ANEXO 4: LDAP KRB5

Índice general

Capítulo 1

Almacén de metadatos LDAP, protocolo de autenticación KERBEROS

Contents

El tipo de despliegue de OpenLDAP y MIT Kerberos que vamos a llevar a cabo presenta una serie de interdependencias:

- MIT Kerberos utilizará a OpenLDAP como su almacén de metadatos.
- OpenLDAP utilizará a MIT Kerberos para la autenticación, incluída la relacionada con los proceso de replicación.

Estas interdependencias nos inducen a presentarlos conjuntamente en un mismo capítulo, pues de otra forma es fácil perder el curso lógico del despliegue y no entender la necesidad de cada paso. Como contrapartida, este capítulo es especialmente complejo. Para no perdernos, si lo miramos en su conjunto podemos encontrar este mapa de ruta (aunque al configurar OpenLDAP se entrelaza un poco por simplificar):

- En una primera parte desplegaremos OpenLDAP, es decir, al programa slapd. Cualquier operación que necesite autenticación se realizará de forma simple (sin KERBEROS).
    - Puesto que se ha de modificar el comportamiento por defecto de slapd, se aprovechará ya para crear algunas estructuras que usará más tarde MIT Kerberos.
- En una segunda parte desplegaremos MIT Kerberos: sus componentes AS (Autenticacion Server), TGS (Ticket Granting Server)[1] y su gestor de base de datos. Éste último, ayudado de una autenticación simple, usará a OpenLDAP (ya disponible) como backend de almacenamiento.
- En la tercera y última parte haremos que OpenLDAP use KERBEROS para toda autenticación excepto la vinculada con MIT Kerberos que acabamos de referir.

[1] A veces se suele usar KDC ("Key Distribution Center") para refirirse al TGS, pero estrictamente el KDC comprende a la dupla AS + TGS.

1.1. DKLAB1

1.1.1. Instalación de OpenLDAP

Instalamos el servidor (slapd) así como los clientes (ldapsearch, ldapmodify etc). Adicionalmente, nos aseguramos que entre las librerías del proyecto Cyrus SASL que utilizan los anteriores, esté instalada la que provee autenticación SASL-GSSAPI basada en MIT Kerberos:

```
apt-get install slapd ldap-utils libsasl2-modules-gssapi-mit
```

```
-"Omit OpenLDAP server configuration?"
No
-"DNS domain name to construct the base DN of the LDAP directory":
casafx.dyndns.org
-"Organization name":
casafx.dyndns.org
-"Administrator password:"
ldapadmin       (apenas lo vamos a usar antes de ser eliminado)
-"Database backend to use":
HDB
-"Do you want the database to be removed when slapd is purged?"
No
-"Allow LDAPv2 protocol?"
No
```

```
nmap -p 389 localhost
```

```
mkdir ~/ldif

slapcat -b cn=config                    > ~/ldif/ldap-pristine_cn_config.ldif
slapcat -b dc=casafx,dc=dyndns,dc=org   > ~/ldif/ldap-pristine_dc_casafx.ldif
cp -R /var/lib/ldap                       ~/ldif/ldap-pristine_var-lib-ldap
```

1.1.2. Modificaciones sobre clientes LDAP dependientes de libldap

```
vim /etc/ldap/ldap.conf
```

```
...
####fx
BASE    dc=casafx,dc=dyndns,dc=org
#URI     ldap://dklab1.casafx.dyndns.org ldap://dklab2.casafx.dyndns.org
#...si omitimos URI, obligamos a recurrir a los SRV DNS RR;
#   en principio podemos forzar a que se usen los registros SRV pues so'lo
#   los comandos ldap* fallara'n (u'sese -H o expo'rtese la variable LDAPURI;
#   segu'n man ldap.conf; incluso se pueden exportar LDAPCONF y LDAPRC
#   para ficheros .conf alternativos), pero si surgen problemas con los
#   servicios habri'a que explicitar las uri como ahi'. Referencia:
#   http://www.rjsystems.nl/en/2100-dns-discovery-openldap.php
#
####endfx
```

1.1.3. Modificaciones sobre el servidor LDAP; Directory Information Tree's

Tras la instalación, el árbol único LDAP está dividido a bajo nivel en dos bases de datos o DIT's; el Distinguished Name, DN, de la primera es cn=config y se utiliza para almacenar la configuración de slapd y el resto de DIT's. La segunda base de datos fue creada por debconf nada más le dimos el nombre. En concreto, al responder a la pregunta "DNS domain name to construct the base DN of the LDAP directory" con "casafx.dyndns.org", debconf usó cada subdominio como un nodo de la clase domainComponent (dc) de los esquemas estándar de LDAP, creando por tanto el nodo dc=casafx,dc=dyndns,dc=org que es finalmente la parte del árbol LDAP donde se almacenarán todos los metadatos de los servicios de nuestra organización.

Debconf también creó una nodo que tiene propiedades de usuario-ldap, es decir un nodo que puedes usar para identificarte en el árbol. No lo vamos a utilizar así que vamos a borrala. La cuenta que creó es el nodo "cn=admin,dc=casafx,dc=dyndns,dc=org", por tanto en el DIT "dc=casafx,dc=dyndns,dc=org". Constituye un usuario-ldap porque, si

lo inspeccionásemos con ldapsearch, veríamos que tiene atributos especiales (como organizationalRole para darle un nombre o, más importante aún porque requiere que la entidad tenga un password, la clase simpleSecurityObject, una clase definida entre los esquemas estándar de LDAP; otras posibilidades de objetos que requieren configurar una credencial son los de la clase shadowAccount o los de strongAuthenticationUser). El password asociado a esa entidad que le dimos a debconf fue "ldapadmin", y vamos a aprovecharlo para que slapd nos autentique y autorice a realizar cambios, en este caso el cambio será precisamente borrar esa entidad:

```
ldapdelete -xh localhost -D cn=admin,dc=casafx,dc=dyndns,dc=org \
           -w ldapadmin cn=admin,dc=casafx,dc=dyndns,dc=org
ldapsearch -LLLQY EXTERNAL -H ldapi:/// -b dc=casafx,dc=dyndns,dc=org
```

- El flag -x significa autorización simple (con password y sin intento de negociación SASL).
- El flag -D es el organizationalRole con el que accedemos, -w para el password asociado.
- El último argumento es el nodo sobre el que operamos y, al ser ldapdelete el comando, el que eliminamos.

Tras esta modificación preliminar, vamos con las modificaciones que son nuestro objetivo último ahora. Son necesarias sobre ambos DIT's esta vez. Para llevarlas a cabo utilizaremos ficheros en formato LDIF (LDAP Directory Interexchange Format) que constituyem el formato en texto simple (la alternativa binaria que usan las aplicaciones es ASN.1 codificado) para describir los cambios que deseamos hacer y que, una vez pasados a la utilidad ldapmodify, serán procesados por slapd. Dos ficheros LDIF con nuestros cambios:

- El primero, kerberos.ldif, afecta al DIT cn=config, en concreto a su subdirectorio cn=schema, y consiste en añadir el esquema (es decir, la declaración de objetos y atributos) luego usado por MIT Kerberos para almacenar sus datos en alguna subrama del DIT de nuestra organización (alguna subrama de dc=casafx,dc=dyndns,dc=org).
- El segundo fichero, ldap1_olc-mod.lidf, como se verá, afecta al resto de subdirectorios tanto de cn=config como de dc=casafx,dc=dyndns,dc=org.
  - Crearemos la mencionada subrama que usará MIT Kerberos (su proceso kadmind concretamente), además de dos simpleSecurityObject que usará para sus autenticaciones simples.
  - También configura la posibilidad de autenticación SASL-GSSAPI entre los clientes y el servidor LDAP.
  - Y además, configura la replicación LDAP multi-master entre los dos servidores slapd.

Para ir por partes, vamos a exponer en primer lugar cómo se lleva a cabo la carga de ambos ficheros utilizando a ldapmodify:

```
# Carga de kerberos.lidf:
ldapmodify -a -QY EXTERNAL -H ldapi:/// \
           -f ~/ldif/kerberos.ldif

# Carga de ldap1_olc-mod.lidf:
ldapmodify -QY EXTERNAL -H ldapi:/// \
           -f ~/ldif/ldap1_olc-mod.ldif \
           -c -S ~/ldif/ldiferrors-‘date +%Y%m%d.%H%M_%S‘.ldif
```

- El flag -a indica que ldapmodify tiene simplemente que añadir los objetos y atributos del esquema, sin esperar que se le indique otro comportamiento dentro del fichero. Ésto simplifica enormemente la escritura de kerberos.ldif, si bien lo creará un autómata y no nosotros. Mencionamos que "ldapmodify -a" equivale a "ldapadd".
- -QY EXTERNAL -H ldapi:/// indica que se usará el mecanismo SASL EXTERNAL (-Y) de forma silenciosa (-Q) sobre la interfaz ldapi:/// (-H). Lo que intentamos hacer con ello es utilizar el árbol sin hacer una autenticación basada en simpleSecurityObject y password, tampoco otro mecanismo similar sino que si accedemos al socket /var/run/slapd/ldapi con uid=0 y gid=0, se dé por hecha la fase de autenticación.
- -f <fichero.ldif>para indicarle a ldapmodify el fichero LDIF que cargar.
- -c -S ~/ldif/ldiferrors-‘date + %Y %m %d. %H %M_ %S‘.ldif pasado al segundo comando, es simplemente una forma conveniente de cargar un archivo complejo como es ldap1_olc-mod.ldif; ldapmodify intentará de forma continua (-c) procesar el archivo, pero para cada transacción errónea, antes de pasar a la siguiente, creará un log al respecto (-S) para que podamos revisarlo después específicamente.

Y en segundo lugar, tras conocer cómo cargar los ficheros ldif, vamos a describir ya el contenido de ambos ficheros ldif. Lo haremos recorriendo el árbol tal como está ahora, describiendo qué cambios se hacen en cada rama.

- Aún algunas notas previas:
  - -n -v para comprobar parcialmente la sintaxis: ldapmodify -n -v -f ~/file.ldif
  - un fichero ldif permite incluir ficheros: "include: file:///tmp/example.com.ldif", si bien nosotros no vamos a usar esta posibilidad por claridad de exposición.

- ante errores que nos dejen aparentemente atascados, se puede empezar desde cero con:
  - dpkg-reconfigure -plow slapd, e indicando que elimine las db ante purges. Entonces:
  - apt-get purge slapd, y a continuación
  - apt-get install slapd
- ```
  man slapd
  ```

cn=config

Añadimos un esquema para MIT Kerberos Bajo cn=schema,cn=config almacenaremos el esquema que utiliza MIT Kerberos para definir sus datos.

Bajaremos un paquete de donde poder extraerlo. Este kerberos.schema se encuentra en un formato diferente a LDIF, luego hemos de transformarlo para poder cargarlo en cn=schema,cn=config. El autómata encargado de ello es el propio slapd, que dispone de una funcionalidad a este respecto.
  ```

```
cd /var/cache/apt/archives/
apt-get download krb5-kdc-ldap
dpkg-deb --fsys-tarfile krb5-kdc-ldap_* | \
 tar Ox --wildcards '*/usr/share/doc/krb5-kdc-ldap/kerberos.schema.gz'  | \
 zcat > /etc/ldap/schema/kerberos.schema

cd ~
mkdir ~/ldif/krb_schema/

echo "include /etc/ldap/schema/kerberos.schema" \
     > ~/ldif/krb_schema/schema_convert.conf
slapcat -f ~/ldif/krb_schema/schema_convert.conf -F ~/ldif/krb_schema/ \
        -s "cn=kerberos,cn=schema,cn=config"

cp ~/ldif/krb_schema/cn\=config/cn\=schema/cn\=\{0\}kerberos.ldif \
   ~/ldif/kerberos.ldif
```

Aún hay que hacer cambios en el recién creado kerberos.ldif; podemos hacer una edición no interactiva con sed así:

```
sed -i'.pre_sed.BAK' \
    -e s/'dn: cn={0}kerberos'/'dn: cn=kerberos,cn=schema,cn=config'/g \
    -e s/'cn: {0}kerberos'/'cn: kerberos'/g \
    -n -e '/structuralObjectClass: olcSchemaConfig/,$!p' \
    ~/ldif/kerberos.ldif
```

El resultado es que las tres primeras líneas, así como las últimas, tengan el aspecto:

```
view ~/ldif/kerberos.ldif
```

```
dn: cn=kerberos,cn=schema,cn=config
objectClass: olcSchemaConfig
cn: kerberos
...
# Y no deberi'a aparecer al final ninguna de las siguientes:
# structuralObjectClass: olcSchemaConfig
# entryUUID: 8817918a-8dc3-102f-8103-2371e8422500
# creatorsName: cn=config
# createTimestamp: 20101126161112Z
# entryCSN: 20101126161112.176097Z#000000#000#000000
# modifiersName: cn=config
# modifyTimestamp: 20101126161112Z
```

Ya podemos cargar el primer fichero ldif, tal como se indicó:

```
ldapmodify -a -QY EXTERNAL -H ldapi:/// -f ~/ldif/kerberos.ldif
```

Podemos comprobar el resultado con ldapsearch:

