cada cliente y especifícalo en la directiva `TLSCACertificateFile` de ldap.conf.