Autenticación de acceso II: LDAP.

Autor: Enrique V. Bonet Esteban

Introducción.

Lightweight Directory Access Protocol (Protocolo Ligero de Acceso a Directorios) es un conjunto de protocolos usados para acceder a información guardada de forma centralizada en una red. LDAP esta descrito en los RFCs 1777 y 2251 y se basa en X.500[1], aunque tan solo implementa las funciones principales de X.500, de forma que es mucho más sencillo y menos estricto que X.500 y permite su ejecución directamente sobre la red.

La estructura de almacenamiento de la información en LDAP es el servicio de directorio, donde el termino servicio de directorio no se refiere a una estructura para almacenar archivos, etc., tal y como entenderíamos en términos informáticos clásicos, sino que por servicio de directorio debemos entender una base de datos especial, diseñada para frecuentes consultas y muy pocas actualizaciones de datos. Además, un servicio de directorio LDAP, a diferencia de las bases de datos clásicas, no contiene soporte para transacciones o funcionalidad de vuelta atrás (rollback). Por otra parte, los servicios de directorios pueden ser replicados fácilmente, para incrementar su disponibilidad y fiabilidad, permitiéndose incluso la existencia de inconsistencias temporales entre las distintas replicas, siempre y cuando estas inconsistencias acaben siendo eliminadas mediante la sincronización de los servicios de directorios.

En LDAP, un cliente se conecta a un servidor para consultar un servicio de directorio o intentar modificarlo. En caso de una consulta, el servidor puede contestar la consulta o, si no puede contestarla localmente, puede redirigir la consulta a otro servidor LDAP que tenga la respuesta. Si el cliente intenta modificar la información del servicio de directorio de LDAP, el servidor comprueba que el usuario tiene permiso para realizar la operación solicitada y después añade o actualiza la información.

La principal ventaja de usar LDAP es que la información se puede centralizarse, permitiendo una administración más sencilla y eficaz de listas de usuarios, etc. Por otra parte, y dado que LDAP soporta SSL y TLS, es posible su uso mediante protocolos seguros, lo que permite enviar y recibir información de forma segura.

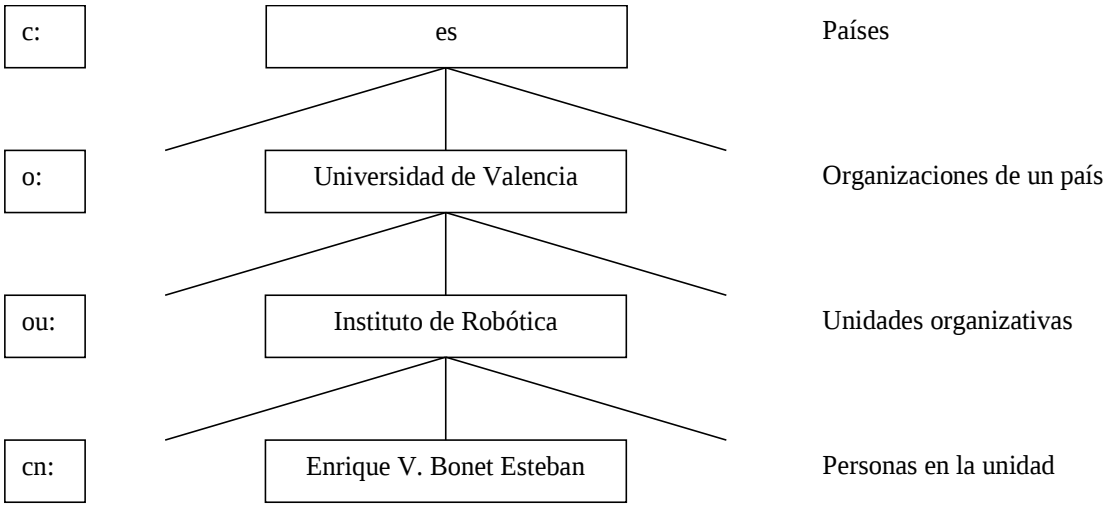
Estructura de la información.

En LDAP, el servicio de directorio debe representar, mediante un modelo abstracto, la información de la empresa, etc., debiendo estar cada elemento identificado por un nombre distintivo (Distinguished Name) que debe ser único dentro de LDAP.

Para asegurar que cada entrada posee un nombre único, la información se puede organizar como el organigrama gráfico de la organización, como la localización geográfica de los elementos que la componen (edificios y oficinas, por ejemplo), como

[1] X.500 es un protocolo definido por ISO como parte del protocolo de interconexión de sistemas abiertos usado como soporte para el correo electrónico basado en X.400.

una estructura plana, etc. Por ejemplo, si queremos situar a una persona dentro de un servicio de directorio de LDAP podemos hacerlo como:



Donde tenemos que el nombre distintivo de esta entrada (dn) esta formado por el nombre de la persona (Common Name), que pertenece a una unidad organizativa (Organization Unit), esta a su vez a una organización de un país (Organization) y, por último, la organización a un país (Country), pudiendo haberse introducidos niveles intermedios entre, por ejemplo, la organización y el país, como pueden ser la provincia, etc.

Aunque existen muchas formas de construir nombres distintivos que aseguren que el nombre de cada elemento es único y por tanto, es posible distinguir de forma univoca los elementos de LDAP, para la construcción de los nombres distintivos de las entradas se suele utilizar el sistema de nombres DNS[2], indicando estos nombres mediante Domain Component.

Así, por ejemplo, si queremos distinguir la entrada anterior construyendo el esquema mediante elementos de tipo dc, y dado que la Universidad de Valencia tiene como dominio de Internet uv.es, pondríamos cn=Enrique V. Bonet Esteban, dc=uv, dc=es, que indica que la persona está dentro de la Universidad de Valencia (uv) y a su vez la UV está dentro de España (es).

Si una estructura de este tipo generá un árbol donde existe un gran número de entradas en la misma rama del árbol, puede ser difícil asegurar que tengamos un nombre distintivo único para cada entrada. En estos casos, puede ser conveniente introducir algún nivel intermedio, no basado en el DNS, como puede ser el de la unidad organizativa. Así, podríamos poner a la persona anterior, dentro de un esquema LDAP como cn=Enrique V. Bonet Esteban, ou=Instituto de Robótica, dc=uv, dc=es, de forma que la unicidad del nombre distintivo de esta entrada es asegurada[3], pues puede existir otra persona con ese mismo nombre en cualquier otra unidad organizativa sin que el nombre distintivo se duplique.

[2] En tema posterior estudiaremos los clientes y servidores de Domain Name Server y veremos como un nombre de dominio es único en todo Internet.
[3] Dividir los esquemas grandes mediante unidades organizativas permite también mejorar el rendimiento de LDAP.

