Administración básica del directorio:

Ya tenemos el servidor LDAP funcionando y escuchando en el puerto 389 TCP. Ahora deberíamos generar la estructura de entradas de nuestro directorio y rellenarlas de datos.

Procedimiento genérico

La forma más básica de añadir información a un directorio cualquiera es utilizar ficheros de texto cuyo contenido está escrito en el formato LDIF (LDAP Data Interchange Format). El formato básico de una entrada es...:

```
# comentario
dn: <nombre global único>
<atributo>: <valor>
<atributo>: <valor>
...
```

...donde los atributos indicados deben pertenecer a algún objeto perteneciente a algún "schema" previamente reconocido por el servidor para que sean interpretados correctamente (por suerte, todos los objetos que utilizaremos en este documento están definidos en "schemas" que ya vienen incluidos de serie en cualquier software de tipo servidor LDAP actual).

> NOTA: En el caso de utilizar, por ejemplo, el servidor 389DS, los objetos que emplearemos a lo largo de este documento están definidos en el fichero "/usr/share/dirsrv/schema/00core.ldif" (el de tipo "organizationalUnit") y ""/usr/share/dirsrv/schema/10rfc2307.ldif" " (los de tipo "posixGroup", "posixAccount" y "shadowAccount"), etc Un objeto que también podríamos haber utilizado (pero no lo hemos hecho) es el objeto "hostObject" (definido en "/usr/share/dirsrv/schema/60nss-ldap.ldif"), cuyo atributo más importante es "host" (el valor del cual puede ser una IP o un nombre DNS) y que permitiría alojar en el directorio información sobre las máquinas que formarían un dominio (y, eventualmente, aplicarles reglas de acceso, etc)

Entre dos entradas consecutivas escritas en un archivo LDIF debe existir siempre una línea en blanco. Por otro lado, si una línea es demasiado larga, podemos repartir su contenido entre varias, siempre que las líneas de continuación comiencen con un carácter de tabulación o un espacio en blanco.

Una vez creados estos ficheros, para añadirlos al directorio (incluso con el servidor en marcha) podemos utilizar el comando *ldapadd* (disponible al instalar el paquete llamado “ldap-utils” -en Ubuntu-, o "openldap-clients" -en Fedora- aunque en el caso de haber instalado el paquete "389-ds-base" ya se habrá instalado automáticamente como dependencia). La mayoría de veces necesitaremos indicar los siguientes parámetros:

> *-H ldap://nomDnsServidor:nºpuerto* : Indica el servidor LDAP (y, opcionalmente, el número de puerto) contra el cual se van a ejecutar. Si el servidor LDAP estuviera funcionando en la misma máquina donde se ejecuta la comanda *ldapadd* (o cualquier otra de su familia: *ldapsearch, ldapmodify*, etc), normalmente este parámetro se podrá omitir. Si el servidor LDAP funciona sobre TLS (es decir, es un servidor LDAP seguro), la URL a indicar deberá ser
> *-H ldaps://nomDnsServidor:nºpuerto*
>
> > NOTA: Si el comando cliente (cualquiera de los que estudiaremos: *ldapadd, ldapsearch, ldapmodify*, etc) se ejecuta en la misma máquina donde está funcionando el servidor, en vez de conectar con éste mediante TCP (que es lo que pasa cuando se usa la url del tipo ldap://...) automáticamente utiliza sockets internos que emplean el mecanismo IPC, que es un sistema de intercomunicación entre procesos locales más óptimo para estos casos. Aunque ya hemos dicho que no sería necesario, si se quisieraespecificar la url, en este caso entonces se debería escribir así: "ldapi://%2fruta%2fcarpeta%2ffichero.socket" (notar que el protocolo es "ldapi" y no "ldap" y que las "/" se sustituyen por "%2f").
> >
> > NOTA: En el caso de contactar con un servidor LDAPS (es decir, funcionado sobre TLS), dicho servidor puede o bien ofrecer un certificado autofirmado o bien un certificado firmado por una determinada CA (Autoridad de Certificación). En el primer caso, para que los clientes admitan ese servidor como seguro se ha establecer previamente la variable de entorno LDAPTLS_REQCERT al valor "never" (simplemente precediendo el comando en cuestión con la expresión ***LDAPTLS_REQCERT=never*** ya sería suficiente). En el segundo caso, se ha de establecer previamente la variable de entorno LDAPTLS_CACERT indicando la ruta del certificado de la CA (simplemente precediendo el comando en cuestión con la expresión ***LDAPTLS_CACERT=/etc/dirsrv/slapd-miservidor/ca.crt***, por ejemplo, ya sería suficiente). Para que estas variables sean permanentes, se pueden indicar dentro del archivo "/etc/openldap/ldap.conf"