```
ldapsearch -LLLQY EXTERNAL -H ldapi:/// -b cn=config cn={4}kerberos | less
```

En el resto de nuestro recorrido por cn=config y, posteriormente, por dc=casafx, dc=dyndns, dc=org, todos los cambios que se planteen forman parte ya del segundo fichero ldif. Se recomienda respetar los espacios en blanco mostrados: no introducir nuevos y eliminar los que hay. Excepto el primer fragmento, los demás suelen tener un espacio en blanco que separa el código LDIF anterior del comentario para el código LDIF subsiguiente etc; otras veces no aparece dicho espacio sino un guión "-" y salto de línea, que indica al procesador LDIF que debe aprovechar el DN sobre el que se produjo la modificación anterior. Consúltese "man ldif" en caso de duda.

Cambios en el nivel de logging; none->stats Efectivamente en cn=config tenemos un nodo que controla la verbosidad de slapd (por defecto logea a syslog). Nos conviene aumentarla durante nuestro despliegue.

Como se comentó, comienza aquí las modificaciones que van en el segundo fichero ldif; por tanto lo abrimos y vamos incorporando ésta y las demás que vayamos decidiendo:

```
vim ~/ldif/ldap1_olc-mod.ldif
```

```
##############################################################
#####                      DB cn=config                  #####
##############################################################


#### Loglevels en:
#  http://www.openldap.org/doc/admin24/slapdconfig.html#Global%20Directives
dn: cn=config
changetype: modify
replace: olcLogLevel
olcLogLevel: stats
```

Mapeos necesarios para las autenticaciones entre principals y nodos En cn=config también existen nodos que le instruyen al subprotocolo SASL sobre cómo autenticar a entidades que usan un sistema de nombres (como los principal Kerberos) a entidades propias de OpenLDAP, los organizationalRole (u otra clase). Tres son los cambios

- Mapear los principal relacionados con los consumidores, concepto éste relativo a la replicación entre servidores LDAP.
- Mapear los principal relacionados con los usuarios finales de nuestra organización.
- Establecer el REALM Kerberos (básicamente, la unidad administrativa en la que el sistema Kerberos agruparía a sus usuarios). Diremos, congruentemente a lo que declararemos al desplegar MIT Kerberos, que será "CASAFX.DYNDNS.ORG".

(Las entradas olcAuthzRegexp: aparecen divididas en 3 y 2 partes delimitadas por \ y cambio de línea. Realmente cada una debe aparecer en una sóla línea, sin \ y cambio de línea:)

```
#### Sasl Auth Mapping
#www.openldap.org/doc/admin24/sasl.html#Mapping%20Authentication%20Identities
#man slapd-config
dn: cn=config
changetype: modify
add: olcAuthzRegexp
#Recordamos que el si'mbolo "\" y el salto de li'nea sobran en olcAuthzRegexp:
olcAuthzRegexp: uid=ldap/([^/\.]+).casafx.dyndns.org,cn=casafx.dyndns.org,\
cn=gssapi,cn=auth \
cn=$1,ou=consumers,dc=casafx,dc=dyndns,dc=org
-
add: olcAuthzRegexp
#Recordamos que el si'mbolo "\" y el salto de li'nea sobran en olcAuthzRegexp:
olcAuthzRegexp: uid=([^,]+),cn=casafx.dyndns.org,cn=gssapi,cn=auth \
uid=$1,ou=users,ou=accounts,dc=casafx,dc=dyndns,dc=org
-
add: olcSaslRealm
olcSaslRealm: CASAFX.DYNDNS.ORG
# La siguiente le permitiri'a a slapd, en su papel de consumidor en el
# sistema de replicacio'n, saber que' llave kerberos exportada en su
# keytab tiene que usar: ldap/<olcSaslHost>@<olcSaslRealm>, pero nosotros
# so'lo tendremos una exportada y por tanto no lo usamos:
#-
#add: olcSaslHost
#olcSaslHost: dklab1.casafx.dyndns.org
```

Soporte para replicación LDAP: carga del módulo, declaración de los ID. El sistema de replicación requiere se configura parcialmente en cn=config, y el resto en la subrama de cn=config que represente al DIT que vaya a ser objeto de replicación:

- cn=config
    - Previamente necesitamos cargar el módulo con la funcionalidad de replicación. Para ello, al nodo cn=module le añadimos un atributo olcModuleLoad de valor "syncprov", synchronization provider.
    - Atributo olcServerID de cn=config, su valor será el identificador unívoco de este slapd entre los demás replicadores en un esquema Multi-Master.
- subrama de cn=config para el DIT objeto de replicación, se verán justo a continuación
    - objeto olcOverlay con valor syncprov
    - atributo olcSyncRepl
    - atributo olcMirrorMode

Nosotros no vamos a replicar el DIT cn=config, por lo que ahora sólo nos interesa la primera parte, cargar el módulo y dar un identificador a este slapd:

```
#### Ldap-replication

# Carga del mo'dulo para replicacio'n (no fue compilado esta'ticamente)
dn: cn=module{0},cn=config
changetype: modify
add: olcModuleLoad
olcModuleLoad: syncprov

# Server ID, que identifica a este slapd uni'vocamente, en Multi-Master
# Replication, como fuente. Son un nu'mero de hasta 3 di'gitos y
# una opcional url, que si aparece permite tener varios atributos
# olcServerID para anunciar a todos los slapd del grupo Multi-Master
# (pero es opcional excepto quiza's cuando replicas tambie'n el
# DIT cn=config). En man slapd-config dice que el FQDN "should be use"
# como url.
dn: cn=config
changetype: modify
replace: olcServerID
olcServerID: 001 ldap://dklab1.casafx.dyndns.org
#olcServerID: 002 ldap://dklab2.casafx.dyndns.org No parece hacer falta
#                                                 si no replico {0}config.
```

olcDatabase={1}hdb,cn=config; olcRootDN, LDAP-replication, Acl, Índices La rama olcDatabase={1}hdb del DIT cn=config, representa la configuración del DIT indexado como {1}, es decir dc=casafx,dc=dyndns,dc=org. Vamos a reconfigurar aquí los siguientes cuatro aspectos:

olcRootDN Usaremos el nombre "root/admin", que no corresponde a un ningún nodo. Al desplegar MIT Kerberos, crearemos un principal para esa entidad, de forma que (gracias a los mapeos anteriores), el poseedor de las credenciales relativas a ese principal pueda autenticarse y conseguir autorización en el árbol como administrador.

El atributo olcRootPW asociado lo conservamos porque sin él no será posible usar el olcRootDN hasta que SASL-GSSAPI esté habilitado, lo cual es un problema porque entre este momento y entonces, existirá la necesidad de usar olcRootDN con autenticación simple (no SASL-GSSAPI). En concreto al usar la herramienta kdb5_ldap_util de MIT Kerberos (que crea precisamente las estructuras de datos en LDAP), parece ser que no es capaz de usar -Y EXTERNAL, y nos será imprescindible usar autenticación simple con olcRootDN y olcRootPW.

Así pues, tras usar kdb5_ldap_util será cuando eliminemos olcRootPW y, a la vez, obliguemos a hacer sasl-gssapi para usar el olcRootDN con el principal "root/admin".

```
#### Rootdn
dn: olcDatabase={1}hdb,cn=config
changetype: modify
replace: olcRootDN
olcRootDN: uid=root/admin,ou=users,ou=accounts,dc=casafx,dc=dyndns,dc=org
#-
#delete: olcRootPW
```

Configuración de replicación LDAP Como se ha comentado, la replicación se configura parcialmente en cn=config (básicamente, los identificadores) y el resto, que nos compete ahora, en la subrama olcDatabase de cn=config para el DIT objeto de replicación. Puesto que nuestro esquema de replicación es Multi-Master, slapd en dklab1 debe ser tanto proveedor como consumidor.

- La configuración como proveedor es, básicamente, declarar esta condición.
- La configuración de consumidor es más compleja, ya que necesitamos especificar cómo conectarnos al proveedor. Es reseñable
  - rid asigna un identificador a este atributo syncrepl del rol de consumidor
  - provider, la URL del proveedor
  - type refreshAndPersist, establece una conexión permanente con el proveedor en lugar de sondear periódicamente.
  - retry no es incompatible con lo dicho anteriormente, pues establece intervalos pero para reintentar una conexión que se ha perdido.
  - searchbase indica a partir de qué nodo en el árbol queremos mantenernos sincronizados.
  - syncdata sólo tiene importancia para compatibilidad hacia atrás, y el resto se relacionan con el proceso de autenticación.

```
#### Ldap Replication Protocol
#    http://tools.ietf.org/html/rfc4533
#Overlay syncprov hace referencia al rol de provider
#Syncrpl          hace referencia al rol de consumer
#Mirrormode       permite las escrituras

#Provider:
#http://www.zytrax.com/books/ldap/ch6/syncprov.html
dn: olcOverlay=syncprov,olcDatabase={1}hdb,cn=config
changetype: add
objectClass: olcOverlayConfig
objectClass: olcSyncProvConfig
olcOverlay: syncprov
olcSpCheckpoint: 100 5
```

```
#Consumer:
#http://www.openldap.org/doc/admin24/slapdconfig.html#syncrepl
dn: olcDatabase={1}hdb,cn=config
changetype: modify
add: olcSyncRepl
olcSyncRepl: rid=001
             provider=ldap://dklab2.casafx.dyndns.org
             type=refreshAndPersist
             retry="5 5 300 +"
             searchbase="dc=casafx,dc=dyndns,dc=org"
             syncdata=default
             bindmethod=sasl
             saslmech=gssapi
-
#Mirror mode:
add: olcMirrorMode
olcMirrorMode: TRUE
```

```
#Limits:
#http://www.openldap.org/doc/admin24/limits.html
#Por defecto hay unos li'mites en el taman~o de datos,
#que nosotros vamos a revertir:
-
add: olcLimits
olcLimits: dn.onelevel="ou=consumers,dc=casafx,dc=dyndns,dc=org"
           size.soft=unlimited
           size.hard=unlimited
           time.soft=unlimited
           time.hard=unlimited
```

ACL Autorizacion en OpenLDAP. Esta sección es crítica para comprender el comportamiento de slapd y el nivel de exposición de nuestra información, pues al fin y al cabo explicita qué y quién puede hacer qué, y qué no.

Lo fundamental es que se evalúan en orden descendiente, y se aplica la primera que case. Respecto a la sintaxis, esta leyenda informal puede ser útil:

```
#### ACL: PRIVILEGIOS DE ROLES LDAP, consiste en declarar sentencias "olcAccess"

access to <what> [ by <who> [<accesslevel>] [<control>] ]+
          |           |      |               |           '-> hace de AND
          |           |      |               '-> stop-> (default) si match, pasa del resto de acl
          |           |      |                   continue-> si no match, prueba sgte who de esta
          |           |      |                   break-> si match, pa'sate ya a la siguiente acl
          |           |      |
          |           |      '->[self]{<level>|<priv>}
          |           |          |      |      '->con memoria: =   es reset lo q hubiese
          |           |          |      |                     +/-x es add/rv, con x w,r,s,c,x
          |           |          |      '-> none,auth,read,search,write,compare...
          |           |          '->trato especial a attr q representan al q accede:
          |           |             ejem quitarte de una lista GroupPosix...
          |           '-> *         ->cualquiera (por cierto, puedes poner varios <who>)
          |           '-> anonymous ->no autentificados
          |           '-> users     ->cualquiera autentificado
          |           '-> dn[.type[,modifier]]=pattern
          |           '-> self      -> accedes a un nodo si el pass de auth esta' ahi'
          |           '-> peername, sockname, sockurl, domain... por do'nde has entrado
          |           '-> tls_ssf=n, ssf=n... nivel encriptacio'n n
          '-> dn[.type]=pattern
          |       base    -> (default) te refieres exactam a lo q case pattern. = exact.
          |       subtree -> el nodo q case pattern y lo q venga debajo
          |       children-> el nodo q venga debajo, sin contar lo q case el pattern
          |       regexp  -> creo q se expande a lista q luego usas con $1... en el who
          '-> attrs=attrslist -> ahora no busca en nombre nodo sino atributos
          |                      a los que aplicar algo (por tanto a todo nodo q los tenga)
          |                      La lista puede tener palabras especiales:
          |                      entry-> por defecto, te refieres al nodo q tenga ese atributo
          |                      children-> tb a sus hijos
          |                      @objectclass->la lista de los attr de esa clase (ej: inetOrgPers
          '-> filter=ldapfilters (ver rfc4515)
          '-> * -> para indicar q te refieres a todo nodo del dit

Tests: man slapacl
```

```
"The access directive (ACL) is brutally complex."
"Perhaps the best way to understand this directive is to skim
 through the detail descriptions, go to the examples, then go
 back and re-read (theory)".
"One or more access to directives may be included in either
 or both of the global section (olcDatabase={-1}frontend en
 cn=config) or a  specific DIT (0, 1, ...)"

http://www.zytrax.com/books/ldap/ch6/#access
```

OpenLDAP viene con algunas predefinidas y, para añadir otras, es válido tanto interponerlas a éstas usando los prefijos "{<ordinal>}" como borrarlas todas y comenzar de nuevo, camino éste que hemos elegido.

Algunas de las ACL controlan ramas que aún no hemos creado pero que ordenaremos que se haga cuando nuestro recorrido salga de este DIT (cn=config) y pase al DIT de nuestra organización (dc=casafx,dc=dyndns,dc=org). Ejemplo serán las ramas a partir de las que cuelgan nuestros datos de MIT Kerberos, o de nuestras cuentas de usuario etc, todas de la forma <cualquiercosa>,dc=casafx,dc=dyndns,dc=org; en numerosas ocasiones, <cualquiercosa>es "ou" porque indica que el nodo usa el objeto organizationalUnit, hecho que lo convierte de alguna forma en el equivalente a un directorio en un sistema de ficheros, es decir una entidad de la que pueden colgar otras.

- Las tres primeras acciones (comentadas como -1, -2, -3) eliminan las ACL preexistentes.