Cada entrada de LDAP puede estar formada por un conjunto de atributos, de forma que un atributo es la información directamente asociada a un elemento. Así, el elemento anterior, que corresponde a una persona, puede tener asociado los atributos de domicilio, número de teléfono, dirección de correo electrónico, etc. En todo elemento es posible especificar que atributos son obligatorios y que atributos son opcionales.

Los ficheros de extensión LDIF.

En LDAP, los ficheros de extensión LDIF corresponden a ficheros de texto UTF-8 que permiten representar, introducir y extraer la información de las entradas de LDAP. Además, los ficheros LDIF permiten especificar modificaciones que deben realizarse en las entradas de LDAP. La sintaxis de los ficheros LDIF que representan la información de LDAP es la siguiente[4]:

```
dn: <distinguished name>
{<attrdesc>: <attrvalue> | <attrdesc>:: <base64-encode-value> |
<attrdesc>:< <URL>}
...
```

Donde cada entrada puede tener tantos *<attrdesc>* como sean necesarios, considerándose toda línea que comienza por el símbolo # como una línea de comentario.

El final de un registro se indica por la inserción después del mismo de una línea en blanco o por el final del fichero, por lo que un fichero LDIF puede contener múltiples registros LDAP, estando cada uno de ellos separado del siguiente por la existencia de esa línea en blanco. Así, por ejemplo, un fichero LDIF con tres registros tendría la estructura:

```
# Comienzo registro 1
dn: <nombre distintivo del registro 1>
<atributo>: <valor del atributo>
...
# Fin registro 1 (le sigue línea en blanco que indica su fin)

# Comienzo registro 2
dn: <nombre distintivo del registro 2>
<atributo>: <valor del atributo>
...
# Fin registro 2 (le sigue línea en blanco que indica su fin)

# Registro 3
dn: <nombre distintivo del registro 2>
<atributo>: <valor del atributo>
...
# Fin registro 3 (le sigue línea en blanco o bien, por ser el
# último registro, el final de fichero).
```

Donde las líneas que comienzan por # son comentarios, como se ha indicado con anterioridad, y han sido puestas solo para facilitar la compresión del ejemplo, pudiendo eliminarse del fichero LDIF pues no aportan nada al mismo.

[4] Una descripción más completa de la especificación de LDIF se puede encontrar en el RFC 2849.

Por su parte, los ficheros LDIF que permiten especificar modificaciones en las entradas de LDAP tienen la sintaxis:

```
dn: <nombre distintivo>
changetype: <{add|modify|delete|modrdn}>
...
```

Donde los valores que continúan dependen de la operación (*changetype*) que se especifique. Así, si *changetype* es *add*, el formato es:

```
dn: <nombre distintivo>
changetype: add
<attrdesc1>: <valor1>
<attrdesc1>: <valor2>
...
<attrdescN>: <valor1>
<attrdescN>: <valor2>
```

Mientras que si *changetype* es *modify* el formato es:

```
dn: <nombre distintivo>
changetype: modify
add: <attributetype>
<attrdesc>: <valor1>
<attrdesc>: <valor2>
```

Para añadir un nuevo atributo y sus valores, o bien:

```
dn: <nombre distintivo>
changetype: modify
replace: <attributetype>
<attrdesc>: <valor1>
<attrdesc>: <valor2>
```

Para reemplazar los valores que tenga un atributo, o bien:

```
dn: <nombre distintivo>
changetype: modify
delete: <attributetype>
<attrdesc>: <valor1>
<attrdesc>: <valor2>
```

Para borrar los atributos que tengan el valor especificado. Tanto en el caso de *replace* como de *delete*, si no se especifica ninguna descripción de atributo (*attrdes*), el atributo entero es eliminado.

Por otra parte, si *changetype* es *delete* el formato es:

```
dn: <nombre distintivo>
changetype: delete
```

No siendo necesario ningún otro valor, pues la entrada con ese nombre distintivo será eliminada. Por último, si changetpe es modrdn, la sintaxis es:

```
dn: <nombre distintivo>
changetype: modrdn
newrdn: <nuevo nombre distintivo>
deleteoldrdn: 0 | 1
[newsuperior: <DN>]
```

Donde el valor 1 en *deleteoldrdn* indica que se borren los valores que tenía la entrada que ha sido modificada, mientras que el valor de 0 indica que no se borren los valores que tenía la entrada modificada. Por último, si se encuentra la línea de *newsuperior* especifica donde debe ser movida la entrada en el árbol de LDAP.

Los fichero de extensión SCHEMA.

La información que se desea introducir en cada uno de los registros (elementos) de LDAP se debe encontrar previamente definida en unos archivos, cuya extensión es SCHEMA, y que son conocidos como esquemas de LDAP. Los esquemas (schemas) contienen la información relativa a los atributos (attributes) y a las clases de objectos (object classes) que podemos almacenar en LDAP. En la configuración de LDAP, que veremos con posterioridad, se deben incluir todos los esquemas que deseemos utilizar[5]. Los esquemas que existen por defecto se encuentran generalmente en el directorio */etc/openldap/schema* o sus subdirectorios.

En los esquemas, los atributos son los elementos que forman parte de un registro de LDAP, por ejemplo, el nombre, apellidos, teléfono, dirección de correo, etc., pudiendo utilizarse cualquier atributo que exista en los esquemas incluidos.

Las clases de objeto describen el tipo de registro que se está intentando construir, indicando que atributos son obligatorios y que atributos son opcionales, de forma que todo atributo debe estar asociado con la clase de objeto apropiada. Un registro puede estar formado por atributos pertenecientes a una o varias clases de objeto.

Los atributos de una clase de objeto pueden tener dos calificadores, *MUST* o *MAY*. Si un atributo tiene el calificador *MUST*, significa que esa clase de objeto obliga a que la presencia de un valor para ese atributo es obligatoria para esa clase de objeto, mientras que si se encuentra con el calificador *MAY*, ese atributo puede tomar o no valor para esa clase de objeto.

Por ejemplo, si encontramos una clase de objeto en la cual el atributo usuario esta calificado como *MUST*, pero el atributo teléfono como *MAY*, para insertar un elemento de esta clase de objeto debemos especificar obligatoriamente el usuario, pero podemos especificar o no su número de teléfono.