-D cn=admin : Indica el "common name" de la cuenta de usuario (guardada en el propio servidor) con la que nos autenticaremos en el servidor LDAP para realizar la modificación del directorio; esta cuenta ha de tener privilegio para ello, así que usaremos la cuenta "admin" que creamos en la instalación del servidor.

-W : Solicita interactivamente la contraseña de la cuenta anterior. Otra opción sería indicar la contraseña como un parámetro más, así : *-w contraseña* (notar que en este caso la "w" es minúscula).

-f fichero.ldif : Indica el fichero cuyo contenido se desea agregar

1.- a) Crea un fichero llamado “base.ldif” con el contenido mostrado a continuación y seguidamente agrégalo al directorio con el comando: ***ldapadd -D cn=admin -W -f base.ldif*** . Con esto habrás generado dos entradas de tipo “unidad organizativas” que servirán para contener (a modo de “carpetas”) los usuarios y grupos que generaremos a continuación.

```
dn: ou=usuarios,dc=midominio,dc=local
objectClass: organizationalUnit
ou: usuarios

dn: ou=grupos,dc=midominio,dc=local
objectClass: organizationalUnit
ou: grupos
```

b) Crea un fichero llamado “grupos.ldif” con el siguiente contenido y seguidamente agrégalo al directorio con un comando similar al del apartado anterior:

```
dn: cn=grupoldap,ou=grupos,dc=midominio,dc=local
objectClass: posixGroup
cn: grupoldap
gidNumber: 10000
```

c) Crea un fichero llamado “usuarios.ldif” con el siguiente contenido y seguidamente agrégalo al directorio con un comando similar al del apartado anterior:

```
dn: uid=usu1ldap,ou=usuarios,dc=midominio,dc=local
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
uid: usu1ldap
sn: Lopez
givenName: Juan
cn: Juan Lopez
displayName: Juan Lopez
uidNumber: 3000
gidNumber: 10000
userPassword: XXX
gecos: Es muy tonto
loginShell: /bin/bash
homeDirectory: /home/jlopez
shadowExpire: -1
shadowFlag: 0
shadowWarning: 7
shadowMin: 8
shadowMax: 999999
shadowLastChange: 10877
mail: juan.lopez@gmail.com
postalCode: 29000
####################LINEA EN BLANCO#######################
dn: uid=usu2ldap,ou=usuarios,dc=midominio,dc=local
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
```

uid: usu2ldap
sn: Perez
givenName: Perico
cn: Pedro Perez
displayName: Pedro Perez
uidNumber: 3001
gidNumber: 10000
userPassword: XXX
gecos: Es un crack
loginShell: /bin/bash
homeDirectory: /home/pperez
shadowExpire: -1
shadowFlag: 0
shadowWarning: 7
shadowMin: 8
shadowMax: 999999
shadowLastChange: 10877
mail: pedrito@yahoo.es
postalCode: 29001

NOTA: Tal como se puede ver, cada objeto "usuario" está formado a partir de la unión de diferentes tipos predefinidos de objeto (posixAccount, shadowAccount, inetOgPerson), donde cada uno aporta un determinado conjunto de atributos: posixAccount incluye la información que encontraríamos en el archivo "/etc/passwd" clásico, shadowAccount incluye la información que encontraríamos en el archivo "/etc/shadow" clásico y inetOrgPerson incluye información extra del usuario dentro de la organización (como el correo, cód. Postal...).