```
#### ACL


# Eliminamos acl preexistentes
# -1
dn: olcDatabase={1}hdb,cn=config
changetype: modify
delete: olcAccess
olcAccess: {2}to *
  by self write
  by dn="cn=admin,dc=casafx,dc=dyndns,dc=org" write
  by * read
-
# -2
delete: olcAccess
olcAccess: {1}to dn.base=""
  by * read
-
# -3
delete: olcAccess
olcAccess: {0}to attrs=userPassword,shadowLastChange
  by self write
  by anonymous auth
  by dn="cn=admin,dc=casafx,dc=dyndns,dc=org" write
  by * none
```

- La comentada como +1: permite el acceso a cualesquiera atributos userPassword y shadowLastChange. De manera que:
  - si la entidad que accede está definida en los nodos que cuelgan inmediatamente (one) de "ou=consumers,dc=casafx,dc=dyndns,dc=org", tienen permiso para leer (lógico, esas entidades representan a los otros slapd replicando los cambios producidos aquí en esos atributos).

- si accedemos como la entidad especial anonymous (sin autenticar), se permite usarlos para efectuar una autenticación.
- cualquier otra entidad no tiene ninún permiso.

```
-
# An~adimos acl
# +1 userPass
add: olcAccess
olcAccess: to attrs=userPassword,shadowLastChange
  by dn.one="ou=consumers,dc=casafx,dc=dyndns,dc=org" read
  by anonymous auth
  by * none
```

- La comentada como +2: esta vez regula el acceso al subárbol (subtree) donde se guardarán los datos de kerberos, es decir, "ou=krb5,dc=casafx,dc=dyndns,dc=org".
  - si la entidad que accede es un nodo que representa al proceso kadmind, es decir al DBMS de MIT Kerberos, se le permite escribir.
  - si la entidad que accede es un nodo que representa al centro de autenticación de MIT Kerberos, se le permite leer y será suficiente.
  - si la entidad que accede es las referidas en la regla anterior para las replicaciones (los consumers), se le permite leer por las mismas razones que entonces.
  - el resto, evidentemente, no deberían tener ningún tipo de acceso bajo ningún motivo.

```
-
# +2 KRBsubtree
add: olcAccess
olcAccess: to dn.subtree="ou=krb5,dc=casafx,dc=dyndns,dc=org"
  by dn="cn=adm-srv,ou=krb5,dc=casafx,dc=dyndns,dc=org" write
  by dn="cn=kdc-srv,ou=krb5,dc=casafx,dc=dyndns,dc=org" read
  by dn.one="ou=consumers,dc=casafx,dc=dyndns,dc=org" read
  by * none
```

- La comentada como +3: se refiere al atributo loginShell que se encuentre definido como uno de los atributos del nodo correspondiente. Representa al shell preferida de un usuario para acceder al SO y, como tal, lo lógico es que sólo ése usuario pueda cambiarla. Por tanto

  - si la entidad que intenta acceder es el nodo cuyo atributo loginShell pretende modificar (self), se le permite.
  - los consumers podrán leerlo para poder replicar los cambios.
  - el resto pueden leer. En concreto, en despliegues donde, como se verá, los sistemas de resolución de meatadatos para cuentas tal como NSS se conectan anónimamente, lo necesitamos así.

```
-
# +3 loginShell
add: olcAccess
olcAccess: to attrs=loginShell
  by self write
  by dn.one="ou=consumers,dc=casafx,dc=dyndns,dc=org" read
  by * read
```

- La comentada como +4: permite a cualquiera poder leer el raiz del árbol LDAP. Bajo ese mecanismo se ofrece la posibilidad de consultar los mecanismos de autenticación soportados. Puesto que a todos se permite leer, implícitamente queda recogida la posibilidad de replicación que todas las reglas anteriores han preservado y ésta no es menos por tanto.

```
-
# +4 dn.base
# e'sto permite poder consultar los mecanismos de auth soportados:
#   ldapsearch -H ldap://dklab1.casafx.dyndns.org -x -b "" \
#              -s base -LLL supportedSASLMechanisms
add: olcAccess
olcAccess: to dn.base=""
  by * read
```

- La comentada como +5 se refiere al subárbol "ou=accounts,dc=casafx,dc=dyndns,dc=org" que, como su propio nombre indica, es la zona del árbol donde almacenaremos la información de las cuentas de usuario de nuestra organización. Todo el mundo puede leer esa parte del árbol, ya que consideramos que la información sensible son las credenciales Kerberos, y éstas se encuentran en otro subárbol, "ou=krb5,dc=casafx,dc=dyndns,dc= que controlaba la ACL "+2".

```
-
# +5 accounts, sus atributos no son informacio'n sensible
#    (el pass esta' en kerberos) asi' que debieran poder ser
#    consultadas ano'nimamente:
# ldapsearch -x -b ou=accounts,dc=casafx,dc=dyndns,dc=org
# E'sto facilitara' el despliegue, pero si en un momento futuro
# la poli'tica cambia, slapd estara' (al final de este
# capi'tulo) kerberizado y se podra' pedir autenticacio'n robustamente.
add: olcAccess
olcAccess: to dn.subtree="ou=accounts,dc=casafx,dc=dyndns,dc=org"
  by * read
```

- La última regla, comentada como +6, efectivamente decide qué se hace cuando el resto de reglas no se han aplicado, es decir cuando intentamos acceder a un nodo o atributo que no requiere de un comportamiento excepcional. Este comportamiento por defecto será:
  - los usuarios autenticados puden leer.
  - el resto no tiene permiso alguno.

```
-
# +6 resto
add: olcAccess
olcAccess: to *
  by users read
  by * none
```

- Anotación: ya explicamos cómo los flags "-QY EXTERNAL -H ldapi:///" nos permitían acceder con uid=0 y gid=0 a través de la interfaz IPC (Inter Process Communication, en este caso era un "named socket"). Evidentemente, una ACL debe permitir ésto, concretamente esa ACL es:

```
olcAccess: {0}to * by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external
```

```
,cn=auth manage by * break
```

Sin embargo cuando eliminamos, dijimos, las ACL predefinidas no encontrábamos a ésta. La razón es que está definida, pero en otro lugar. Existe una rama del árbol cn=config, "olcDatabase: {-1}frontend,cn=config", que controla la configuración por defecto de los DIT, y donde están definidas ACL adicionales que se intentan aplicar primero. Puede inspeccionarse con

```
slapcat -b cn=config | grep -B1 -A1 olcAccess.*gid
```

Índices Los índices[2] son unas estructuras que facilitan las búsquedas relativas a ciertos atributos. Básicamente, si conocemos de antemano que un atributo va a ser objeto de numerosas búsquedas o filtros (caso de "uid" por ejemplo) o si diretamente slapd protesta en syslog sobre la falta de indexación de algún atributo, le permitimos que lo indexe porque el desempeño puede verse muy afectado en otro caso[3].

[2] http://www.openldap.org/doc/admin24/tuning.html#Indexes
[3] http://www.openldap.org/faq/data/cache/42.html

```
#### I'ndices
# 1
dn: olcDatabase={1}hdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: uid eq
-
# 2
add: olcDbIndex
olcDbIndex: cn eq
-
# 3
add: olcDbIndex
olcDbIndex: ou eq
-
# 4
add: olcDbIndex
olcDbIndex: dc eq
-
# 5
add: olcDbIndex
olcDbIndex: uidNumber eq
-
# 6
add: olcDbIndex
olcDbIndex: gidNumber eq
```

```
-
# 7
add: olcDbIndex
olcDbIndex: memberUid eq
-
# 8
add: olcDbIndex
olcDbIndex: uniqueMember eq
-
# 9
add: olcDbIndex
olcDbIndex: krbPrincipalName eq,pres,sub
-
# 10
add: olcDbIndex
olcDbIndex: krbPwdPolicyReference eq

# 11 (relacionado con replicacio'n)
dn: olcDatabase={1}hdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: entryUUID eq
-
# 12 (relacionado con replicacio'n)
add: olcDbIndex
olcDbIndex: entryCSN eq
```

dc=casafx, dc=dyndns, dc=org

Seguimos completando el segundo fichero ldif, ahora con los cambios que requiere el DIT de nuestra organización, dc=casafx,dc=dyndns,dc=org.

```
#############################################################
#####       DB dc=casafx,dc=dyndns,dc=org                #####
#############################################################

#En este DIT's las modificaciones consisten en an~adir subdirectorios
#con las entidades necesarias para almacenar las db de kadmind,
#identificar los organizationalRole al replicar. Tambie'n podemos crear
#ya los subdirectorios para las cuentas de sistema.

#######
# Previo:
# por sencillez, nos aseguramos que est'a definida la ACL que nos
# permite cargar este ldif en dc=casafx,dc=dyndns,dc=org usando
# IPC (unix socket en este caso) y la condicio'n uid=gid=0, gracias
# a las opciones "-QY EXTERNAL -H ldapi:///" de ldapmodify).
# Sera' eliminada al terminar puesto que, una vez que kerbericemos la
# autenticacio'n en slapd, usaremos la cuenta root/admin declarada
# para el rootdn.

dn: olcDatabase={1}hdb,cn=config
changetype: modify
add: olcAccess
olcAccess: {0}to *
    by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
    manage by * break
#
#######
```

Rama para MIT Kerberos Definimos el nodo a partir del cual colgarán los recursos que necesite MIT Kerberos. Además, creamos dos organizationalRole, uno para el proceso gestor de los datos (kadmind) y otro para el responsable de las autenticacioens (krb5kdc):

```
#### MIT Kerberos Storage
dn: ou=krb5,dc=casafx,dc=dyndns,dc=org
changetype: add
ou: krb5
objectClass: organizationalUnit

dn: cn=kdc-srv,ou=krb5,dc=casafx,dc=dyndns,dc=org
changetype: add
cn: kdc-srv
objectClass: simpleSecurityObject
objectClass: organizationalRole
description: Default bind DN for the Kerberos KDC server
userPassword: kdcpass

dn: cn=adm-srv,ou=krb5,dc=casafx,dc=dyndns,dc=org
changetype: add
cn: adm-srv
objectClass: simpleSecurityObject
objectClass: organizationalRole
description: Default bind DN for the Kerberos Administration server
userPassword: kadmindpass

# NOTA: los servidores krb5kdc y kadmind accedera'n por IPC
#       y que por e'so no es problema que esos atributos userPassword
#       este'n en claro.
#       Evidentemente no se puede usar autenticacio'n SASL-GSSAPI para
#       ellos.
```

Roles para LDAP replication Creamos un nodo del que cuelgan las entidades que usarán los consumidores para autenticarse y ser autenticados. Puesto que sólo habrá dos slapd en nuestro testbed, definimos dos entidades.

```
#### Roles LDAP Replication

dn: ou=consumers,dc=casafx,dc=dyndns,dc=org
changetype: add
ou: consumers
objectClass: organizationalUnit

dn: cn=dklab1,ou=consumers,dc=casafx,dc=dyndns,dc=org
changetype: add
cn: dklab1
objectClass: simpleSecurityObject
objectClass: organizationalRole
description: LDAP server2 replicator
userPassword: {CRYPT}*

dn: cn=dklab2,ou=consumers,dc=casafx,dc=dyndns,dc=org
changetype: add
cn: dklab2
objectClass: simpleSecurityObject
objectClass: organizationalRole
description: LDAP server2 replicator
userPassword: {CRYPT}*
```

Cuentas de usuarios finales Creamos un nodo del que cuelgan

```
#### Cuentas: Usuarios y Grupos
dn: ou=accounts,dc=casafx,dc=dyndns,dc=org
changetype: add
objectClass: organizationalUnit
ou: accounts

dn: ou=users,ou=accounts,dc=casafx,dc=dyndns,dc=org
changetype: add
objectClass: organizationalUnit
ou: users

dn: ou=groups,ou=accounts,dc=casafx,dc=dyndns,dc=org
changetype: add
objectClass: organizationalUnit
ou: groups
```

```
#######
# Post:
# deshabilitamos la ACL que permite modificar el DIT usando IPC y uid=gid=0
dn: olcDatabase={1}hdb,cn=config
changetype: modify
delete: olcAccess
olcAccess: {0}to *
     by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
     manage by * break
#
#######
#
###############################################################################
#                                  FIN
###############################################################################
```

Y ya podemos cargar este segundo ldif; tal como se indicó:

```
ldapmodify -QY EXTERNAL -H ldapi:/// \
           -f ~/ldif/ldap1_olc-mod.ldif \
           -c -S ~/ldif/ldiferrors-‘date +%Y%m%d.%H%M_%S‘.ldif
```

Si todo ha ido bien, el fichero ~/ldif/ldiferrors-* estará vacío y podremos borrarlo, en otro caso incluirá qué y por qué no se pudo cargar.

Podemos hacer un volcado del estado actual de nuestro árbol LDAP. La herramienta que permite hacer volcados offline es slapcat, utiliza directamente los ficheros BerkeleyDB (en /var/lib/ldap/) luego necesita ejecutarse con una cuenta privilegiada del SO (root) o la que usa slapd (openldap). Los volcados pueden cargarse con slapadd (utilidad offline, slapd debería no estar cargado mientras se usara slapadd).