Las clases de objeto que se utilizan en los registros de una entrada de LDAP se seleccionan en función de la información que queramos almacenar y de los datos que deseamos que sean obligatorios y opcionales. Por ejemplo, si deseamos almacenar los

[5] La configuración por defecto de LDAP incluye todos los esquemas que se encuentran por defecto en openldap, por lo que si no se crean nuevos esquemas no es necesario tener en cuenta como incluir los esquemas. Información sobre la creación de nuevos esquemas se puede encontrar en el RFC 2522.

datos de un usuario de un ordenador, podemos encontrar en el esquema *nis.schema* la clase de objeto *posixAccount* definida como[6]:

```
object class ( 1.3.6.1.1.1.2.0 NAME 'posixAccount'
      DESC 'Abstraction of an account with POSIX attributes'
      SUP top AUXILIARY
      MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory )
      MAY ( userPassword $ loginShell $ gecos $ description ) )
```

La cual como podemos ver posee obligatorios los atributos *cn* (nombre del usuario), *uid* (nombre), *uidNumber* (identificador del usuario), *gidNumber* (identificador del grupo del usuario) y *homeDirectory* (directorio raíz del usuario) y como opcionales *userPassword* (contraseña del usuario), *loginShell* (shell del usuario), *gecos* (nombre real del usuario) y *description* (descripción de la cuenta).

Y para poder utilizar estos atributos en una entrada LDIF que insertará datos de un registro en LDAP, bastaría con insertar antes de los atributos un tipo de atributo en la entrada LDIF que indicará que se utilice esta clase de objeto. Esto se realiza como:

```
objectclass: <clase de objeto>
```

Que indica que se van a utilizar atributos definidos en la clase de objeto especificada. En el caso de nuestra clase *posixAccount*, la línea sería:

```
objectclass: posixAccount
```

Este estructura, que de entrada parece sencilla, se complica, pues una clase de objeto puede depender estructuralmente de otra clase de objeto, la cual puede encontrarse o no en su esquema. En el ejemplo anterior, la clase objeto *posixAccount* depende de la clase de objeto *account*, por lo que en un fichero LDIF deberíamos insertar las clases de objeto en el orden adecuado. Así, en nuestro caso, el orden de inserción sería:

```
objectclass: account
objectclass: posixAccount
```

Lo cual indica que vamos a utilizar atributos de la clase de objeto *account*, atributos que son necesarios para la clase de objeto *posixAccount*.

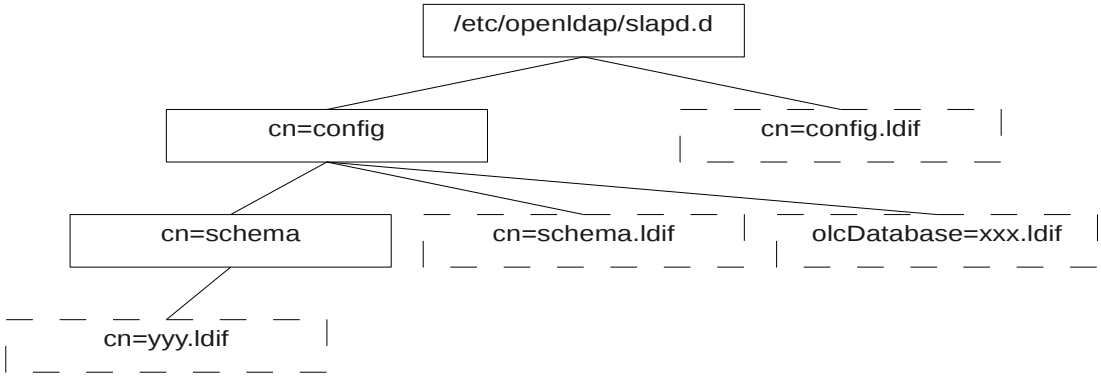
Configuración del servidor.

Previamente a configurar el servidor de LDAP, hemos de crear una autoridad de certificación y un certificado firmado para nuestro ordenador por la citada autoridad de certificación[7]. Utilizaremos *localhost.localdomain* como nombre de nuestro ordenador, pues todas las pruebas que realizaremos serán utilizando el interfaz de loopback y por tanto podemos utilizar el nombre del citado interfaz como nombre de nuestro ordenador.

[6] El ejemplo que aquí se esta exponiendo esta orientado al uso de LDAP para la autenticación de usuarios.
[7] En temas anteriores hemos visto como crear una autoridad de certificación (CA) y como crear un certificado para su firma por una CA y como firmar ese certificado mediante la CA.

El servidor de LDAP es el programa */usr/sbin/slapd*. En las versiones antiguas de LDAP, la configuración se realizaba en el fichero */etc/openldap/slapd.conf*[8], mientras que en la actualidad la configuración se realiza mediante un backend[9] de *slapd* donde se almacena la configuración. Este cambio de formato de configuración permite modificar la configuración de LDAP y aplicar los cambios sin necesidad de rearrancar el servidor de LDAP.

El backend de configuración se encuentra dentro del directorio */etc/openldap/slapd.d*. Dentro de ese directorio podemos encontrar un fichero, *cn=config.ldif* que contiene las opciones globales básicas de configuración del funcionamiento del servidor de LDAP (directorio de configuración, fichero de almacenamiento del identificador de proceso, tamaño de buffers, etc.), y un subdirectorio *cn=config*, dentro del cual se encuentran los ficheros que definen los diferentes tipos de backends existentes y un subdirectorio *cn=schema* que contiene los esquemas básicos de LDAP. Por tanto, la estructura de directorio y ficheros que tenemos puede verse en la figura siguiente:



Los nombres de los backed existentes (ficheros *olcDatabase=xxx.ldif*) se forman con un número entero encerrado entre { y } y el nombre de uno de los tipos de backend validos. Los tipos de backend validos pueden ser los listados en la tabla siguiente o el tipo *frontend*.

Tipo	**Descripción**
bdb	Backend transaccional de tipo de base de datos Berkeley.
config	Backend de configuración de slapd.
dnssrv	Backend DNS SRV (especificación de datos a los DNS de los ordenadores y número de puertos de servicios).
hdb	Variante jerárquica de un backend BDB.
ldap	Backend de proxy LDAP.
ldif	Backend LDIF (formato de intercambio de datos de LDAP).
meta	Backend de meta directorio.
monitor	Backend de monitorización.

[8] Una breve explicación de la configuración del fichero *slapd.conf* se encuentra en el apéndice A.
[9] Por backend nos referimos al tipo especial de base de datos diseñado para frecuentes consultas y muy escasas actualizaciones que hemos visto con anterioridad que utiliza LDAP.

Tipo	Descripción
passwd	Proporciona acceso en modo de solo lectura al fichero de password.
perl	Backend de perl programable.
shell	Backend de shell (programa externo).
sql	Backend programable de SQL.

El tipo *frontend* es un tipo especial de backend que es usado para indicar opciones que deben aplicarse al resto de backends, pudiendo los backend particulares modificar o anular las definiciones de las opciones especificadas por este tipo.

El backend *frontend* y el backend *config* son siempre creados de forma implícita si no han sido configurados de forma explicita, y son cargados antes de cualquier otro tipo de backend.