NOTA: Hay que tener muy en cuenta que al añadir nuevos usuarios los valores de los atributos uidNumber y homeDirectory (además de userPassword) deben ser diferentes para cada usuario. Lo mismo ocurre con el atributo gidNumber para los grupos. Además, los valores de uidNumber y gidNumber no deben coincidir con el uid y gid de ningún usuario y grupo local de los clientes.

NOTA: El valor del atributo userPassword está protegido por ACLs predefinidas del servidor para que tan solo el usuario administrador - o el propietario del objeto que contenga dicho atributo- pueda consultar su valor (hasheado). Internamente, los valores de todos atributos se almacenan en el backend seleccionado pero en el caso del atributo userPassword este almacenamiento se realiza del hash de la contraseña según el algoritmo predefinido que el servidor tenga configurado. Así pues, un intruso que tuviera acceso directo al backend no obtendría (de entrada) las contraseñas porque las tendría que "crackear". Este aspecto es independiente del hecho de que las conexiones entre cliente y servidor deberían siempre estar encriptadas mediante TLS para evitar el robo de la contraseña en tránsito.

No obstante, alguien podría objetar que escribir la contraseña de los usuarios tal cual en un archivo ldif es un gran riesgo de seguridad. En realidad, así es como se recomienda que se escriba este valor para dejar al servidor LDAP realizar el hash de la forma que internamente requiera con las comprobaciones pertinentes. De todas formas, si se desea, es posible escribir el valor del atributo userPassword de forma que esté "hasheado". Para ello primero hay que utilizar alguna utilidad que nos genere el hash necesario; en el caso de usar 389DS disponemos del comando *pwdhash* , el cual, mediante su parámetro *-s* permite especificar el algoritmo hash deseado. Por ejemplo, para obtener el hash SHA-512 salteado (el mismo algoritmo que se utiliza actualmente para guardar las contraseñas hasheadas en el archivo "/etc/shadow" de la mayoría de distribuciones Linux) de la contraseña "hola" deberíamos ejecutar ***pwdhash -s SSHA512 hola*** El valor obtenido debería ser asignado tal cual al campo userPassword; la parte entre llaves indica al servidor LDAP que dicho valor será un hash de ese tipo y que, por tanto, no deberá hashearlo él sino almacenarlo directamente.

Para comprobar si el contenido anterior se ha añadido correctamente, podemos usar el comando *ldapsearch* (también del paquete "ldap-utils"), el cual permite hacer una búsqueda en el directorio. La mayoría de veces necesitaremos indicar, además de los parámetros *-H ldap://nomDnsServidor:nºpuerto, -D cn=admin* y *-W/-w contraseña* ya conocidos, otros como los siguientes:

-LLL : Indica a *ldapsearch* que muestre la respuesta en modo "no verboso" (más fácil de leer)

-b "dc=midominio,dc=local" : Indica el DN "base" a partir del cual se empezará la búsqueda por las entradas inferiores del árbol. El ejemplo anterior buscaría a partir de la entrada raíz "para abajo" por todas las subentradas. Pero no es necesario que la entrada raíz sea el DN "base": si, por ejemplo, escribiéramos *-b "ou=grupos,dc=midominio,dc=local"* entonces solo se buscaría a partir de la organizationalUnit "grupos" "para abajo" por las subentradas que contuviera.

NOTA: En el servidor 389DS, l'entrada "cn=config" pot fer-se servir com a DN base per obtenir tota la configuració del propi servidor

-s {base | one | sub } : El comportamiento descrito en el párrafo anterior de buscar recursivamente todas las entradas por debajo de una entrada "base" dada (incluyendo ésta) es el comportamiento por defecto, que equivaldría a indicar el parámetro *-s sub* Pero otros valores de este parámetro modifican el "scope" de la búsqueda; por ejemplo, *-s base* solamente muestra información de la entrada marcada como "base" y nada más (es decir, no recorre ninguna rama del árbol); por su parte, *-s one* solamente muestra información de las entradas directamente colgando de la entrada "base" pero nada más (es decir, sólo muestra las entradas "hijas": ni muestra entradas "nietas" ni la propia entrada "base")

"(atributo=valor)" "(atributo=valor)" ... **:** Filtro que permite "quedarse", de todas las entradas recorridas, solo con las que contienen el atributo indicado con el valor indicado. Existen otros filtros más sofisticados que se pueden estudiar en el cuadro inferior. Si aparece este parámetro, ha de ir escrito después de cualquier otro parámetro de los anteriores. Si no aparece, todas las entradas recorridas serán "válidas" para mostrarse.