```
slapcat -b cn=config > ~/ldif/ldap1_cn_config.ldif
slapcat -b dc=casafx,dc=dyndns,dc=org > ~/ldif/ldap1_dc_casafx.ldif
```

1.1.4. Instalación de MIT Kerberos

```
apt-get install krb5-admin-server krb5-kdc krb5-kdc-ldap \
                krb5-config krb5-user
```

- krb5-kdc y krb5-kdc-ldap, provee el KDC y el módulo para operar sobre LDAP
- krb5-admin-server, provee el gestor de la base de datos con backend LDAP entre otros.
- krb5-config, provee el fichero de configuración para clientes /etc/krb5.conf
- krb5-user, provee las utilidades de cliente (kinit etc)

```
-"Default Kerberos version 5 realm":
CASAFX.DYNDNS.ORG
-"Add locations of default Kerberos servers to /etc/krb5.conf?"
No
-"Create the Kerberos KDC configuration automatically?"
Yes
-"Run the Kerberos V5 administration daemon (kadmind)?"
Yes
```

Puede que aparezcan los siguientes errores, que no tienen importancia en este momento:

```
...
krb5kdc: cannot initialize realm CASAFX.DYNDNS.ORG - see log file for details
kadmind: No such file or directory while initializing, aborting
...
```

Las dependencias entre MIT Kerberos y OpenLDAP que se comentaron nos obligan a reflexionar sobre los scripts de inicio. Los procesos krb5kdc/kadmind necesitan que slapd esté cargado con anterioridad (es precisamente por ello que nosotros hemos desplegado primero a OpenLDAP, y efectivamente tenemos a slapd funcionando). Pero cuando el ordenador se reinicie, nada nos asegura que se cumpla esta situación porque los DD (Debian Developer) no han explicitado esa dependencia en los scripts de inicio, de lo cual debemos ocuparnos nosotros ahora.

En la versión 7 de Debian, se usa un sistema Init System V basado en dependencias. Necesitamos modificar las cabeceras LSB Init de los scripts que inician krb5kdc y kadmind para declarar su dependencia con slapd; entonces ejecutamos el programa insserv para que dé constancia de ésto en el orden de ejecución de los scripts de inicio.

```
vim /etc/init.d/krb5-kdc
```

Para no perturbar posiblemente el formato, diremos ahora que el cambio consiste en añadir slapd tal como se indica:

```
### BEGIN INIT INFO
...
# Required-Start:       $local_fs $remote_fs $network $syslog slapd
# Required-Stop:        $local_fs $remote_fs $network $syslog slapd
...
### END INIT INFO
```

Y en el script de kadmind, procedemos de idéntica forma:

```
vim /etc/init.d/krb5-admin-server
```

```
### BEGIN INIT INFO
...
# Required-Start:       $local_fs $remote_fs $network $syslog slapd
# Required-Stop:        $local_fs $remote_fs $network $syslog slapd
...
### END INIT INFO
```

Ejecutamos insserv:

```
insserv /etc/init.d/krb5-kdc
insserv /etc/init.d/krb5-admin-server


# Nota:
# Entendemos que los cambios se realizan en los ficheros indicados,
# pero es cierto que para krb5-kdc existe
# /etc/insserv/overrides/krb5-kdc
# http://wiki.debian.org/LSBInitScripts/DependencyBasedBoot
```

Más adelante será de extrema importancia tener certeza de cuál es el FQDN[4] que todo host que vaya a ofrecer servicios kerberizados cree tener asignado, así como que su resolución directa e inversa sean una biyección. Efectivamente la política es estricta en cuanto a los nombres, si hay alguna discripancia entre los nombres usados en los tickets y aquellos que los hosts que ofrecen los servicios consideran que tienen, las autenticaciones fallarán.

En lo que respecta a dklab1 y dklab2, la configuración del sistema DNS refleja esa biyectividad, así mismo ocurre que cada máquina puede construir (a través de /etc/hostname y la línea search de /etc/resolv.conf) ese mismo nombre que le asigna el resolvedor; damos un ejemplo de cómo comprobarlo en un equipo similar:

```
# La maquina entiende que su FQDN es la salida de:
FQDN=‘hostname --fqdn‘
# A su vez, el resolvedor entiende que a ese nombre le corresponde la IP:
RDIR=‘dig @10.168.1.1 ${FQDN} A +short‘
# Y de forma inversa, el resolvedor entiende que a esa IP
# le corresponde el nombre:
RINV=‘dig -x @10.168.1.1 $(echo $RDIR | tac -s \. |
      tr ’\n’ ’.’)IN-ADDR.ARPA. any +short | tr ’\n’ ’ ’‘

# Tenemos, por tanto:
printf "${FQDN}\t -> ${RDIR},\n ${RDIR}\t\t\t -> ${RINV}\n"
```

```
dklab1.casafx.dyndns.org    -> 10.168.1.1
10.168.1.1                  -> dklab1.casafx.dyndns.org.  (y nada ma’s)
```

1.1.5. Modificaciones sobre clientes KERBEROS

La librería libkrb5 con que se enlazan las aplicaciones que utilizan a MIT Kerberos, busca su configuración en /etc/krb5.conf. Puesto que dklab1 necesitará actuar también como cliente, vamos a adaptar a nuestro despliegue el fichero de ejemplo que dejó el paquete krb5-config:

- La función primordial del /etc/krb5.conf es anunciar la localización de los servicios (KDBM y KDC) y realm (reino, el nombre de la unidad administrativa), propio y

[4]Fully Qualified Domain Name, básicamente es el nombre que devuelve el resolvedor para un host.

también ajenos. Nosotros proveemos el descubrimiento de todos a través de registros SRV y TXT en DNS, sin embargo. Si declaramos no obstante el realm como el realm por defecto, podremos ahorrarnos su escritura en la mayoría de las ocasiones, hecho deseable.

- Se define cómo está configurado el backend de almacenamiento de datos en LDAP: qué entidades pueden acceder, cómo se autentican (dónde pueden encontrar su credencial), cuál es la rama dedicada a MIT Kerberos, etc.
- Cómo reportar mensajes de depuración (logging), en este caso no se usa syslog sino ficheros dedicados; a continuación crearemos una entrada en el rotador de logs del sistema para gestionarlos.

```
vim /etc/krb5.conf
```

```
[libdefaults]
        default_realm = CASAFX.DYNDNS.ORG
        ...
        forwardable = true
        proxiable = true
        ####fx:
        #si versio'n >=1.7 y usamos telnet u OpenAFS, explici'tese:
        allow_weak_crypto = true
        # Si algo falla, podemos activar debug:
        #debug = true
        ####endfx
...
[realms]
####fx
        CASAFX.DYNDNS.ORG = {
        # SRV _kerberos._ucp.casafx.dyndns.org. es para los kdc:
                #kdc = krb1.casafx.dyndns.org
                #kdc = krb2.casafx.dyndns.org

        # SRV kerberos-adm._tcp.casafx.dyndns.org, no esta'
        # claro que kadmin lo use au'n, por tanto los hacemos
        # expli'citos localmente:
            admin_server = dklab1.casafx.dyndns.org
            admin_server = dklab2.casafx.dyndns.org

            database_module = openldap_ldapconf
        }
####endfx
...
```

```
...
[domain_realm]
####fx:
        # TXT _kerberos.casafx.dyndns.org. debiera valer para
        # mapear ambos, asi' pues los descomentamos:
        #.casafx.dyndns.org = CASAFX.DYNDNS.ORG
        #casafx.dyndns.org = CASAFX.DYNDNS.ORG
####endfx
...
[login]
...
####fx:
# Atencio'n: no confu'ndase la seccio'n login (de la que no
# especificamos nada pero que precede) con esta seccio'n logging
# que creamos ahora:
[logging]
         kdc = FILE:/var/log/krb5/kdc.log
         admin_server = FILE:/var/log/krb5/kadmin.log
         default = FILE:/var/log/krb5/klib.log
         # Quiza's pueda ser interesante conocer que puede usar a syslog:
         # default = SYSLOG:INFO:LOCAL6
```

```
[dbdefaults]
        ldap_kerberos_container_dn = ou=krb5,dc=casafx,dc=dyndns,dc=org
[dbmodules]
        openldap_ldapconf = {
        db_library = kldap
        ldap_kdc_dn = cn=kdc-srv,ou=krb5,dc=casafx,dc=dyndns,dc=org
        ldap_kadmind_dn = cn=adm-srv,ou=krb5,dc=casafx,dc=dyndns,dc=org
        ldap_service_password_file = /etc/krb5kdc/service.keyfile
        ldap_conns_per_server = 5
        }
####endfx
```

Rotación de los logs declarados:

```
mkdir /var/log/krb5


vim /etc/logrotate.d/krb5
```

```
/var/log/krb5/kadmin.log /var/log/krb5/kdc.log /var/log/krb5/klib.log {
        daily
        missingok
        rotate 7
        compress
        delaycompress
        notifempty
}
```

1.1.6. Modificaciones sobre los servidores de MIT Kerberos

- Anotación: kdb5_ldap_util no es capaz de usar -Y EXTERNAL. Ésto, que se comentó anteriormente, hacía que necesitase autenticación simple a alguna entidad como la del olcRootDN del DIT que usa MIT Kerberos. Crear las estructuras kerberos en ldap será la última acción que necesite a olcRootDN para autenticación simple, por tanto eliminaremos a olcRootPW justo después para obligar a usar sólo autenticación SASL-GSSAPI cuando pretendamos ser autorizados como olcRootDN.

Creación del REALM

Utilizamos la utilidad kdb5_ldap_util para crear la rama del DIT dc=casafx,dc=dyndns,dc=org donde se almacenará la base de datos de MIT Kerberos para el reino de autenticación CASAFX.DYNDNS.ORG. La opción "-s" le indica que almacene en un fichero la clave maestra criptográfica del contenido de la base de datos ("Kerberos master key stash file") para ser leída por el software de administración; su localización por defecto es /etc/krb5kdc/stash.

```
kdb5_ldap_util -D uid=root/admin,ou=users,ou=accounts,dc=casafx,dc=dyndns,dc=org\
               -H ldap://dklab1.casafx.dyndns.org \
               create -r CASAFX.DYNDNS.ORG -s
```

```
Password for "uid=root/admin,ou=users,ou=accounts,dc=casafx,dc=dyndns,dc=org":
ldapadmin
Enter KDC database master key: kdcmasterkey
```

- La ausencia del flag -subtrees (véase "man kdb5_ldap_util") merece ser destacada. En nuestro caso hemos optado por tener los principals y demás información de MIT Kerberos en una rama creada a tal efecto, con la ventaja de poder ocultarla fácilmente a todos excepto los servicios de MIT Kerberos. Sin embargo no es la única opción, parece ser posible distribuir los principals entre los nodos que representan las identidades de los usuarios utilizando el flag -subtrees, de hecho es el camino usado en la guía oficial [5].

[5] http://web.mit.edu/kerberos/krb5-1.8/krb5-1.8.3/doc/krb5-admin.html#Configuring%20Kerberos%20with%20OpenLDAP%20back-end

Creación del fichero con credenciales de administración LDAP

Efectivamente, en el caso particular (pues no ocurrirá así en otros) del software de administración de la base de datos de MIT Kerberos, su autenticación y autorización en LDAP será simple a través de una contraseña, y conficencial al utilizar un socket local como método de comunicación interproceso. Para que dicha contraseña pueda usarse la almacenaremos en /etc/krb5kdc/service.keyfile con los permisos adecuados (legible por root sólo). Evidentemente, esas contraseñas y entidades las declaramos nosotros en LDAP cuando cargamos el segundo LDIF de modificaciones, por tanto ahora debemos tener cuidado de volver a usar esos valores:

```
kdb5_ldap_util -D uid=root/admin,ou=users,ou=accounts,dc=casafx,dc=dyndns,dc=org\
               stashsrvpw -f /etc/krb5kdc/service.keyfile \
               cn=kdc-srv,ou=krb5,dc=casafx,dc=dyndns,dc=org
```

```
Password for "uid=root/admin,ou=users,ou=accounts,dc=casafx,dc=dyndns,dc=org":
ldapadmin
Password for "cn=kdc-srv,ou=krb5,dc=casafx,dc=dyndns,dc=org":
kdcpass
#... kdcpass es lo que se asigno' al atributo userPassword para cn=kdc-srv
```

```
kdb5_ldap_util -D uid=root/admin,ou=users,ou=accounts,dc=casafx,dc=dyndns,dc=org\
               stashsrvpw -f /etc/krb5kdc/service.keyfile \
               cn=adm-srv,ou=krb5,dc=casafx,dc=dyndns,dc=org
```

```
Password for "uid=root/admin,ou=users,ou=accounts,dc=casafx,dc=dyndns,dc=org":
ldapadmin
Password for "cn=adm-srv,ou=krb5,dc=casafx,dc=dyndns,dc=org":
kadmindpass
#... kadmindpass es lo que se asigno' al atributo userPassword para cn=adm-srv
```

```
ls -l /etc/krb5kdc/service.keyfile
```

Ya podemos borrar el atributo olcRootPW; como sabemos se encuentra en la rama del DIT cn=config, que almacena la configuración del DIT dc=casafx,dc=dyndns,dc=org (indexado como {1}):

```
cat <<EOF | ldapmodify -a -QY EXTERNAL -H ldapi:///
dn: olcDatabase={1}hdb,cn=config
changetype: modify
delete: olcRootPW
EOF
# Podri'amos haber utilizado alternativamente a ldapdelete.
```

ACL de kadmind

El KDBM, kadmind, permite como cualquier gestor de base de datos la configuración de reglas de control de acceso. Efectivamente, las peticiones sobre adiciones o modificaciones sobre el material criptográfico almacenado en LDAP se realizan sobre el gestor kadmind, no directamente contactando con slapd, y es por tanto kadmind el responsable de decidir si la entidad que las pide está autorizada y, si es así, de llevarlas a cabo.