En nuestro caso particular, dentro del directorio *cn=config* encontramos los backends:

```
olcDatabase={0}config.ldif
olcDatabase={-1}frontend.ldif
olcDatabase={1}monitor.ldif
olcDatabase={2}hdb.ldif
```

Que son creados en la instalación del servidor a partir del fichero */usr/share/openldap-servers/slapd.ldif*, mediante la ejecución del comando:

```
/usr/libexec/openldap/convert-config.sh -f /usr/share/openldap-
servers/slapd.ldif
```

De forma general, la configuración que tiene el fichero *slapd.ldif* debe ser modificada para adaptarla a los requisitos del backend que deseamos crear. En nuestro caso, debemos modificar el fichero slapd.ldif modificando o añadiendo las líneas[10]:

```
...
#
# TLS settings
#
olcTLSCipherSuite: HIGH:MEDIUM:+SSLv2
olcTLSCACertificateFile: /etc/openldap/certs/cacert.pem
olcTLSCertificateFile: /etc/openldap/certs/slapdcert.pem
olcTLSCertificateKeyFile: /etc/openldap/certs/slapdkey.pem
...
#
# Schema settings
#

dn: cn=schema,cn=config
objectClass: olcSchemaConfig
```

[10] Es muy recomendable realizar una copia del fichero slapd.ldif en otra localización y modificar este nuevo fichero, el cual además debe ser de root:root y no tener permisos para que los usuarios puedan leer su contenido.

```
cn: schema

include file:///etc/openldap/schema/core.ldif
include file:///etc/openldap/schema/cosine.ldif
include file:///etc/openldap/schema/nis.ldif
...
#
# Backend database definitions
#
...
olcSuffix: dc=irobot,dc=uv,dc=es
olcRootDN: cn=administrador,dc=irobot,dc=uv,dc=es
olcRootPW: {SSHA}k97qF7R6BGpHzDgZHt6PX3XoeIAjV5Un
...
```

De las líneas que hemos añadido o modificado, las que se encuentran dentro de la configuración de TLS (*TLS settings*) permiten indicar los cifrados permitidos, así como la localización de los certificados de seguridad que LDAP utilizará para transmitir la información de forma segura a través de la red[11].

La opción *olcTLSCipherSuite* indica una lista de cifrados que el servidor seleccionará para negociar la conexión TLS, en orden decreciente de preferencia[12]. Los cifrados los hemos seleccionado en nuestro caso mediante los comodines que OpenSSL soporta. En concreto le hemos indicado en primer lugar cifrados *HIGH*, que son los que utilizan claves superiores a 128 bits. A continuación le indicamos los cifrados *MEDIUM*, que son cifrados con una clave de128 bits, y por último, con *+SSLv2* le indicamos cualquier cifrado de SSL versión 2 sin importar la longitud de la clave.

La línea *olcTLSCACertificateFile* indica la localización de la clave pública de la autoridad de certificación que firma los certificados de nuestro servidor.

Las líneas *olcTLSCertificateFile* y *olcTLSCertificateKeyFile* indican, respectivamente, la localización de la clave pública y privada del certificado que ha sido firmado para nuestro servidor.

Por otra parte, dentro de la configuración de los esquemas (*Schema settings*), hemos añadido dos líneas que permiten incluyen ficheros LDIF que añaden más esquemas al backend que estamos configurando. En nuestro caso hemos añadido las líneas:

```
include file:///etc/openldap/schema/cosine.ldif
include file:///etc/openldap/schema/nis.ldif
```

Que nos permitirán posteriormente definir usuarios, etc., para almacenar su información en el backend.

[11] Los certificados pueden ser firmados por una autoridad de certificación o autofirmados, pero en cualquier caso su propietario debe ser el usuario que ejecuta el servidor de LDAP (generalmente usuario ldap) y tener permisos 0400 para la clave privada y 0644 para la clave pública. En nuestro caso particular los certificados han debido ser creados para localhost.localdomain y firmados por nuestra autoridad de certificación.

[12] Para ver los cifrados que la instalación de OpenSSL soporta se puede ejecutar el comando *openssl ciphers -v ALL*.

Por último, dentro de la definición del backend (*Backend database definitions*) hemos añadido o modificado los atributos *olcSuffix*, *olcRootDN* y *olcRootPW*.

El atributo, *olcSuffix*, indica que consultas responderá este backend. En el ejemplo, se ha indicado como sufijo *irobot.uv.es*, de forma que se responderán a todas las consultas cuyo nombre distintivo (*dn*) sea, por ejemplo, del tipo *cn=XXX*, *dc=irobot*, *dc=uv*, *dc=es*.

Los dos siguientes atributos (*olcRootDN* y *olcRootPW*) indican el usuario y la contraseña que deben ser suministrados por los comandos para realizar acciones administrativas sobre el backend. La contraseña puede guardarse como texto plano, o texto cifrado utilizando crypt, ssha ó md5[13]. Para generar la contraseña se utiliza el comando:

```
slappasswd -h {modo}
```

Donde *modo* puede ser *CRYPT* para cifrar mediante el algoritmo crypt, *SHA* ó *SSHA* para cifrar mediante el algoritmo SHA-1, y *MD5* ó *SMD5* para cifrar mediante el algoritmo MD5[14].

Un atributo que no suele ser necesario modificar, pero que por su naturaleza es necesario explicar, es el atributo *olcDbDirectory*, que indica el directorio donde se almacenará el backend. Generalmente este directorio, que debe existir previamente, suele ser siempre */var/lib/ldap* (valor que tiene el atributo en nuestro ejemplo) y debe tener como propietario al usuario como el que se ejecutará el servidor de LDAP (generalmente usuario *ldap*) y tener como permisos 0700 (permisos de acceso, etc., solo para el propietario del directorio).

Una vez hemos modificado la configuración del fichero *slapd.ldif*, debemos volver a generar la configuración del servidor de LDAP. Para ello, debemos eliminar la configuración que ha sido creada en la instalación, borrando todos los directorios y ficheros que se encuentran dentro de */etc/openldap/slapd.d*, y luego ejecutar el comando:

```
/usr/libexec/openldap/convert-config.sh -f slapd.ldif
```

Es necesario tener en cuenta que el comando debe ejecutarse con el archivo que se ha modificado, que en nuestro ejemplo suponemos que se encuentra en el directorio desde el cual ejecutamos el comando, y que el usuario *ldap* puede acceder al mismo, pues el script *convert-config.sh* ejecuta parte del mismo con los permisos del usuario ldap y no con los de root. En caso de que el script no pueda acceder al fichero, se mostrará el mensaje de error:

```
Configuration conversion failed:
slapd.ldif: Permission denied
```

Una vez hemos configurado correctamente el fichero *slapd.ldif*, y regenerada la configuración del servidor de LDAP, podemos arrancarlo utilizando el comando:

[13] Utilizar una contraseña almacenada en texto plano es completamente desaconsejado por los problemas de seguridad que puede generar.
[14] Si se ejecuta slappasswd sin especificar ningún modo se utiliza por defecto SSHA.

```
systemctl start slapd.service
```

En este momento es necesario resaltar que LDAP sobre Linux funciona por defecto con autenticación mediante SASL[15]. Si la autenticación mediante SASL no es configurada, debe usarse siempre la opción *-x*, en la ejecución de comandos, pues dicha opción indica expresamente que se utilice una autenticación simple en lugar de autenticación mediante SASL.