NOTA: * is a special value representing “any possible value”. It can be used to build a "presence" filter that requests all objects where the attribute is present and has a valid value, but we do not care what the value is. For instance, by default, *ldapsearch* provides the filter "(objectClass=*)"; because all objects must have an objectClass, this filter is the equivalent to saying “all valid objects”.

atributo atributo ... : De la información encontrada en las entradas recorridas (y ya filtradas, si fuera el caso), se muestra solo aquellos atributos indicados. Si aparece este parámetro, ha de ir escrito después del filtro (si hubiera). Si no aparece -o se escribe "*"-, se mostrarán todos los atributos pertenecientes a las entradas recorridas (y ya filtradas si fuera el caso); en el caso de escribir "+" se mostrarán los metaatributos de la entradas recorridas (como la fecha de creación de la entrada, el usuario que la creó, etc).

NOTA: To show what basedns are available, you can query the special ‘’ or blank rootDSE (Directory Server Entry) using "namingContexts" as the attribute to look up, like this:
ldapsearch -D cn=admin -W -LLL -b "" -s base namingContexts

2.- Ejecuta el comando ***ldapsearch -D cn=admin -W -LLL -b "dc=midominio,dc=local" uid=usu1ldap sn givenName*** Con este comando de ejemplo estaremos buscando un usuario con uid=usu1ldap y pediremos que nos muestre solo el contenido de los atributos sn y givenName. Comprueba que efectivamente sea así:

NOTA: Es posible realizar consultas anónimas (es decir, sin necesidad de autenticarse) pero para ello sería necesario configurar el servidor 389DS convenientemente (ver para más información) y sustituir los parámetros *-D* y *-W* de *ldapsearch* por *-x*

*A filter can request objects whose attribute values are greater/less than a value by *"(uid>=test0005)"* or *"(uid<=test0005)"*. If used with letters, it compares alphabetically.

A filter can request a partial match of an attribute value on the object by using the ‘’ operator (multiple times if necessary): *"(uid=*005)"* or *"(uid=*st000*)"*. Note you should always have at least 3 characters in your substring filter, else indexes may not operate efficently.

*Filters can be nested with AND (&) or OR (|) conditions. Condition applies to all filters that follow within the same level of brackets, that is: *(condition (attribute=value)(attribute=value))*.
- AND requires that for an object to match, all filter elements must match; this is the “intersection” operation and is written like this: *"(&(uid=test0006)(uid=guest0006))"*
- OR filters will return the aggregate of all filters; this is the union operation so provided an object satisfies one condition of the OR, it will be part of the returned set; it's written like this: *"(|(uid=test0006)(uid=guest0007))"*

*A NOT filter acts to invert the result of the inner set. For example: *"(!(uid=test0010))"* Note you can’t list multiple parameters in a "not" condition: to combine NOT's you need to use this in conjunctionwith AND and OR.

NOTA: *You can nest AND, OR and NOT filters to produce more complex directed queries. For instance this query:
"(&(objectClass=person)(objectClass=posixAccount)(|(uid=test000))(!(uid=test0001)))"* would be equivalent to...:

```
(&
   (objectClass=person)
   (objectClass=posixAccount)
   (|
      (uid=test000*)
   )
   (!(uid=test0001))
)
```

...and it expresses "All person whose name starts with test000* and not test0001".
Another example would be this: *"(|(& (...K1...) (...K2...))(& (...K3...) (...K4...)))*", which means "(K1 AND K2) OR (K3 AND K4)"

