ANEXO 7: SHELLPOSIX

Índice general

Capítulo 1

Servicio de Cuentas Centralizadas de Shell POSIX

Contents

Hasta ahora, LDAP y MIT Kerberos (que a su vez usa LDAP como almacén) contienen de una forma centralizada los metadatos necesarios para el servicio de cuentas de shell POSIX. Efectivamente, MIT Kerberos dispone el protocolo y las credenciales de autenticación, mientras que en LDAP nos adelantamos a cargar metadatos (la mayoría típicos de una cuenta de shell POSIX tales como el par uid/gid -éstos base del modelo de autorización en SO POSIX-, el directorio home o la shell por defecto). OpenAFS por su lado provee almacenamiento distribuído de ficheros con autenticación basada en KERBEROS.

Ninguno de ellos proveería por sí sólo un servicio que ofrecer al usuario final, pero es que el nivel de integración entre ellos que hemos obtenido hasta ahora tampoco es óptimo para tal fin.

1.1. Introducción al bloque de servicios finales

Hasta ahora podemos lanzar comandos que nos autentican, resuelven metadatos etc, pero lo ideal es que esos procesos se integren y lleven a cabo automaticamente al entrar al sistema (log-in), constituyendo, entonces sí, un autentico servicio: el de Cuenta Centralizada de Shell POSIX. Un razonamiento parecido se podría hacer para cualquier servicio de nuestra infraestructura; como conclusión: tras desplegar la infraestructura de servicios base, tenemos que integrarla ahora con los servicios finales que se pretenden poner a disposición del usuario.

Por tanto comienza ahora un proceso de integración cuya primera fase es la integración de los servicios base con los subsistemas de log-in del SO para desarrollar un auténtico servicio de Cuenta Centralizada de Shell POSIX. En el caso de Debian, la parte del SO que orquesta la resolución de metadatos de sus cuentas (y otros, como los nombres de dominio) desde diferentes fuentes, se llama NSS. Por su lado la que hace de interfaz (API) para los programas que implementan el log-in en el sistema, es PAM. Así pues:

- el sistema PAM intermediará entre los programas de entrada a la infraestructura (log-in), permitiendo utilizar KERBEROS y obteniendo tickets TGT KERBEROS y Tokens AFS.
- el sistema NSS (apuntando a OpenLDAP), permitirá que el usuario acceda a los metadatos necesarios para hacer uso de una Cuenta Centralizada de Shell POSIX en el SO. Efectivamente no necesitará de la intermediación de una cuenta local, sino que directamente utilizará las identidades centralizadas.
    - Sabemos, además, que gracias a los sistemas de mapeo de identidades, éstas serán una identidad única en el sentido de que la identidad de autenticación de su cuenta en cualquier servicio será sistemáticamente la obtenida desde su principal.

Acabamos de mencionar que PAM puede obtener el Token AFS, y NSS puede obtener metadatos de LDAP como la localización del home: así, cuando el log-in se produzca,

el "Current Working Directory" de la shell será un directorio totalmente funcional en la celda AFS.

El despliegue para Cuentas Centralizadas de Shell POSIX no necesita más: la infraestructura está ahí y sólo queda que diferentes tipos de máquinas y SO la integren en el sentido comentado y la usen como clientes.

1.1.1. Alcance de este capítulo

Lo ideal, sería poder utilizar cualquier SO, no sólo Debian, como cliente de nuestra infraestructura. Por ello en este capítulo, tras explicar con todo detalle el caso de Debian (PAM, NSS etc), se harán comentarios para otros sistemas operativos.

Es más, hemos concluído que la parte de cliente de otros servicios que integramos en otros capítulos (mensajería instantánea, VoIP, correo) así como las instrucciones de registro de un nuevo usuario en todos los servicios de nuestra infraestructura, son en general cuestiones tanto algo más sencillas y repetidas como, sobre todo, más susceptibles de utilizarse en múltiples ocasiones: merecería la pena que estuviesen recogidas en un sólo documento en lugar de sólo distribuídas en cada capítulo posterior.

Por tanto, resumiendo, en este capítulo mostramos:

- Cómo integrar los servicios base con PAM y NSS en dklab1 y dklab2, en el estilo de exposición que teníamos hasta ahora.
- Una sección extra sobre cómo incorporar una nueva computadora a nuestro parque informático, útil sobre todo cuando se haya desplegado éste con posteriores servicios. Consiste pues en describir qué software instalar para que pueda asumir el rol de cliente de nuestra infraestructura telemática (todos los servicios).
    - Tanto dklab1 como dklab2 son sistemas Debian que han sido configurados como servidores pero también como clientes, así que no será difícil hacer sumario de qué hace falta desplegar para que cualquier otra máquina con Debian haga de cliente de la infraestructura.
    - En el caso de Windows se expondrá algo parecido, y se advertirá de sus deficiencias sin usar un controlador de dominio SAMBA.
    - En el caso de otros SO POSIX se darán algunas directrices generales y enlaces.
- Otra sección extra sobre cómo incorporar un nuevo usuario a nuestra organización; es decir cómo puede el administrador crear una cuenta y todos sus recursos asociados (incluyendo ficheros de configuración en su home AFS para las aplicaciones cliente) de todos los servicios.