Podemos ahora comprobar que el servidor ha arrancado correctamente mediante la ejecución del comando:

```
ldapsearch –x –H ldap://localhost.localdomain –b
'dc=irobot,dc=uv,dc=es'
```

Como comentario final, indicar que en el arranque del servidor de ldap, en los mensajes de registro del arranque aparece el mensaje[16]:

```
dbd_db_open: warning – no DB_CONFIG file found in directory
/var/lib/ldap: (2).
```

Dicho mensaje de aviso puede ser ignorado o bien, si se desea eliminar su aparición, copiar el fichero */usr/share/openldap-servers/DB_CONFIG.example* en */var/lib/ldap/DB_CONFIG* con usuario y grupo *ldap* (o el usuario como el que se ejecute el servidor de LDAP)[17].

Inserción de datos.

Una vez hemos configurado el servidor, debemos introducir los datos en el backend. Aunque existen herramientas disponibles para ello, como puede ser *gq*, que se encuentra disponible en la distribución de Fedora 17[18], insertaremos los registros mediante las utilidades contenidas por defecto en LDAP.

En primer lugar, siempre es necesario introducir una primera entrada LDIF que defina el elemento raíz, generalmente la organización, del backend en el que queremos insertar datos. En nuestro caso, como el backend es *"dc=irobot,dc=uv,dc=es"*, crearemos el siguiente archivo LDIF, de nombre, por ejemplo, *registro.ldif*.

```
dn: dc=irobot,dc=uv,dc=es
objectclass: dcObject
objectclass: organization
dc: irobot
o: IRTIC
```

[15] Secure Authentication and Secure Layer es un mecanismo que permite autenticar a un usuario de un servidor y proteger el acceso al mismo.
[16] El mensaje puede obtenerse ejecutando el comando *systemctl status slapd.service*.
[17] En todo este comentario hemos supuesto que el directorio de LDAP es /var/lib/ldap. En cualquier caso, debe corresponder al directorio indicado en la línea Directory de la configuración del backend.
[18] Si se desea instalar la herramienta basta con ejecutar el comando *yum install gq* en una shell del sistema.

Insertando dicho registro en el backend mediante el comando[19]:

```
ldapadd -x –H ldap://localhost.localdomain -D
"cn=administrador,dc=irobot,dc=uv,dc=es" -W -f registro.ldif
```

Pudiendo comprobar que dicho registro se ha insertado correctamente en el backend mediante la ejecución del comando:

```
ldapsearch –x –H ldap://localhost.localdomain –b
'dc=irobot,dc=uv,dc=es'
```

Una vez insertado el elemento raíz, podemos insertar cualquier otro elemento con la información que deseamos que se contenga en el backend. Por ejemplo, si queremos insertar información con los usuarios de los ordenadores de la organización mediante un archivo LDIF, crearemos el siguiente fichero de texto, que llamaremos *miembros.ldif*, en el que se introducen dos elementos[20]:

```
dn: uid=ebonet,dc=irobot,dc=uv,dc=es
objectclass: account
objectclass: posixAccount
uid: ebonet
cn: Enrique V. Bonet Esteban
uidNumber: 1000
gidNumber: 100
homeDirectory: /home/ebonet
loginShell: /bin/bash

dn: uid=mamloba,dc=irobot,dc=uv,dc=es
objectclass: account
objectclass: posixAccount
uid: mamloba
cn: M. Amparo Lopez Ballesteros
uidNumber: 1001
gidNumber: 100
homeDirectory: /home/mamloba
loginShell: /bin/bash
```

Donde hemos indicado el nombre distintivo de la entrada, el identificador del usuario, el nombre del usuario, su identificador de usuario y de grupo, su directorio raíz y la shell que utiliza.

Ejecutando de igual forma que antes el comando:

```
ldapadd -x –H ldap://localhost.localdomain -D
"cn=administrador,dc=irobot,dc=uv,dc=es" -W -f miembros.ldif
```

[19] Todos los comandos de inserción, consulta, etc., pueden utilizarse con la opción -Z para que se ejecuten bajo TLS y por tanto utilicen una conexión segura siempre que se haya configurado previamente el cliente de LDAP para utilizar TLS.

[20] Inicialmente, se recomienda insertar los registros uno a uno, pues si en un archivo existen varios registros, si en uno de ellos ocurre un error, el resto del archivo no será procesado, por lo que esos registros quedarán sin ser incluidos en el backend.

Insertamos todos los miembros existentes, en este caso dos, dentro del backend deseado. Si ahora ejecutamos una consulta como la realizada con anterioridad, podremos comprobar que todas los elementos han sido insertados correctamente.

Inserción de contraseñas.

Hasta ahora hemos visto como insertar datos en el backend. Sin embargo, la información insertada no incluye ningún tipo de dato que podamos decir que es confidencial, como es el caso de una contraseña.

La inserción de contraseñas se realiza mediante el comando *ldappasswd*, el cual permite insertar contraseñas para los elementos de un backend. Así, si deseamos insertar contraseñas para el usuario *ebonet* anterior ejecutaremos el comando:

```
ldappasswd -S -x –H ldap://localhost.localdomain -D
"cn=administrador,dc=irobot,dc=uv,dc=es" -W
"uid=ebonet,dc=irobot,dc=uv,dc=es"
```

El cual solicitará la introducción por teclado de la contraseña del usuario. Podemos comprobar que dicha contraseña se ha insertado en el usuario ejecutando una consulta sobre el backend. En el listado del backend aparece el usuario con un campo *userPassword* y la contraseña introducida en formato cifrado.

Configuración de un cliente de LDAP.

La configuración de un cliente de LDAP se realiza mediante el fichero */etc/openldap/ldap.conf*[21]. Dicho fichero de configuración debe incluir las siguientes líneas[22]:

```
URI             ldap://147.156.222.65
BASE            dc=irobot,dc=uv,dc=es
TLS_REQCERT     never
TLS_CACERT      /etc/openldap/certs/cacert.pem
```

Donde la entrada *URI* tiene la sintaxis:

```
URI             ldap://<nombre|dirección IP>[:puerto]
```

Indicando el nombre o dirección IP del servidor y un puerto opcional si no se utiliza el puerto por defecto. En una misma entrada URI pueden indicarse varios servidores distintos, separando los mismos mediante espacios.

La línea *BASE* indica el sufijo por defecto que se utilizará en las consultas al backend.