Otros comandos importantes del paquete "ldap-util" son *ldapdelete* y *ldapmodify*. Un ejemplo del primero podría ser: ***ldapdelete -D cn=admin -W "uid=usu2ldap,ou=usuarios,dc=midominio,dc=local"***, donde se ha indicado el DN del elemento que se desea eliminar. El segundo tiene tres formas de modificar una entrada: cambiando el valor de un atributo , añadiendo un nuevo atributo o eliminando un atributo existente:

1.-Para cambiar, por ejemplo, el atributo *uidNumber* de un usuario, podríamos ejecutar el comando ***ldapmodify -D cn=admin -W -f fichero.cambios*** donde "fichero.cambios" debería tener un contenido como el siguiente (donde se especifica qué entradas se quieren modificar y de qué manera):

```
dn:uid=usu2ldap,ou=usuarios,dc=midominio,dc=local
changetype:modify
replace:uidNumber
uidNumber:3002
```

NOTA: La información anterior la podríamos haber introducido directamente desde la entrada estándar si no hubiéramos especificado el parámetro -f.

2.-Para añadir un atributo nuevo (en este caso llamado *jpegPhoto*) deberíamos ejecutar el mismo comando *ldapmodify* anterior pero ahora "fichero.cambios" debería tener un contenido como este:

```
dn:uid=usu2ldap,ou=usuarios,dc=midominio,dc=local
changetype: modify
add:jpegPhoto
jpegPhoto:file:///tmp/foto.png
```

3.-Para eliminar un atributo (en este caso llamado *jpegPhoto*) deberíamos ejecutar el mismo comando *ldapmodify* anterior pero ahora "fichero.cambios" debería tener un contenido como este:

```
dn:uid=usu2ldap,ou=usuarios,dc=midominio,dc=local
changetype: modify
delete:jpegPhoto
```

NOTA: Ja que la configuració del servidor 389DS es troba accessible en forma directori, la comanda *ldapmodify* es podria utilitzar per modificar aquesta configuració "en calent". Per exemple, sabent que les directives "nsslapd-ldapilisten" i "nsslapd-ldapifilepath" activen el mecanisme LDAPI i especifiquen la ruta del socket adient, respectivament, per activar aquesta característica (que ja ve de sèrie activada, però és només un exemple), podríem fer com sempre: *ldapmodify -D cn=admin -W -f fichero.cambios* pero ara el contingut del fitxer "fichero.cambios" seria:

```
dn: cn=config
changetype: modify
replace: nsslapd-ldapilisten
nsslapd-ldapilisten: on
-
add: nsslapd-ldapifilepath
nsslapd-ldapifilepath: /var/run/slapd-exemple.socket
```

NOTA: En realitat, per afegir entrades i per eliminar entrades no caldria fer servir les comandes *ldapadd* i *ldapdelete* respectivament, ja que amb *ldapmodify* ja n'hi hauria prou. En el cas de voler afegir una entrada el fitxer "fichero.cambios" hauria de tenir un contingut semblant al següent...:

dn: <dn to add>
changetype: add
objectclass: ...
attribute1: ...
attribute2: ...

...i en el cas de voler eliminar una entrada, el seu contingut hauria de ser semblant a aquest:

dn: <dn to delete>
changetype: delete

3.-a) Modifica el atributo “gecos” del usuario Juan López para que muestre la descripción: “Ronca”. Comprueba mediante *ldapsearch* que el cambio lo has realizado correctamente

b) Añade el atributo “jpegPhoto” del usuario Juan López indicando la ruta (ficticia) de su foto identificativa. Comprueba mediante *ldapsearch* que el cambio lo has realizado correctamente

c) Elimina el atributo “jpegPhoto” anterior.Comprueba mediante *ldapsearch* que el cambio lo has realizado correctamente

ACIs

By default, the directory server denies access to everything. The administrator must allow certain types of access to certain resources for users to be able to use the directory server. The operational attribute *aci* is used for access control. This attribute can be applied to any entry in the directory tree, and has subtree scope: it applies to the entry that contains it and any children and descendants of that entry.