```
vim /etc/krb5kdc/kadm5.acl
```

```
####fx:
*/admin *
####endfx
```

Véase "man kadmind", sección ACL FILE SYNTAX. La ACL anterior puede interpretarse como que todo principal, cuyo /<instance>sea admin e independientemente de su <primary>/ tiene permisos para cualquier tipo de manipulación. Evidentemente, en la regla anterior casa root/admin, que vamos a crear a continuación como nuestro principal de administrador.

- Recordatorio: gracias a la flexibilidad de las ACL y otros mecanismos, existía un consenso sobre cómo dar significado a las distintas formas que puede adoptar un principal (en general, <primary>/<instance>@<realm>), recordamos:
  - <user>@<realm>es la forma de una entrada de usuario normal.
  - <user>/admin@<realm>de administrador (el /instance daría el rol).
  - <host>/<hostname>@<realm>o <servicio>/<hostname>@<realm>para servicios.

Configuración de los subservicios KDC/AC/KDBM: /etc/krb5kdc/kdc.conf

En este fichero se configura el AC/KDC específicamente. Algunas de las variables son también leídas y entendidas por el KDBM, kadmind, luego a pesar del nombre, kadmind usa también este fichero y no tiene uno de configuración específico. En general aquí se indica la localización de otros ficheros, los algoritmos de encriptación que se emplearán, períodos de validez que se acaban aceptando en este REALM. Véase "man kdc.conf".

```
vim /etc/krb5kdc/kdc.conf
```

```
[kdcdefaults]
    ####fx:
    # Si algo falla, podemos activar debug:
    #debug = true
    ####endfx
...
[realms]
   CASAFX.DYNDNS.ORG = {
   ...
   ####fx:
      max_life = 1d 0h 0m 0s
      max_renewable_life = 90d 0h 0m 0s
   #-max_life = 10h 0m 0s
   #-max_renewable_life = 7d 0h 0m 0s
   ####endfx
   ...
```

Creación del principal root/admin; modificación de la caducidad de los tickets TGT

Utilizamos la utilidad kadmin.local, que es la versión de kadmin que:

- opera sin necesidad de autenticación KERBEROS, ya que no se conectará por TCP a kadmind (el componente, recordamos, que escucha en puerto TCP 749 para permitir

la administración remota de la base de datos de MIT Kerberos, pero que aún no hemos cargado en memoria).

- en su lugar utilizará la interfaz local de slapd (socket en /run/slapd/ldapi) y que por tanto sigue el modelo de autorización POSIX, donde la cuenta local de root tendrá permisos suficientes para establecer contacto. La comunicación subsiguiente estará autorizada según las ACL olcAccess definidas (en olcDatabase={1}hdb,cn=config) para la rama ou=krb5 del DIT dc=casafx,dc=dyndns,dc=org: 'to dn.subtree="ou=krb5,dc=cas by dn="cn=adm -srv,ou=krb5,dc=casafx,dc=dyndns,dc=org'. Esa identidad autorizada, cn=adm-srv, corresponde a la identidad y password declarados en /etc/krb5.conf y /etc/krb5kdc/service.keyfile respectivamente, ambos creados por nosotros hace unos instantes. Por tanto kadmin.local puede descubrir esa identidad y password leyendo dichos ficheros y autorizar su comunicación LDAP, la cual será confidencial en virtud de la localidad del socket.

Definitivamente kadmin.local siempre opera en local[6] y sin necesidades de autenticación y autorización fuera de nuestro alcance, por tanto es la única interfaz de administración que podemos utilizar en este momento. Más adelante podremos utilizar el DBMS kadmind y su cliente kadmin para administración remota con autenticación KERBEROS.

Previamente a crear el principal (add_princ), crearemos 4 políticas de contraseñas. Ésto nos permitirá especificar una vez qué complejidad debe tener una contraseña según esté destinada a un principal de administrador, de usuario, de servicio o de host. La complejidad se mide en el número mínimo de caracteres y el número mínimo de clases de caracteres (letras/números/símbolos...).

Después creamos el principal, su nombre completo es root/admin@CASAFX.DYNDNS.ORG, pero puesto que se configuró ese realm como realm por defecto, podríamos suprimir la arroba y el realm. Nos pedirá una contraseña, que debemos recordar.

El password asignado al principal root/admin será: r00tprincipa!
[7]

[6] De hecho esta premisa se da cualquiera que sea el backend de almacenamiento, por ejemplo es conocido que si la base de datos estuviese en ficheros DBD en lugar de tras LDAP, los abriría directamente, etc.

[7] Donde 00 corresponde al símbolo del numero cero dos veces.

```
kadmin.local
 > listprincs
 > add_policy -minlength 8 -minclasses 3 admin
 > add_policy -minlength 8 -minclasses 4 host
 > add_policy -minlength 8 -minclasses 4 service
 > add_policy -minlength 8 -minclasses 2 user
 > addprinc -policy admin root/admin
Enter password for principal "root/admin@CASAFX.DYNDNS.ORG":
r00tprincipa!
 > getprinc krbtgt/CASAFX.DYNDNS.ORG@CASAFX.DYNDNS.ORG
#  Adicionalmente, puedo usar "1 day" o "25 hours" u otros como
#  peri'odo de validez de los tickets (en lugar de las 10h por defecto).
#  Debiera casar lo configurado aqu'i con lo configurado anteriormente
#  en kdc.conf.
 > modprinc -maxlife "1 day" -maxrenewlife "90 day" \
           krbtgt/CASAFX.DYNDNS.ORG@CASAFX.DYNDNS.ORG
 > quit
```

Así pues, tenemos dos cuentas de administrador:

- Cuenta local POSIX "root", de contraseña "root".
- Cuenta centralizada "root/admin", de contraseña "r00tprincipa!". En principio es administrador para el sistema MIT Kerberos, pero conforme vayamos configurando autenticación KERBEROS y mapeando ese nombre a la cuenta de administrador de otros servicios, se convertirá en la cuenta centralizada de administrador en general. Así estamos ya cerca de tenerlo hecho con OpenLDAP (olcRootDN es root/admin). Y así lo haremos con OpenAFS.

Reinicio de MIT Kerberos

```
invoke-rc.d krb5-admin-server start
invoke-rc.d krb5-kdc start


nmap -sU -sT -p U:88,464,T:464,749 localhost
```

```
...
PORT    STATE         SERVICE
464/tcp open          kpasswd5
749/tcp open          kerberos-adm
88/udp  open|filtered kerberos-sec
464/udp open|filtered kpasswd5
```

1.1.7. Kerberización de OpenLDAP

Al desplegar OpenLDAP, se hicieron las modificaciones pertinentes en el DIT:

- Ya comentamos que la autenticación simple a través de contraseña es algo específico de las comunicaciones LDAP a través de un socket local que realiza el software de administración de la base de datos MIT Kerberos (kadmind, kadmin.local). El resto de las comunicaciones LDAP utilizarán KERBEROS como protocolo de autenticación, presentando un ticket de servicio asociado a un principal. Así, se crearon mapeos entre el sistema de nombres que constituyen la identidad que se asume en un sistema KERBEROS (los principal) y el sistema de nombres que constituyen la identidad que se asume en un sistema LDAP (los DN de nodos "organizationalRole" etc).
- Establecimos el REALM KERBEROS por defecto.
- Aunque no creamos ningún nodo en la subrama ou=users para cuentas de usuario final, ésto no nos impedirá usar inmediatamente la autenticación SASL-GSSAPI porque sí dimos valor al atributo olcRootDN de forma que pueda ser mapeado al principal root/admin que ya hemos registrado.

Pero aún quedan algunas modificaciones que afectan a slapd para que pueda utilizar autenticación KERBEROS:

- Debemos crear un principal de servicio específico para LDAP, exportarlo al sistema de ficheros en un fichero keytab e informar de ello a slapd para que lo lea al arranque. Así, dotamos a slapd de capacidad para autenticarse con el tercero de confianza,

el KDC KERBEROS (proceso krb5kdc). Los clientes LDAP también necesitarán autenticarse ante el KDC (para recibir transparentemente sus tickets de servicio), pero por otro camino ya que, al ser clientes usados interactivamente, su ticket garantía de otros tickets (el TGT) lo consiguen autenticándose primero contra el AS KERBEROS presentando su contraseña. Por tanto los clientes no necesitan tener exportado su principal, sino que necesitan conseguir el TGT interactivamente. Así:

- En el caso de los clientes queda declarar que ya pueden usar SASL-GSSAPI y su REALM por defecto (más tarde, evidentemente, registraremos también sus principal).
- Kerberizar slapd afecta, minúsculamente, a las replicaciones en tanto que obligan a slapd a actuar como cliente KERBEROS, adicionalmente al acondicionamiento que hemos explicado para su rol de servicio KERBEROS.

Creación del principal para OpenLDAP; exportación al keytab

```
mkdir -p /etc/keytab.d


kadmin.local -p root/admin
```

```
 >addprinc -policy service -randkey ldap/dklab1.casafx.dyndns.org
 >ktadd -k /etc/keytab.d/openldap.keytab \
        -norandkey ldap/dklab1.casafx.dyndns.org
 >quit
```

```
klist -ke /etc/keytab.d/openldap.keytab


chown openldap:root /etc/keytab.d/openldap.keytab
chmod go-rw /etc/keytab.d/openldap.keytab
su openldap -m -c 'klist -t -K -k /etc/keytab.d/openldap.keytab'
```

Anunciamos a slapd la localización de su keytab. Para ello exportamos la variable de entorno KRB5_KTNAME en el script de inicio (en concreto un fichero que es incluído en aquel); slapd, o más concretamente las librerías contra las que se enlaza, se encargarán de buscar esa variable cuando la necesite.

```
vim /etc/default/slapd
```

```
...
####fx:
export KRB5_KTNAME="FILE:/etc/keytab.d/openldap.keytab"
#LDAPGSSAPI_ENCRYPT=yes # Es el default
####endfx
```

Es conveniente reiniciar el servicio si se ha añadido software nuevo (los módulos para SASL-GSSAPI):

```
invoke-rc.d slapd restart #nuevo sw an~adido al instalar gssapi modules
```

Configuración de libldap para uso de SASL-GSSAPI

Los clientes ldap utilizan la librería libldap proveída por el proyecto OpenLDAP. El fichero /etc/ldap/ldap.conf controla por tanto el comportamiento de todos los clientes ldap y es donde declararemos que se utilice SASL-GSSAPI como mecanismo de autenticación. No obstante y según "man ldap.conf", podemos cambiar el comportamiento de un cliente en una situación concreta a través de las variables de entorno LDAPCONF y otras.