Por su parte, la línea *TLS_REQCERT* indica que acción se realizará cuando se solicite y compruebe un certificado de un servidor. Los valores que puede tomar son:

[21] La configuración de un cliente debe hacerse, obviamente, en el ordenador cliente y no en el ordenador servidor, aunque estos pueden coincidir.

[22] Suponemos que el servidor de LDAP se encuentra instalado en el ordenador de dirección IP 147.156.222.65.

- *never*: El cliente nunca solicitará ni comprobará un certificado digital del servidor.

- *allow*: El cliente requerirá un certificado al servidor, continuando la ejecución si el servidor no proporciona un certificado o si este no puede verificarse.

- *try*: El cliente requerirá un certificado al servidor, continuando la ejecución si el servidor no proporciona un certificado, pero terminando la sesión si el certificado no puede verificarse.

- *demand* ó *hard*: El cliente requerirá un certificado al servidor, terminando la ejecución si el servidor no proporciona un certificado o si este no puede ser verificado.

Por último, la línea y TLS_CACERT la clave pública de la autoridad de certificación que debe ser conocida por el cliente de LDAP[23].

Una vez configurado el cliente, podemos comprobar su funcionamiento ejecutando, por ejemplo, la siguiente consulta[24]:

```
ldapsearch -x -Z
```

Donde podemos comprobar como se toma por defecto el servidor especificado en la *URI* y como backend el atributo *BASE* especificado en el fichero de configuración.

Si deseamos ahora obtener la información de una sola entrada en el backend, podemos ejecutar:

```
ldapsearch –x -Z "uid=ebonet"
```

Obteniendo como respuesta tan solo los datos de ese usuario en el backend por defecto del cliente. Si además, deseamos restringir la respuesta a tan solo los atributos que nos interesen, como pueden ser el *uidNumber*, el *gidNumber* y el *userPassword*, ejecutamos:

```
ldapsearch –x -Z "uid=ebonet" uidNumber gidNumber userPassword
```

Control de acceso a la información de LDAP.

Hasta ahora no hemos puesto ningún control al acceso a la información guardada en nuestro servidor de LDAP, de forma que cualquier usuario, desde cualquier ordenador, podía ejecutar una consulta a uno de nuestros backends sin más que conocer la dirección IP de nuestro ordenador y el sufijo del backend que desea consultar.

[23] De forma general las claves públicas deben ser descargadas de Internet, pero en nuestro caso basta con copiar al directorio especificado y con los permisos adecuados, la clave pública de la autoridad de certificación que hemos creado.
[24] A partir de este momento podemos utilizar la opción -Z pues ya hemos configurado el cliente para permitir que utilice conexiones TLS aceptando certificados digitales del servidor.

Esto, que es útil para hacer pública la información, es un problema si lo que estamos almacenando son, como en nuestro ejemplo, identificadores de usuario, su contraseña y demás información asociada. Por ello, hemos de establecer algún mecanismo que nos permita proteger la confidencialidad de la información.

Este mecanismo son las Access Control List (lista de control de acceso), que se especifican en el fichero */etc/openldap/slapd.d/cn=config/olcDatabase={1}hdb.ldif*. Su sintaxis es:

```
olcAccess: to <a que> [by <por quién> <permisos de acceso>
[ <control>] ]+
```

Donde se especifican las condiciones de acceso (especificadas por <*acceso*>) a un conjunto de entradas y/ó atributos (especificados por <*que*>) a uno o más usuarios (especificados por <*quién*>).

El campo <*a que*>.

El campo <*a que*>, que especifica a que entradas se aplican la condiciones de control de acceso, puede tomar uno de los valores:

```
*
dn[.<alcance>]=<DN>
filter=<filtro_ldap>
attrs=<lista_atributos>[val[.<estilo>]=<valor>]
```

El valor * selecciona todas las entradas.

Por su parte, el valor *dn=<DN>* selecciona las entradas basándose en su nombre distintivo y en el <*alcance*> indicado, pudiendo tomar <*alcance*> los valores especificados en la siguiente tabla:

Valor	Descripción
base	Es el valor por defecto, e indica los elementos cuyo dn coinciden con el especificado.
one	Indica elementos cuyo padre es el dn especificado.
subtree	Indica todos los elementos que se encuentran en el subárbol que empieza en el dn especificado, incluyendo el propio dn.
children	Igual que `subtree` pero sin incluir el propio dn.

Por ejemplo, si LDAP contiene las siguientes entradas:

1. `o=Universidad de Valencia`
2. `cn=Enrique V. Bonet Esteban,o=Universidad de Valencia`
3. `ou=Instituto de Robótica,o=Universidad de Valencia`
4. `uid=ebonet,ou=Instituto de Robótica,o=Universidad de Valencia`
5. `uid=mamloba,ou=Instituto de Robótica,o=Universidad de Valencia`

6. `cn=administrador,uid=ebonet,ou=Instituto de Robótica,o=Universidad de Valencia`

Entonces la especificación:

```
dn.base="ou=Instituto de Robótica,o=Universidad de Valencia"
```

Se refiere solo a la tercera entrada, mientras que la especificación:

```
dn.one="ou=Instituto de Robótica,o=Universidad de Valencia"
```

Se refiere a la cuarta y quinta entradas, mientras que la especificación:

```
dn.subtree="ou=Instituto de Robótica,o=Universidad de Valencia"
```

Se refiere a las entradas tercera, cuarta, quinta y sexta, y la especificación:

```
dn.children="ou=Instituto de Robótica,o=Universidad de Valencia"
```

Se refiere a las entradas cuarta, quinta y sexta.

Continuando, el valor *filter=<filtro_ldap>* indica una cadena de caracteres representando un filtro LDAP tal y como se describe en el RFC 2254. Por ejemplo:

```
filter=(objectClass=account)
```

Indica que se filtra para permitir objetos de clase *account*. Destacar, por su utilidad, que una entrada seleccionada mediante su dn puede ser restringida mediante un filtro, como puede verse en el siguiente ejemplo:

```
dn.one="ou=Instituto de Robótica,o=Universidad de Valencia"
filter=(objectClass=account)
```

Por último, la opción *attrs=<lista_atributos>* selecciona los atributos de acuerdo a una lista de nombres de atributos separada por coma, existiendo dos valores especiales, *entry*, que indica la propia entrada, y *children*, que indica las entradas hijo. La lista de atributos puede contener nombre de clases objeto, las cuales se especifican por su nombre con el prefijo @.

La opción *attrs*, si se utiliza con la forma completa:

```
[attrs=<lista_atributos>[val[.<estilo>]=<valor>]]
```

Permite especificar un valor particular de un atributo. En este caso, solo un atributo puede ser indicado en la lista de atributos. El campo *<estilo>* puede tomar los valores *exact* (el valor por defecto), que indica que el atributo debe ser igual que el valor indicado, mientras que si toma el valor *regex*, el valor indicado es considerado una expresión regular.