An ACI consists of 3 parts: a target, a subject, and the type of access (permission):

*The target represents the entry or resource is being protected. The target specification should be a DN (written with the syntax "*ldap:///DN*"), but can be stretched by specifying one or more atributes to restrict its application, or even a LDAP filter. Writing a target it's optional: by default it applies to all entries in subtree scope of the entry which ACI is defined (that's is which contains the *aci* attribute).

*The subject represents the entity which is granted (or denied) access to the target. The subject may be **userdn="ldap:///DN"** or **groupdn="ldap:///DN"** (where DN can be any DN of a user or a group, respectively). It also can be some of these values: **userdn="ldap:///self"** (to specify the user who has done the query), **userdn="ldap:///anyone"** (to specify anonymous binds), or **userdn="ldap:///all"** (to specify any authenticated user). Modifiers can be used to allow or deny access based on the client IP address or hostname, access time, or connection type (encrypted or not). You can also use attributes to specify a subject with *userattr="attrname"* syntax.

*The permission is any combination of the following keywords:

read	Indicates whether directory data may be read.
write	Indicates whether directory data may be changed or created. This permission also allows directory data to be deleted but not the entry itself: to delete an entire entry, the user must have delete permissions.
search	Indicates whether the directory data can be searched. This differs from the read permission in that read allows directory data to be viewed if it is returned as part of a search operation. For example, if searching for common names is allowed as well as read permission for a person's room number, then the room number can be returned as part of the common name

search, but the room number itself cannot be used as the subject of a search. Use this combination to prevent people from searching the directory to see who sits in a particular room.

add Indicates whether child entries beneath the targeted entry can be created.

delete Indicates whether targeted entry can be deleted.

all Means all types of above accesses

The permission is modified by using **allow** or **deny** If there are conflicts between what is allowed and what is denied, the deny wins. Better explained: when a user attempts any kind of access to a directory entry, Directory Server applies the *precedence rule*. This rule states that when two conflicting permissions exist, the permission that denies access always takes precedence over the permission that grants access. For example, if write permission is denied at the directory's root level, and that permission is applied to everyone accessing the directory, then no user can write to the directory regardless of any other permissions that may allow write access. To allow a specific user write permissions to the directory, the scope of the original deny-for-write has to be set so that it does not include that user. Then, there must be additional allow-for-write permission for the user in question.

The aci attribute has the following syntax (*version 3.0* and *acl "name"* values are mandatory and, as it can be seen, several targets can be specified as well as several *allow(permission)(subject)*... values):

aci: (target="ldap:///DN)(target=...)(version 3.0; acl "name"; allow(per,miss,ion)(userdn="ldap:///subject"); allow...;)

NOTA: Some interesting modifiers to add in a ACI to restrict targeted entries are "targetattr" and "targetfilter" keywords, (which must be specified after these targeted entries). Another interesting keyword is "targetscope"

For instance, in the following example, the ACI states that "bjensen" has rights to modify all attributes in her own directory entry: *aci: (target="ldap:///uid=bjensen,dc=example,dc=com") (targetattr="*")(targetscope="subtree")(version 3.0; acl "example"; allow (write)(userdn="ldap:///self");)*

Access control rules can be placed on any entry in the directory although administrators often place access control rules on entries with the object classes "domainComponent", "organizationalUnit", "inetOrgPerson", or "group". You can add/replace/delete an *aci* attribute like any other using LDIF files and *ldapmodify* command. For instance, this content would add the permission of anonymous searches/reads to all the directory's content:

```
dn:dc=midominio,dc=local
changetype: modify
add:aci
aci: (targetattr!="userPassword")(version 3.0; acl "xxx"; allow(read,search)(userdn="ldap:///anyone");)
```

The *aci* attribute is a multi-valued operational attribute that can be read and modified by directory users. Therefore, the ACI attribute itself should be protected by ACIs. Administration users are usually given full access to the *aci* attribute doing *ldapsearch -D cn=admin -W -b entryDN -s base "(objectclass=*)" aci*