```
vim /etc/ldap/ldap.conf
```

```
...
####fx:
SASL_MECH GSSAPI
SASL_REALM CASAFX.DYNDNS.ORG
#SASL_ENCRYPT yes
####endfx
```

Adaptación de slapd para utilizar SASL-GSSAPI como consumidor; kstart

El proceso de replicación LDAP implica que un slapd en su rol de consumidor, se autentica como *cliente* KERBEROS en otro slapd. Si la autenticación se basa en KER-

BEROS, necesita conseguir de forma interactiva un TGT y con éste un ticket de servicio que presentar al slapd proveedor.

Para evitar el paso interactivo tanto al inicio como todas las veces que haya que renovar el TGT (debido a su periódo de validez de 1 día, tal como se estableció), utilizamos el software Kstart. Su programa k5start puede ser instruído para conseguir un TGT a partir de un keytab, y renovarlo puntualmente.

Para que k5start se ejecute, a su vez, al inicio y esté supervisado (se vuelva a levantar si por alguna razón falla), utilizamos las capacidades de supervisor de init, creando una nueva entrada en su fichero de configuración inittab:

```
apt-get install kstart


vim /etc/inittab
```

(La línea KSL:... aparece dividida en 2 partes delimitadas por \ y cambio de línea. Realmente debe aparecer en una sóla línea, sin \ y cambio de línea:)

```
...
####fx:
#Nota: init no nos permite poner el nombre del principal porque excederi'a
#el ma'ximo de caracteres permitido por li'nea, entonces usamos -U para que
#k5start coja el primer principal exportado en /etc/keytab.d/openldap.keytab
#Debe tenerse en cuenta este detalle ante cualquier modificacio'n futura
#en ese keytab: k5start espera encontrarse el suyo en primer lugar.
KSL:2345:respawn:/usr/bin/k5start -U -f /etc/keytab.d/openldap.keytab \
-K 10 -l 24h -k /tmp/krb5cc_`id -u openldap` -o openldap
####endfx
```

- -f, indica el keytab donde está la llave a partir de la cual generar un TGT no interactivamente.
- -U, usa el primer principal del fichero anterior, en caso de haber varios exportados ahí.
- -K 10, indica que se relance cada 10 minutos, comprueba cuánto falta para que el TGT expire y lo renueva en su caso.
- -l 24h, pide un periódo de validez de 24 horas para el TGT

- -k indica la localización, en el sistema de ficheros, de la caché de credenciales Kerberos, donde se almacenará el TGT. El nombre dado es también (véase FILES en "man klist") donde por defecto lo buscará el proceso que lo necesite. Evidentemente sólo el proceso que lo necesite debiera poder leerlo, en este caso slapd ejecutándose bajo la cuenta local "openldap", por ello añadimos:
- -o, cambia la ACL POSIX de la cache de credenciales kerberos para que el propietario sea "openldap"; sólo el propietario puede leer y escribir el archivo.

Hacemos a init (primer proceso del sistema) releer su configuración mandándole la señal HUP:

```
kill -HUP 1 # Para que init relea el /etc/inittab


ls -l /tmp/krb5cc_‘id -u openldap‘

su openldap -m -c klist


ps aux | grep k5start
```

1.1.8. Tests principal root/admin

- Vamos a conseguir un TGT para la entidad root/admin@CASAFX.DYNDNS.ORG gracias a la untilidad kinit.
- Despúes comprobaremos que dicho ticket se ha conseguido consultando la caché de credenciales kerberos /etc/krb5cc_<uid>utilizando la utilidad kinit.

```
kinit root/admin

klist
```

- Ahora utilizamos un cliente de un servicio kerberizado. El cliente será ldapsearch que ordenará una búsqueda a slapd. Entre ldapsearch y slapd se negociará una autenticación SASL-GSSAPI, de forma que ldapsearch pide un ticket del servicio LDAP utilizando el TGT para root/admin que acabamos de obtener, y este ticket de servicio que le concede el KDC lo almacena también en la caché de credenciales, el fichero /tmp/krb5cc_‘id -u‘ donde se encuentra el TGT. Por su lado slapd utiliza su principal exportado en su keytab para que el KDC le conceda su ticket de servicio. Con ambos tickets concedidos por el KDC, las dos partes pueden comprobar la identidad de la otra, dar por concluída la autenticación y pasar al uso del servicio LDAP (búsqueda de ou=users).

```
ldapsearch -LLL ou=users   # Usa SASL y no autenticacio'n simple
                           # al no pasar expli'citamente el flag -x.
                           # -LLL hace que sea menos verborreico.

klist                      # Aparecera' tambie'n el ticket de servicio recabado
                           # transparentemente para LDAP.
```

En este punto, tenemos dos identidades. A bajo nivel, la identidad local que utilizamos en el SO, la cual depende del proceso login de entrada al sistema y puede consultarse en cualquier momento con whoami. A alto nivel, la identidad centralizada que nos otorgan la infraestructura desplegada (servicios OpenLDAP y MIT Kerberos), la cual depende del TGT kerberos que obtuvimos ante el AS gracias a kinit, así como al mapeo entre principals KERBEROS y DN olcRootDN de OpenLDAP. Esta última identidad puede consultarse con ldapwhoami:

```
whoami
```

```
root
```

```
ldapwhoami
```

```
dn: uid=root/admin,ou=users,ou=accounts,dc=casafx,dc=dyndns,dc=org
```

Ambas identidades son independientes. Nosotros podemos hacen login en el SO con la cuenta "user" (en lugar de "root") y lanzar después kinit para obtener de nuevo la identidad uid=root/admin, ou=users, ou=accounts, dc=casafx, dc=dyndns, dc=org. La única condición que hay que cumplir para obtener una identidad es superar su proceso de autenticación (login en un caso, kinit en el otro etc).

Sin embargo, en la mayoría de los casos (cuando el usuario no es el administrador) querríamos mejor operar como sigue: nos autenticamos ante el proceso login de entrada al SO, pero éste contacta directamente con el AS Kerberos tal que tras la autenticación se expide un TGT y, además, se nos autoriza a usar los recursos directamente con la cuenta centralizada sin que intervenga una local.

Podríamos decir, entonces, que hasta el momento nuestro sistema está preparado para tener los dos tipos de identidades, pero nuestro sistema de cuentas de shell no está preparado para utilizar la segunda identidad (la centralizada), sólo la local. Será en el capítulo para el "Servicio de Cuentas de Shell Centralizadas" donde configuraremos los sistemas NSS/PAM que el sistema operativo dispone para operar así [8].

[8] Para entonces, además, habremos desplegado previamente a OpenAFS, tal que el usuario final además

El sistema NSS mencionado necesitará que en LDAP se encuentren asociados al nodo algunos atributos y objetos adicionales (definidos en el esquema NIS, proveído por defecto por OpenLDAP). Básicamente esos atributos brindan la oportunidad de tener los metadatos que el SO utiliza para tratar una cuenta, centralizados. Así esos atributos proveen una manera de tener el uid, gid, home etc en OpenLDAP.

Lo importante del párrafo anterior es que, a continuación, vamos a crear una nueva identidad centralizada y, cuando creemos su nodo en OpenLDAP, podemos añadir los atributos que necesita el sistema NSS y tenerlos disponibles para más adelante[9].

1.1.9. Creación de una identidad de usuario final centralizada: "umea"

Vamos a crear los recursos MIT Kerberos y OpenLDAP para un nuevo usuario, umea[10].

- MIT Kerberos necesita que exista un principal asociado: umea@CASAFX.DYNDNS.ORG.
- Por su lado OpenLDAP necesita crear un nodo en la rama ou=users,ou=accounts del DIT dc=casafx,dc=dyndns,dc=org. Como se ha comentado, su nodo en OpenLDAP llevará adicionalmente toda una serie de atributos que tendrán completo sentido en el capítulo del "Servicio de Cuentas de Shell Centralizadas".

Para la creación del principal:

---

de entrar con una única cuenta centralizada, reciba un único directorio Home sea cual sea la máquina en que haga login.

[9] Téngase en cuenta que la identidad root/admin utiliza en OpenLDAP al olcRoodDN para existir, por lo que anteriormente no necesitamos remarcar nada de ésto al trabajar con root/admin. Sin embargo la identidad umea de a continuación sí tiene su nodo convencional bajo ou=users,ou=accounts,dc=casafx,dc=dyndns,dc=org.

[10] Umeå es el nombre de la ciudad más grande del norte de Suecia. Su universidad participa junto a la de Jaén en el programa Erasmus.

```
kadmin.local
 >addprinc -policy user umea@CASAFX.DYNDNS.ORG
  Enter password for principal "umea@CASAFX.DYNDNS.ORG":
umeapa55
 >quit
# Hemos usado kadmin.local y no kadmin por omitir ma's explicaciones;
# no obstante e'ste seri'a el proceso: para usar kadmin necesitamos
# tener en cuenta las ACL de kadmind (definidas en /etc/krb5kdc/kadm5.acl)
# las cuales permiten realizar cambios a cualquier principal con instance
# "/admin". Por tanto antes de usar kadmin, conseguiri'amos un TGT para
# el principal root/admin utilizando a kinit. En el prompt de kadmin
# efectuari'amos las mismas acciones que hemos mostrado para kadmin.local.
```

Para el nodo en LDAP y demás atributos creamos un fichero LDIF:

```
vim ~/ldif/umea_user_group.ldif
```

```
dn: cn=umea,ou=groups,ou=accounts,dc=casafx,dc=dyndns,dc=org
cn: umea
gidNumber: 31000
objectClass: top
objectClass: posixGroup

dn: uid=umea,ou=users,ou=accounts,dc=casafx,dc=dyndns,dc=org
uid: umea
uidNumber: 31000
gidNumber: 31000
cn: umea casafx user
sn: nydala
objectClass: top
objectClass: person
objectClass: posixAccount
objectClass: shadowAccount
loginShell: /bin/bash
homeDirectory: /afs/casafx.dyndns.org/user/u/um/umea
userPassword: {CRYPT}*
#jpegPhoto:< file://path/to/jpeg/file.jpg
#userSMimeCertificate:
#preferedLanguage:
#mail: umea@casafx.dyndns.org
```

Cargamos el fichero LDIF utilizando ldapadd (equivalente a ldapmody -a). Previamente conseguimos un ticket de root/admin para llevar a cabo exitosamente la autenticación SASL-GSSAPI:

```
kinit root/admin
ldapadd -Qf ~/ldif/umea_user_group.ldif


kdestroy
```

La identidad centralizada "umea" ya está creada. Podemos comprobarlo:

```
kinit umea    # Obtenemos TGT umea.
klist         # Inspeccionamos la cache' de credenciales, el TGT esta' ahi'.
ldapwhoami -Q # Comprobamos el mapeo principal->nodo en LDAP. -Q para quiet.
klist         # El ticket de servicio LDAP se ha an~adido a la cache'.


ldapwhoami -d "-1" 2>&1 | less
# Con -d "-1" ldapwhoami devolvera' la ma'xima cantidad posible de detalles:
#http://www.openldap.org/doc/admin24/runningslapd.html#Command-Line%20Options
```

1.2. DKLAB2

1.2.1. Instalación de OpenLDAP

```
apt-get install slapd ldap-utils libsasl2-modules-gssapi-mit
```

```
-"Omit OpenLDAP server configuration?"
No
-"DNS domain name to construct the base DN of the LDAP directory:"
casafx.dyndns.org
-"Organization name:"
casafx.dyndns.org
-"Administrator password:"
ldapadmin
-"Database backend to use:"
hdb
-"Do you want the database to be removed when slapd is purged?"
No
-"Allow LDAPv2 protocol?"
No
```

```
nmap -p 389 dklab1.casafx.dyndns.org
nmap -p 389 localhost

mkdir ~/ldif
slapcat -b cn=config                    > ~/ldif/ldap-pristine_cn_config.ldif
slapcat -b dc=casafx,dc=dyndns,dc=org   > ~/ldif/ldap-pristine_dc_casafx.ldif
cp -R /var/lib/ldap                       ~/ldif/ldap_pristine_var-lib-ldap
```

1.2.2. Modificaciones sobre clientes LDAP dependientes de libldap

```
vim /etc/ldap/ldap.conf
```

```
...
####fx
BASE    dc=casafx,dc=dyndns,dc=org
#URI     ldap://dklab2.casafx.dyndns.org ldap://dklab1.casafx.dyndns.org
#...en principio podemos forzar a que se usen los registros SRV pues so'lo
#   los comandos ldap* fallara'n (u'sese -H o expo'rtese la variable LDAPURI;
#   segu'n man ldap.conf, incluso se pueden exportar LDAPCONF y LDAPRC
#   para .conf alternativos), pero si surgen problemas con los servicios
#   habri'a que explicitar las uri como ahi'. Referencia:
#   http://www.rjsystems.nl/en/2100-dns-discovery-openldap.php
#
####endfx
```

1.2.3. Modificaciones sobre el servidor LDAP; Directory Information Tree's

Eliminamos la cuenta creada por debconf:

```
ldapdelete -xh localhost -D cn=admin,dc=casafx,dc=dyndns,dc=org \
           -w ldapadmin cn=admin,dc=casafx,dc=dyndns,dc=org
ldapsearch -LLLQY EXTERNAL -H ldapi:/// -b dc=casafx,dc=dyndns,dc=org
```

A continuación recorremos el árbol LDAP para construir dos ficheros LDIF que lo modifiquen. Al igual que hicimos en dklab1, el primero carga un esquema para MIT Kerberos, y el segundo modifica el resto del árbol, pero con una excepción: puesto que el DIT dc=casafx,dc=dyndns,dc=org estará bajo replicación, los cambios sobre ese DIT no los aplicamos ahora sino que esperamos a la primera replicación. Así, el segundo fichero LDIF realiza modificaciones sobre cn=config sólamente.

cn=config

Añadimos un esquema para MIT Kerberos Nos bastará con copiar el LDIF dejado en dklab1:

```
scp dklab1.casafx.dyndns.org:~/ldif/kerberos.ldif ~/ldif/
```

Ya podemos cargar el schema tal como se indicó en dklab1:

```
ldapmodify -a -QY EXTERNAL -H ldapi:/// -f ~/ldif/kerberos.ldif


ldapsearch -LLLQY EXTERNAL -H ldapi:/// -b cn=config cn={4}kerberos
```

Cambios en el nivel de logging; none->stats Las siguientes modificaciones estarán en el segundo fichero LDIF.

Volvemos a repetir las precauciones pertinentes: se recomienda respetar los espacios en blanco: no añadir donde no los hay ni suprimir donde los hay. Excepto el primer fragmento, los demás suelen tener un espacio en blanco que separa el código LDIF anterior del comentario para el código LDIF subsiguiente etc; otras veces no aparece dicho espacio sino un guión "-" y salto de línea, que indica al procesador LDIF que debe aprovechar el DN sobre el que se produjo la modificación anterior. Consúltese "man ldif" en caso de duda.