El campo <por quién>.

El campo <*por quién*>, que especifica a que usuarios se aplica una regla de control de acceso, puede especificarse de múltiples maneras, siendo los más utilizadas[25]:

```
*
anonymous
users
self
dn[.<estilo>]=<valor>
dn.<alcance>=<DN>
```

Pudiendo aparecer múltiples campos <*por quién*> para indicar los diferentes privilegios de acceso que, para el mismo recurso, pueden tener diferentes usuarios.

El valor * indica todo el mundo, mientras que el valor *anonymous* indica clientes no autenticados y suele utilizarse para limitar el acceso a los recursos de autenticación, mientras que *users* se refiere a los clientes autenticados y *self* indica que el acceso a la entrada es permitida a la propia entrada, por ejemplo, cuando un usuario autenticado intenta acceder a su contraseña para modificarla por ejemplo.

Por otra parte:

```
dn[.<estilo>]=<valor>
```

Permite indicar un valor <*valor*>, tomando <*estilo*> los valores *exact* (el valor por defecto), que indica que el atributo debe ser igual que el valor indicado, mientras que si toma el valor *regex*, el valor indicado es considerado una expresión regular.

Por último, la especificación:

```
dn.<alcance>=<DN>
```

Es idéntica a la especificación de igual sintaxis del campo <*qué*> visto con anterioridad pudiendo tomar <*alcance*> los mismos valores, etc.

El campo <permisos de acceso>.

El campo <*permisos de acceso*> especifica el nivel de acceso o los privilegios específicos de acceso que tiene el campo <*por quién*>. Puede tomar los valores:

```
self
<nivel>
<privilegios>
```

Donde *self* indica que se permiten realizar operaciones sobre la entrada a la propia entrada. Por ejemplo, si un elemento pertenece a un grupo de elementos, puede tener la posibilidad de eliminarse de su pertenencia al grupo sin que ello deba afectar al grupo o al resto de miembros del grupo.

Por su parte <*nivel*> es uno de los siguientes valores:

[25] Una descripción de todos los modos de especificación del campo <quién> puede encontrarse en la URL http://www.openldap.org/doc/admin.

Valor	Descripción
none	Sin permisos de acceso.
auth	Acceso solo para autenticación, obteniendo como respuesta si ha sido correcta la autenticación.
compare	Acceso para comparar valores, obteniendo como respuesta si se ha encontrado.
search	Acceso para realizar búsquedas mediante filtros, obteniendo como respuesta si algún valor coincide con el filtro.
read	Acceso para leer los resultados de búsquedas mediante filtros. La respuesta es los valores obtenidos.
write	Acceso total a los datos, permitiendo su escritura, modificación y borrado.

Donde el acceso a un nivel implica todos los privilegios de los niveles inferiores, así, el acceso *read*, por ejemplo, implica también la posibilidad de realizar un acceso *compare*, por ejemplo.

Por último, <*privilegios*> tiene la sintaxis:

```
<privilegios> = {=|+|-}{w|r|s|c|x|0}+
```

Donde = elimina todos los privilegios indicados con anterioridad y comienza su asignación, + añade privilegios a los anteriormente dados y - elimina privilegios. Los privilegios son *w* para escritura, *r* para lectura, *s* para búsqueda, *c* para comparación, *x* para autenticación y *0* para ningún privilegio.

Esta forma de especificación es mucho más exacta que la anterior, pues permite especificar, por ejemplo, que es posible modificar y borrar datos (*w*) sin poder realizar búsquedas en los mismos (*s*). La equivalencia entre la especificación mediante <*nivel*> y mediante <*privilegios*> puede verse en la siguiente tabla:

Nivel	Privilegios
none	=0
auth	=x
compare	=cx
search	=scx
read	=rscx
write	=wrscx

El campo <control>.

El campo <*control*> puede tomar los valores *stop*, *continue* ó *break*, indicando *stop*, el valor por defecto, que si un campo <*quién*> coincide con el solicitado, se dejen de procesar el resto de campos <*quién*> existentes. Por su parte, *continue* indica que se sigan procesando campos <*quién*> aunque este campo sea una coincidencia, lo que permite incrementar los privilegios. Por último, *break* permite que se sigan procesando campos <*acceso*> aunque el encontrado sea una coincidencia. Así, si tenemos la siguiente regla de acceso:

```
olcAccess: to dn.subtree="dc=uv,dc=es" attrs=cn by * =cs
continue by users +r
```

Permitimos el acceso de búsqueda y comparación al atributo cn a todo el mundo del árbol *"dc=uv, dc=es"*, mientras que el permiso de lectura se añade a los usuarios autenticados. Mientras que si escribimos:

```
olcAccess: to dn.subtree="dc=uv,dc=es" attrs=cn by * =cs break
olcAccess: to dn.subtree="id=ebonet,dc=uv,dc=es" by * +r
```

Permitimos el acceso de búsqueda y comparación al atributo cn a todo el árbol *"dc=uv, dc=es"*, mientras que la lectura a solo el usuario de identificador *ebonet* del mismo árbol.

Ejemplo.

Un sencillo ejemplo de control de acceso a un backend que posee contraseñas almacenadas es el siguiente:

```
olcAccess: to attrs=userPassword by
dn="cn=administrador,dc=irobot,dc=uv,dc=es" write by self write
by * auth
olcAccess: to * by dn="cn=administrador,dc=irobot,dc=uv,dc=es"
write by * read
```

Donde podemos ver que protegemos la entrada a la contraseña, impidiendo su lectura excepto para que un usuario pueda modificar su contraseña y para que los clientes puedan autenticar la contraseña, permitiendo el acceso de lectura al resto de atributos del backend[26].

Podemos ahora comprobar como el acceso a la contraseña de los usuarios se encuentra protegida si ejecutamos el comando:

```
ldapsearch -x
```

Pues veremos que nos proporciona los atributos no protegidos de los usuarios (*dn*, *uid*, *cn*, *uidNumber*, *gidNumber*, *homeDirectory* y *loginShell*), pero no nos proporciona el atributo *userPassword*.

Sin embargo, si ejecutamos el comando como:

```
ldapsearch -x -D "cn=administrador,dc=irobot,dc=uv,dc=es" -W
```

Podemos comprobar que ahora si que devuelve el atributo *userPassword* junto con el resto de atributos.

[26] La configuración de cada regla de acceso debe estar en una sola línea, pues en caso contrario se produce un error al arrancar el servidor de LDAP Además, recordar que en los fichero LDIF una línea en blanco significa el final de un registro, por lo que las reglas de control de acceso deben escribirse todas juntas y sin separarse del resto de atributos de configuración del backend.