NOTA: En cualquier caso, recomiendo estudiar los ejemplos mostrados en https://access.redhat.com/documentation/en-US/Red_Hat_Directory_Server/10/html/Administration_Guide/Managing_Access_Control-Access_Control_Usage_Examples.html#Access_Control_Usage_Examples-Granting_Anonymous_Access
y, en general https://access.redhat.com/documentation/en-US/Red_Hat_Directory_Server/10/html/Administration_Guide/Managing_Access_Control-Creating_ACIs_Manually.html

4.-a) Aconsegueix que es puguin realitzar recerques anònimes al llarg de tot el directori (l'ACI necessària a aplicar amb *ldapmodify* es troba als paràgrafs de teoria anteriors). Prova-ho executant la comanda ***ldapsearch -x -LLL -b "dc=midominio,dc=local" uid=usu1ldap sn givenName*** (que és la mateixa que ja vas emprar a l'exercici 2 però aquest cop sense haver d'especificar cap usuari, ni administrador ni cap altre).

NOTA: Fixa't que s'ha afegit el paràmetre *-x* per indicar que es vol fer servir autenticació simple (en aquest cas, anònima)

b) Imagina que vols que l'usuari "usu2ldap" pugui veure o modificar atributs de qualsevol usuari existent al directori però no pas que pugui afegir-ne o eliminar-ne. Per aconseguir això podries aplicar una ACI a aquest usuari en concret o crear un grup (anomenat per exemple "adminsDesegona"), afegir l'usuari a aquest grup i aplicar la ACI a aquest grup. Si esculls la primera opció (més fàcil però menys escalable), ¿creus que aquesta ACI seria correcta?

aci: (target="ldap:///ou=usuarios,dc=midominio,dc=local")(version 3.0; acl "yyy";
allow(read,write)(userdn="ldap:///uid=usu2ldap,ou=usuarios,dc=midominio,dc=local");)

bII) Comprovar-ho. Per això hauries d'aplicar aquesta ACI amb *ldapmodify* i seguidament utilitzar la mateixa comanda per provar de canviar algun atribut d'algun usuari (hauries de poder) i/o provar d'eliminar un usuari o afegir-ne un de nou (no hauries de poder).

c) ¿Per a què serviria la següent ACI?

aci:(target="ldap:///dc=example,dc=com")(targetattr="departmentNumber||manager")
(targetfilter="(businessCategory=group1)")(version3.0;acl"group1-admins-write";
allow(write)(groupdn="ldap:///cn=group1admins,dc=example,dc=com");)

5.-Ja hem habilitat la possibilitat de realitzar consultes anònimes, però en tot cas, ens agradaria que aquestes consultes sempre "viatgessin a través del cable" encriptades amb TLS. Això es pot fer, tal com ja sabem, afegint a les comandes client (com *ldapsearch*, etc) el paràmetre -H amb la URL explícita "ldap**s**://..." (o si, la consulta es fa a la mateixa màquina local on es troba funcionant el servidor, simplement afegint el paràmetre -Z). Però per assegurar-nos, podríem fer que el nostre servidor només fos LDAPS i no LDAP (és a dir, tancar el port 389 i només quedar-nos amb el 686). Això es pot fer de la següent manera:

a) Atura el servidor per tal de poder fer a continuació els canvis de configuració pertinents: *sudo systemctl stop dirsrv@miservidor*

b) Obre l'arxiu "/etc/dirsrv/slapd-*miservidor*/dse.ldif" amb un editor de text (com a root) i canvia el valor de la línia "nsslapd-port" per a què sigui 0 (en comptes del valor actual, que deu ser 389).

c) Reinicia el servidor (*sudo systemctl start dirsrv@miservidor*) i comprova amb la comanda *ss -tln* que ja no es troba escoltant el port 389.

d) Repeteix la comanda de l'apartat a) de l'exercici anterior (afegint el paràmetre *-Z*): hauries d'obtenir el mateix resultat

NOTA: Com que el servidor LDAPS per defecte ofereix un certificat autosignat, per no obtenir errors serà necessari establir, abans d'executar qualsevol comanda client, la variable d'entorn LDAPTLS_REQCERT amb el valor "never", així: *LDAPTLS_REQCERT=never*