```
vim ~/ldif/ldap2_olc-mod.ldif
```

```
##############################################################
#####                    DB cn=config                    #####
##############################################################


#### Loglevels en:
#http://www.openldap.org/doc/admin24/slapdconfig.html#Global%20Directives
dn: cn=config
changetype: modify
replace: olcLogLevel
olcLogLevel: stats
```

Mapeos necesarios para las autenticaciones entre principals y nodos (Las entradas olcAuthzRegexp: aparecen divididas en 3 y 2 partes delimitadas por \ y cambio de línea. Realmente cada una debe aparecer en una sóla línea, sin \ y cambio de línea:)

```
#### Sasl Auth Mapping
#www.openldap.org/doc/admin24/sasl.html#Mapping%20Authentication%20Identities
dn: cn=config
changetype: modify
add: olcAuthzRegexp
#Recordamos que el si'mbolo "\" y el salto de li'nea sobran en olcAuthRegexp:
olcAuthzRegexp: uid=ldap/([^/\.]+).casafx.dyndns.org,cn=casafx.dyndns.org,\
cn=gssapi,cn=auth \
cn=$1,ou=consumers,dc=casafx,dc=dyndns,dc=org
-
add: olcAuthzRegexp
#Recordamos que el si'mbolo "\" y el salto de li'nea sobran en olcAuthRegexp:
olcAuthzRegexp: uid=([^,]+),cn=casafx.dyndns.org,cn=gssapi,cn=auth \
uid=$1,ou=users,ou=accounts,dc=casafx,dc=dyndns,dc=org
-
add: olcSaslRealm
olcSaslRealm: CASAFX.DYNDNS.ORG
# La siguiente le permite a slapd saber que' key escoger del
# keytab: ldap/<olcSaslHost>@<olcSaslRealm>, pero yo so'lo tendre' una.
# -
#add: olcSaslHost
#olcSaslHost: dklab2.casafx.dyndns.org
```

Soporte para replicación LDAP: carga del módulo, declaración de los ID

```
#### Ldap-replication

# Carga del mo'dulo para replicacio'n
dn: cn=module{0},cn=config
changetype: modify
add: olcModuleLoad
olcModuleLoad: syncprov

# Server ID, que identifica a este slapd uni'vocamente, en Multi-Master
# Replication, como fuente. Son un nu'mero de hasta 3 di'gitos y
# una opcional url, que si aparece permite tener varios atributos
# olcServerID para anunciar a todos los slapd del grupo Multi-Master
# (pero es opcional excepto quiza's cuando replicas tambie'n el
# DIT cn=config). En man slapd-config dice que el FQDN "should be use"
# como url.
dn: cn=config
changetype: modify
replace: olcServerID
olcServerID: 002 ldap://dklab2.casafx.dyndns.org
#olcServerID: 001 ldap://dklab1.casafx.dyndns.org No hace falta si
#                                                 no replico {0}config.
```

olcDatabase={1}hdb,cn=config; olcRootDN, LDAP-replication, Acl, Índices

olcRootDN Aprovechando el mapeo principal KERBEROS a nodos bajo ou=users,ou=account asignábamos un nombre adecuado al olcRootDN. El atributo olcRootPW aún no debíamos eliminarlo.

```
dn: olcDatabase={1}hdb,cn=config
changetype: modify
replace: olcRootDN
olcRootDN: uid=root/admin,ou=users,ou=accounts,dc=casafx,dc=dyndns,dc=org
#-
#delete: olcRootPW
```

Configuración de replicación LDAP La sección para proveedor es idéntica a dklab1, la de consumidor será su simétrico:

```
#### Ldap-Replication
#    http://tools.ietf.org/html/rfc4533
#Overlay syncprov hace referencia al rol de provider
#Syncrpl          hace referencia al rol de consumer
#Mirrormode       permite las escrituras

#Provider:
#http://www.zytrax.com/books/ldap/ch6/syncprov.html
dn: olcOverlay=syncprov,olcDatabase={1}hdb,cn=config
changetype: add
objectClass: olcOverlayConfig
objectClass: olcSyncProvConfig
olcOverlay: syncprov
olcSpCheckpoint: 100 5
```

```
#Consumer:
#http://www.openldap.org/doc/admin24/slapdconfig.html#syncrepl
dn: olcDatabase={1}hdb,cn=config
changetype: modify
add: olcSyncRepl
olcSyncRepl: rid=002
             provider=ldap://dklab1.casafx.dyndns.org
             type=refreshAndPersist
             retry="5 5 300 +"
             searchbase="dc=casafx,dc=dyndns,dc=org"
             syncdata=default
             bindmethod=sasl
             saslmech=gssapi
-
#Mirror mode:
add: olcMirrorMode
olcMirrorMode: TRUE
```

```
#Limits:
#http://www.openldap.org/doc/admin24/limits.html
#Por defecto hay unos li'mites en el taman~o de datos,
#que nosotros vamos a revertir:
-
add: olcLimits
olcLimits: dn.onelevel="ou=consumers,dc=casafx,dc=dyndns,dc=org"
           size.soft=unlimited
           size.hard=unlimited
           time.soft=unlimited
           time.hard=unlimited
```

ACL Serán idénticas a las explicadas para slapd en dklab1.

```
# Eliminamos acl preexistentes
# -1
dn: olcDatabase={1}hdb,cn=config
changetype: modify
delete: olcAccess
olcAccess: {2}to *
  by self write
  by dn="cn=admin,dc=casafx,dc=dyndns,dc=org" write
  by * read
-
# -2
delete: olcAccess
olcAccess: {1}to dn.base=""
  by * read
-
# -3
delete: olcAccess
olcAccess: {0}to attrs=userPassword,shadowLastChange
  by self write
  by anonymous auth
  by dn="cn=admin,dc=casafx,dc=dyndns,dc=org" write
  by * none
```

```
-
# An~adimos acl (se evalu'an en orden, el comportamiento es OR'ing)
# +1 userPass
add: olcAccess
olcAccess: to attrs=userPassword,shadowLastChange
  by dn.one="ou=consumers,dc=casafx,dc=dyndns,dc=org" read
  by anonymous auth
  by * none
-
# +2 KRB subtree
add: olcAccess
olcAccess: to dn.subtree="ou=krb5,dc=casafx,dc=dyndns,dc=org"
  by dn="cn=adm-srv,ou=krb5,dc=casafx,dc=dyndns,dc=org" write
  by dn="cn=kdc-srv,ou=krb5,dc=casafx,dc=dyndns,dc=org" read
  by dn.one="ou=consumers,dc=casafx,dc=dyndns,dc=org" read
  by * none
-
# +3 loginShell
add: olcAccess
olcAccess: to attrs=loginShell
  by self write
  by dn.one="ou=consumers,dc=casafx,dc=dyndns,dc=org" read
  by * read
```

```
-
# +4 dn.base
# e'sto permite poder consultar los mecanismos de auth soportados, ejem:
# ldapsearch -H ldap://dklab1.casafx.dyndns.org -x -b "" \
#            -s base -LLL supportedSASLMechanisms
add: olcAccess
olcAccess: to dn.base=""
  by * read
-
# +5 accounts, sus atributos no son informacio'n sensible (el pass esta' en
# kerberos) asi' que debieran poder ser consultadas ano'nimamente:
#  ldapsearch -x -b ou=accounts,dc=casafx,dc=dyndns,dc=org
# E'sto facilitara' el despliegue, pero si en un momento futuro la poli'tica
# cambia, slapd estara' (al final de este capi'tulo) kerberizado y se podra'
# pedir autenticacio'n robustamente.
add: olcAccess
olcAccess: to dn.subtree="ou=accounts,dc=casafx,dc=dyndns,dc=org"
  by * read
```

```
-
# +6 resto
add: olcAccess
olcAccess: to *
  by users read
  by * none
```

Índices Tal como en dklab1:

```
#### I'ndices
# 1
dn: olcDatabase={1}hdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: uid eq
-
# 2
add: olcDbIndex
olcDbIndex: cn eq
-
# 3
add: olcDbIndex
olcDbIndex: ou eq
-
# 4
add: olcDbIndex
olcDbIndex: dc eq
-
# 5
add: olcDbIndex
olcDbIndex: uidNumber eq
-
# 6
add: olcDbIndex
olcDbIndex: gidNumber eq
```

```
-
# 7
add: olcDbIndex
olcDbIndex: memberUid eq
-
# 8
add: olcDbIndex
olcDbIndex: uniqueMember eq
-
# 9
add: olcDbIndex
olcDbIndex: krbPrincipalName eq,pres,sub
-
# 10
add: olcDbIndex
olcDbIndex: krbPwdPolicyReference eq

# 11 (relacionado con replicacio'n)
dn: olcDatabase={1}hdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: entryUUID eq
-
# 12 (relacionado con replicacio'n)
add: olcDbIndex
olcDbIndex: entryCSN eq
```

dc=casafx, dc=dyndns, dc=org (esperamos a la 1ª replicación)

Efectivamente se usará la primera replicación con dklab1, lo cual no pre-requiere tener aquí creados ni el nodo ou=consumers, ni el nodo ou=krb5, ni el nodo ou=accounts etc.

```
##############################################################
#####        DB dc=casafx,dc=dyndns,dc=org                #####
##############################################################

# No se modificara' nada en este DIT, esperaremos a la primera replicacio'n.

#############################################################################
#                                  FIN
#############################################################################
```

Ya podemos cargar, no obstante, el segundo LDIF tal como se indicó:

```
ldapmodify -QY EXTERNAL -H ldapi:/// \
           -f ~/ldif/ldap2_olc-mod.ldif \
           -c -S ~/ldif/ldiferrors-‘date +%Y%m%d.%H%M_%S‘.ldif
```

1.2.4. Instalación de MIT Kerberos

```
apt-get install krb5-admin-server krb5-kdc krb5-kdc-ldap \
                krb5-config krb5-user
```

```
-"Default Kerberos version 5 realm:"
CASAFX.DYNDNS.ORG
-"Add locations of default Kerberos servers to /etc/krb5.conf?"
No
-"Create the Kerberos KDC configuration automatically?"
Yes
-"Run the Kerberos V5 administration daemon (kadmind)?"
Yes
```

Configuramos la dependencia de precedencia de slapd sobre los servicios de MIT Kerberos durante el arranque de la máquina:

```
vim /etc/init.d/krb5-kdc
```

Para no perturbar posiblemente el formato, diremos ahora que el cambio consiste en añadir slapd tal como se indica:

```
### BEGIN INIT INFO
...
# Required-Start:       $local_fs $remote_fs $network $syslog slapd
# Required-Stop:        $local_fs $remote_fs $network $syslog slapd
...
### END INIT INFO
```

```
vim /etc/init.d/krb5-admin-server
```

```
### BEGIN INIT INFO
...
# Required-Start:       $local_fs $remote_fs $network $syslog slapd
# Required-Stop:        $local_fs $remote_fs $network $syslog slapd
...
### END INIT INFO
```

```
insserv /etc/init.d/krb5-kdc
insserv /etc/init.d/krb5-admin-server
```

Y como último preámbulo a la configuración de MIT Kerberos, nos aseguramos que el FQDN que se da a sí misma la máquina coincide con el que le da el resolvedor DNS, y que la resolución directa e inversa son biyectivas:

```
FQDN=‘hostname --fqdn‘
RDIR=‘dig @10.168.1.2 ${FQDN} A +short‘
RINV=‘dig -x @10.168.1.2 $(echo $RDIR | tac -s \. |
      tr ’\n’ ’.’)IN-ADDR.ARPA. any +short | tr ’\n’ ’ ’‘
printf "${FQDN}\t -> ${RDIR},\n ${RDIR}\t\t\t -> ${RINV}\n"
```

```
dklab2.casafx.dyndns.org          -> 10.168.1.2
10.168.1.2                        -> dklab2.casafx.dyndns.org. (y nada ma’s)
```

1.2.5. Modificaciones sobre clientes KERBEROS