Ejercicios.

1- Escribir tres dn distintos del profesor de la asignatura (Enrique Bonet) sabiendo su correo y que es profesor del departamento de informática y miembro del instituto de robótica.

2- Un servidor de LDAP debe almacenar las direcciones de correo electrónico de los profesores (uv.es) de los alumnos (alumni.uv.es). Especificar los backend necesarios.

3- Deseamos que un servidor LDAP almacene los usuarios y contraseñas de los alumnos de la asignatura AGR. Crear los ficheros LDIF y ejecutar los comandos necesarios para insertar los datos en el backend, cuyo dn es `"ou=agr,cn=uv,cn=es"`.

4- Configurar un cliente de LDAP del dominio del anterior problema que tan solo permita su consulta mediante una conexión segura.

5- Realizar los cambios necesarios en la configuración de PAM para que verifique los usuarios y contraseñas mediante el uso del servidor de LDAP del ejercicio 3.

6- Escribir la ACL que permite en un backend a todos los elementos y a todos los usuarios pero solo para lectura.

7- Escribir la ACL que permite en un backend que un usuario pueda cambiar sus datos, que los usuarios anónimos puedan autenticarse y que el resto de usuarios puedan leer los datos.

8- Escribir la ACL que permite en un backend que los usuarios del dominio irobot.uv.es puedan buscar en los datos y los usuarios del dominio uv.es puedan leer los datos.

9- Escribir la ACL que permite a un usuario modificar su número de teléfono (atributo homePhone) y a los de los subdominios de uv.es realizar su búsqueda, pudiendo el usuario de igual forma modificar el resto de sus datos, los de los subdominios de uv.es realizar búsquedas y a los anónimos autenticarse.

Apéndice A: Configuración del fichero slapd.conf.

Como hemos indicado con anterioridad, en versiones antiguas de openldap, el fichero de configuración de LDAP es el fichero */etc/openldap/slapd.conf*[27]. En la actualidad un ejemplo del mismo, con el nombre *slapd.conf.obsolete* en el directorio */usr/share/openldap-servers*[28].

El fichero *slapd.conf* esta formado por una serie de líneas, siendo tomadas todas las líneas que comienzan por el carácter # como comentarios, mientras que si una línea comienza por un espacio en blanco es considerada como una continuación de la línea anterior. Además, todo argumento que contenga espacios en blanco debe ir encerrado

[27] El fichero debe tener permisos suficientes para que el usuario ó grupo como el que se ejecuta el servidor de LDAP, generalmente usuario *ldap* y grupo *ldap*, pueda leer el mismo.

[28] Si se desea utilizar este modo de configuración, debe copiarse dicho fichero en el directorio /etc/openldap con el nombre slapd.conf y mover o eliminar el directorio /etc/openldap/slapd.d.

entre comillas dobles, sucediendo que si un argumento contiene las comillas dobles o el carácter \, estos deben ir precedidos del carácter \.[29]

El fichero de configuración esta formado por las opciones globales de configuración y por cero o más definiciones de bases de datos, que son conocidas como backends, con su información de configuración específica.

La configuración de cada uno de los backend se realiza de forma general mediante las siguientes cinco líneas:

```
database        bdb
suffix          "dc=irobot,dc=uv,dc=es"
rootdn          "cn=administrador,dc=irobot,dc=uv,dc=es"
rootpw          {SSHA}ZKuzYcUMeYF3CJKu+XIIjHY+e8/OqT+f
directory       /var/lib/ldap
```

El primer parámetro, *database*, indica el tipo de base de datos se utilizará, normalmente el valor utilizado es *bdb (Berkley Data Base)*, pues es una base de datos dbm rápida, pero *shell* (para scripts) y *passwd* (para utilizar */etc/passwd*) también son opciones válidas.

El segundo parámetro, *suffix*, indica que consultas responderá este backend. En el ejemplo, se ha indicado como sufijo *irobot.uv.es*, de forma que se responderán a todas las consultas cuyo nombre distintivo (dn) sea del tipo *cn=XXX*, *dc=irobot*, *dc=uv*, *dc=es*.

Los dos siguientes parámetros indican el usuario y la contraseña que deben ser suministrados por los comandos para realizar acciones administrativas sobre el backend. La contraseña se se almacena, crea, etc., tal y como se ha comentado con anterioridad para la configuración actual de LDAP.

Por último, el parámetro *directory* indica el directorio donde se creará el backend, debiendo el directorio tener los permisos, etc., indicados en la configuración de LDAP.

La configuración de TLS en el servidor se realiza de forma similar a como se ha comentado con anterioridad, solo que los atributos a modificar se encuentran en la opciones globales, debiendo incluir en el fichero de configuración las líneas[30]:

```
TLSCipherSuite              HIGH:MEDIUM:+SSLv2
TLSCACertificateFile        /etc/openldap/cacert.pem
TLSCertificateFile          /etc/openldap/slapdcert.pem
TLSCertificateKeyFile       /etc/openldap/slapdkey.pem
```

Donde la opción *TLSCipherSuite* indica una lista de cifrados que el servidor seleccionará para negociar la conexión TLS, en orden decreciente de preferencia.

[29] Esto es similar a lo que sucede en múltiples lenguajes de programación como es el caso de C, C++ y Java.

[30] La localización y el nombre de los certificados de seguridad no tienen porque coincidir con los aquí indicados.

La línea *TLSCACertificateFile* indica donde es posible encontrar el certificado de la autoridad de certificación, y solo debe indicarse si el certificado ha sido firmado por una autoridad de certificación y no ha sido autofirmado, pues en tal caso se producirá un error.

Por último, las dos últimas líneas indican la localización del certificado (*TLSCertificateFile*) y de la clave privada (*TLSCertificateKeyFile*).

Si se desea establecer un mecanismo de control de acceso, este se especifica igual que hemos explicado, dentro de la definición de cada uno de los backends, pero siendo la sintaxis:

```
access to <que> [by <quién> <acceso> [ <control>] ]+
```

Donde se especifican las condiciones de acceso (especificadas por <*acceso*>) a un conjunto de entradas y/ó atributos (especificados por <*que*>) a uno o más usuarios (especificados por <*quién*>).

Tomando los campos <*que*>, <*quién*>, <*acceso*> y <*control*> los valores y significados explicados con anterioridad.

Como comentario adicional, no es necesario que las reglas de control de acceso se escriban en una sola línea, pudiendo escribirse en varias líneas. Por ejemplo, la regla de control explicada para la sintaxis actual de configuración, podría escribirse en el fichero de configuración antiguo como:

```
access to attrs=userPassword
     by dn="cn=administrador,dc=irobot,dc=uv,dc=es" write
     by self write
     by * auth
access to *
     by dn="cn=administrador,dc=irobot,dc=uv,dc=es" write
     by * read
```