```
vim /etc/krb5.conf
```

```
[libdefaults]
        default_realm = CASAFX.DYNDNS.ORG
        ...
        forwardable = true
        proxiable = true
        ####fx:
        #si versio'n >=1.7 y telnet u openafs, explicitar:
        allow_weak_crypto = true
        # Si algo falla, podemos activar debug:
        #debug = true
        ####endfx
...
[realms]
####fx
        CASAFX.DYNDNS.ORG = {
        # SRV _kerberos._ucp.casafx.dyndns.org. es para los kdc:
        #kdc = krb2.casafx.dyndns.org
        #kdc = krb1.casafx.dyndns.org

        # SRV kerberos-adm._tcp.casafx.dyndns.org
        # ... pero no esta' claro ue kadmin lo use au'n, por lo
        # tanto los hacemos expli'citos localmente:
        admin_server = dklab2.casafx.dyndns.org
        admin_server = dklab1.casafx.dyndns.org
        database_module = openldap_ldapconf
}
####endfx
```

```
...
[domain_realm]
####fx:
        # TXT _kerberos.casafx.dyndns.org. debiera valer para
        # mapear ambos:
        #.casafx.dyndns.org = CASAFX.DYNDNS.ORG
        #casafx.dyndns.org = CASAFX.DYNDNS.ORG
####endfx
...
[login]
...
####fx:
# Atencio'n: no confu'ndase la seccio'n login (de la que no
# especificamos nada pero que precede) con esta seccio'n logging
# que creamos ahora:
[logging]
        kdc = FILE:/var/log/krb5/kdc.log
        admin_server = FILE:/var/log/krb5/kadmin.log
        default = FILE:/var/log/krb5/klib.log
        # Quiza's pueda ser interesante conocer que puede usar a syslog:
        # default = SYSLOG:INFO:LOCAL6
```

```
[dbdefaults]
        ldap_kerberos_container_dn = ou=krb5,dc=casafx,dc=dyndns,dc=org
[dbmodules]
         openldap_ldapconf = {
         db_library = kldap
         ldap_kdc_dn = cn=kdc-srv,ou=krb5,dc=casafx,dc=dyndns,dc=org
         ldap_kadmind_dn = cn=adm-srv,ou=krb5,dc=casafx,dc=dyndns,dc=org
         ldap_service_password_file = /etc/krb5kdc/service.keyfile
         ldap_conns_per_server = 5
         }
####endfx
```

Rotación de los logs declarados:

```
mkdir /var/log/krb5


vim /etc/logrotate.d/krb5
```

```
/var/log/krb5/kadmin.log /var/log/krb5/kdc.log /var/log/krb5/klib.log {
       daily
       missingok
       rotate 7
       compress
       delaycompress
       notifempty
}
```

1.2.6. Modificaciones sobre los servidores de MIT Kerberos

No creamos el dominio administrativo en LDAP, como tampoco el principal root/admin: esperaremos a la primera replicación.

Obtención del stashfile y el fichero de credenciales LDAP

Por su lado, stash y service.keyfile se copiarán de dklab1 utilizando un canal seguro:

```
scp dklab1.casafx.dyndns.org:/etc/krb5kdc/s* /etc/krb5kdc/
ls /etc/krb5kdc/s*
```

ACL de kadmind

Modificamos las ACL del gestor de la base de datos, kadmind:

```
vim /etc/krb5kdc/kadm5.acl
```

```
####fx:
*/admin *
####endfx
```

Configuración de los subservicios KDC/AC/KDBM: /etc/krb5kdc/kdc.conf

Ya se modificó el período de duración de los tickets permitida en dklab1 en conexión a su kadmind. En dklab2 queda sólo declararlo.

```
vim /etc/krb5kdc/kdc.conf
```

```
[kdcdefaults]
    ####fx:
    # Si algo falla, podemos activar debug:
    #debug = true
    ####endfx
...
[realms]
    CASAFX.DYNDNS.ORG = {
    ...
####fx:
        max_life = 1d 0h 0m 0s
        max_renewable_life = 90d 0h 0m 0s
#-      max_life = 10h 0m 0s
#-      max_renewable_life = 7d 0h 0m 0s
####endfx
...
```

Reinicio de MIT Kerberos (no)

Aún no se puede lanzar el KDC en dklab2.

1.2.7. Kerberización de OpenLDAP

```
apt-get install libsasl2-modules-gssapi-mit
```

Creación del principal para OpenLDAP en dklab2; exportación al keytab

Nos conectamos a kadmind en dklab1 y creamos el principal:
ldap/dklab2.casafx.dyndns.org@CASAFX.DYNDNS.ORG

```
mkdir -p /etc/keytab.d


kadmin -p root/admin -s dklab1.casafx.dyndns.org
```

```
Password for root/admin@CASAFX.DYNDNS.ORG:
r00tprincipa!
```

```
>addprinc -policy service -randkey ldap/dklab2.casafx.dyndns.org
>ktadd -k /etc/keytab.d/openldap.keytab \
       -norandkey ldap/dklab2.casafx.dyndns.org
>quit
```

```
klist -ke /etc/keytab.d/openldap.keytab


chown openldap:root /etc/keytab.d/openldap.keytab
chmod go-rw /etc/keytab.d/openldap.keytab
su openldap -m -c 'klist -t -K -k /etc/keytab.d/openldap.keytab'
```

Anunciamos a slapd la localización de su keytab:

```
vim /etc/default/slapd
```

```
...
####fx:
export KRB5_KTNAME="FILE:/etc/keytab.d/openldap.keytab"
#LDAPGSSAPI_ENCRYPT=yes # Es el default
####endfx
```

Configuración de libldap para uso de SASL-GSSAPI

```
vim /etc/ldap/ldap.conf
```

```
...
####fx:
SASL_MECH GSSAPI
SASL_REALM CASAFX.DYNDNS.ORG
#SASL_ENCRYPT yes
####endfx
```

Adaptación de slapd para utilizar SASL-GSSAPI como consumidor; kstart

Repetimos los pasos dados en dklab1:

```
apt-get install kstart


vim /etc/inittab
```

(La línea KSL:... aparece dividida en 2 partes delimitadas por \ y cambio de línea. Realmente debe aparecer en una sóla línea, sin \ y cambio de línea:)

```
####fx:
#Nota: init no nos permite poner el nombre del principal porque excederi'a
#el ma'ximo de caracteres permitido por li'nea, entonces usamos -U para que
#k5start coja el primer principal exportado en /etc/keytab.d/openldap.keytab
#Debe tenerse en cuenta este detalle ante cualquier modificacio'n futura
#en ese keytab: k5start espera encontrarse el suyo en primer lugar.
KSL:2345:respawn:/usr/bin/k5start -U -f /etc/keytab.d/openldap.keytab \
-K 10 -l 24h -k /tmp/krb5cc_`id -u openldap` -o openldap
####endfx
```

1.2.8. Replicación inicial de OpenLDAP y ejecución de MIT Kerberos

Previamente, vamos a volcar en ficheros lo que debe ser el árbol LDAP en dklab2; el DIT para nuestra organización (dc=casafx,dc=dyndns,dc=org) está configurado para ser replicado y, por tanto, será indistinguible de el de dklab1 y podemos volcar el de allí. Por su lado, el DIT cn=config es propio de cada slapd, por lo que sólo podemos volcarlo aquí.

```
invoke-rc.d slapd stop
ssh dklab1.casafx.dyndns.org \
    "slapcat -b dc=casafx,dc=dyndns,dc=org" \
    | cat > ~/ldif/ldap1_dc_casafx.ldif
slapcat -b cn=config > ~/ldif/ldap2_cn_config.ldif
```

Por tanto, ante cualquier problema podemos restaurar utilizando esos volcados; es útil la siguiente secuencia usando slapadd:

```
invoke-rc.d slapd stop
/bin/rm /var/lib/ldap/*
su -m openldap -c "/usr/sbin/slapadd -l ~/ldif/ldap2_cn_config.ldif
                                     -l ~/ldif/ldap1_dc_casafx.ldif"
su -m openldap -c /usr/sbin/slapindex
invoke-rc.d slapd start
```

1.2.9. Tests

```
kinit root/admin

klist

ldapsearch -LLL cn=dklab2

klist

whoami

ldapwhoami

kdestroy


kinit umea

klist

ldapwhoami

klist

kdestroy
```

1.3. Confidencialidad proveída por la implementación Cyrus de SASL-GSSAPI

La información almacenada en ldap es extremadamente sensible (pues allí está todo el material criptográfico de kerberos).

La principal motivación, por tanto, de estas capturas es comprobar que las credenciales obtenidas tras la autenticación SASL-GSSAPI, se utilizan para encriptar el tráfico subsiguiente tal como estipula el rfc4752 en su sección 3.2 ("Security Context"). Ésto hace que las transacciones con slapd sean autenticadas, autorizadas y confidenciales. Adicionalmente, no necesitamos proveer esa confidencialidad utilizando una capa adicional como TLS (algo complicado, por cierto, en slapd; se recomienda recompilar contra openssl primero), tal como ocurre en escenarios sin KERBEROS.

Para hacer la captura, lanzamos la herramienta tshark a capturar el tráfico; entonces, efectuamos una autenticación SASL-GSSAPI y, a continuación, una simple.

```
tshark -V -i ovpnCASAFX-SRV -d tcp.port==389,ldap -R tcp.port==389 -x -S \

       | grep -A10 Light
```

1.3.1. Capturas sobre autenticación SASL-GSSAPI

Efectivamente, el tráfico está encriptado y tshark no puede decodificar los mensajes LDAP. Por tanto Cyrus SASL, la implementación de SASL que utiliza slapd y sus utilidades, parece implementar completamente el rfc4752:

```
kinit -p root/admin

ldapsearch -H ldap://dklab2.casafx.dyndns.org \

           -b ou=users,ou=accounts,dc=casafx,dc=dyndns,dc=org

kdestroy
```

Y el fragmento significativo:

```
...
Lightweight-Directory-Access-Protocol

0000  45 00 00 c1 ed 45 40 00 40 06 35 9f 0a a8 01 01   E....E@.@.5.....
0010  0a a8 01 02 b6 0d 01 85 36 c5 95 ad 34 d3 d2 b3   ........6...4...
0020  80 18 06 c0 3f 3a 00 00 01 01 08 0a 02 cb 81 85   ....?:..........
0030  02 c9 b0 40 00 00 00 89 05 04 06 ff 00 00 00 00   ...@............
0040  00 00 00 00 01 bb 5d 6f 46 9a 05 dc bd 08 70 a8   ......]oF.....p.
0050  08 d5 37 2c ae 44 30 73 fc 99 37 ac 1e 0d 3c be   ..7,.D0s..7...<.
0060  f2 30 2a a8 68 a5 05 b2 07 fb 22 25 f0 44 36 75   .0*.h.....\"%.D6u
0070  94 63 e5 13 26 ca 94 fc d8 11 4a b1 f6 b7 a0 ff   .c..&.....J.....
0080  a2 51 96 65 fc a8 a9 c0 31 d7 f8 22 c3 f7 a0 3b   .Q.e....1..\"...;
--
Lightweight-Directory-Access-Protocol

0000  45 00 00 dc 7f e8 40 00 40 06 a2 e1 0a a8 01 02   E.....@.@.......
0010  0a a8 01 01 01 85 b6 0d 34 d3 d2 b3 36 c5 96 3a   ........4...6..:
0020  80 18 08 48 55 7e 00 00 01 01 08 0a 02 c9 b0 85   ...HU~..........
0030  02 cb 81 85 00 00 00 a4 05 04 07 ff 00 00 00 00   ................
0040  00 00 00 00 1e ac f8 5d c8 f4 ff f7 c8 fe 88 ac   .......]........
0050  e5 3e 34 e3 59 d2 62 13 12 b1 31 40 ce f2 c8 cd   .>4.Y.b...1@....
0060  5a 34 60 a3 1b e8 a0 2d 9d 8c ca 43 74 0f 31 53   Z4‘....-...Ct.1S
0070  f2 1f ef 4d dd a3 fc db b5 13 69 da d1 6d 06 b4   ...M......i..m..
0080  d5 d1 a1 25 f9 f1 15 63 6d 14 39 b7 49 8c dd 7b   ...%...cm.9.I..{
--
...
```

1.3.2. Capturas sobre autenticación simple

Éste es el aspecto cuando usamos bind simple anonymous (-x), efectivamente tshark lo recibe sin encriptar ahora, luego puede decodificarlo y darle un formato:

```
ldapsearch -x -H ldap://dklab2.casafx.dyndns.org \
           -b ou=users,ou=accounts,dc=casafx,dc=dyndns,dc=org
```

```
...
Lightweight-Directory-Access-Protocol
    LDAPMessage searchRequest(2) "ou=accounts,dc=casafx,dc=dyndns,dc=org" wholeSubtree
        messageID: 2
        protocolOp: searchRequest (3)
            searchRequest
                baseObject: ou=accounts,dc=casafx,dc=dyndns,dc=org
                scope: wholeSubtree (2)
                derefAliases: neverDerefAliases (0)
                sizeLimit: 0
                timeLimit: 0
                typesOnly: False
...
```

LyX