1.1.2. Alcance de posteriores capítulos

¿Y después? Para que la infraestructura sea útil en una organización, debiera proveer más servicios: como comentábamos en adelante (otros capítulos), volveremos a reutili-

zar la infraestructura para desplegar otros servicios adicionales a las Cuentas de Shell POSIX. Estos servicios, volverán a usar a LDAP como contenedor de metadatos, usarán la memoria que proveen los keytabs o los ticket-granting-tickets KERBEROS del primer log-in para hacer sucesivas autenticaciones (SSO), y usarán OpenAFS en el caso de tener que almacenar información. Idealmente, el software cliente y servidor de esos servicios soportaría todos estos protocolos y sistemas, pero veremos que no siempre es así, y nos veremos obligados a instalar software intermediario o, si no queda más remedio, software adicional.

1.2. Sistema PAM

Como se ha introducido, PAM[1] es la interfaz del SO tal que permite a los programas de login consiguir los recursos para que el usuario disfrute del servicio de cuentas de shell POSIX. En el proceso, PAM ha de recabar y computar información diversa, siendo la mayoría de estos metadatos proveídos por el subsistema NSS del SO; en nuestro despliegue, además necesitará contactar con otros sistemas como el AS KERBEROS para completar la autentificación u OpenAFS para la autorización particular del sistema de archivos. Por último, PAM es suficientemente flexible como para permitir que se puedan probar secuencialmente varios tipos de autentificación para distintas o la misma cuenta[2].

El resultado final es que PAM efectúa la autentificación, recaba los metadatos que se usarán más tarde para autorizar al usuario durante su sesión (uid, gid, token afs...), y recaba también el resto de metadatos relativos a la cuenta (shell, home...). A partir de ahí, el usuario podrá utilizar su shell para ejecutar programas y, en general, acceder al resto de servicios.

Debemos tomar una serie de decisiones para el comportamiento particular del log-in. Queremos que las cuentas locales (ej. root) no necesiten KERBEROS para completar el log-in. Además, queremos la posibilidad de poder probar varios tipos de métodos de autentificación para una misma cuenta. Para ello necesitaremos una secuencia parecida a ésta en los ficheros de configuración de PAM:

```
<...> pam_unix.so

<...> pam_krb5.so minimum_uid=1000
```

Indica que primero se pruebe la autenticación POSIX convencional, y bajo ciertas convenciones, después la de KERBEROS AS; si invertimos el orden, primero se prueba con el KERBEROS AS, luego el método local convencional. En ambos casos si el usuario es local (ejemplo: root) o si es centralizado (ejemplo: umea), puede completar su autenticación. Pero además, el argumento minimum_uid nos permite que lo anterior pueda aplicarse a la misma cuenta pudiendo usar un sistema u otro por el orden de los prompts, o que el segundo esté en reserva por si momentáneamente falla el primero... La razón es que, siendo el uid unívoco para cada cuenta, adduser (la utilidad para crear usuarios locales)

[1] http://www.opengroup.org/rfc/mirror-rfc/rfc86.0.txt
[2] http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/

en su fichero de configuración /etc/adduser.conf distribuye los uid según el tipo de cuenta (de usuarios o del sistema) y su localidad (reserva los que serán uid de cuentas locales):

```
view /etc/adduser.conf +29
```

```
...
FIRST_SYSTEM_UID=100
LAST_SYSTEM_UID=999
FIRST_SYSTEM_GID=100
LAST_SYSTEM_GID=999
FIRST_UID=1000
LAST_UID=29999
FIRST_GID=1000
LAST_GID=29999
...
```

... es decir, si minimum_uid=1000 estaremos permitiendo autenticación POSIX y KERBEROS a usuarios de cuentas locales y centralizadas. Si minimum_uid=31000, estaremos ofreciendo KERBEROS sólo a cuentas que, si se sigue la política de adduser, no son locales. Nosotros usaremos la primera opción, pero se podría cambiar ese valor si se prefiere la segunda. Nuestra cuenta centralizada de ejemplo, uid=umea,ou=users,ou=accounts tiene de uid, precisamente, a 31000. La razón de no tomar directamente 30000 es reservar un margen entre nuestra política de uid para usuarios centralizados y la local de adduser.

Para que PAM autentique usando al AS KERBEROS se instalará la librería libpam-krb5.

- Anotación: limitaciones de PAM impiden usarlo para conseguir Kerberos Service Tickets, sólo es capaz de contactar con el Kerberos AS para conseguir el Ticket-Granting-Ticket's, TGT's.

1.3. Sistema NSS

El sistema NSS[3] permitía configurar básicamente en qué orden se consultan las fuentes de metadatos disponibles, tales como ficheros locales bien conocidos o servicios remotos LDAP, DNS/MDNS, NIS, SQL... El sistema contempla hasta 11 grupos de información, de

[3] http://www.gnu.org/software/libc/manual/html_node/Name-Service-Switch.html

los cuales nos interesan los grupos "passwd", "shadow" y "group". El primero representa la mayoría de metadatos de las cuentas POSIX (password, loginname, uid, sus gid, home, shell, comment) el segundo representa una base de password alternativa y el tercero son los metadatos de grupos (groupname, gid). Podemos mencionar también a "host", que representa las resoluciones de nombres de dominio, muy importantes en este despliegue.

Al crear la cuenta ejemplo uid=umea,ou=users le definimos una serie de atributos POSIX accesibles por tanto a través de LDAP. A la vez, en dependencia de las desisiones que comentamos con PAM, debemos contemplar el uso de cuentas locales y centralizadas. Dicho lo anterior, el siguiente orden de consulta de metadatos debiera funcionar:

1. Metadatos cuentas POSIX (llamados en conjunto grupo "*passwd*" de NSS): son loginname, password, uid, gid, comment, home, shell; se buscarán en: /etc/passwd primero, LDAP si no se encuentra allí.

2. Alternativa al metadato password: grupo "*shadow*" (se usa si en /etc/passwd hay una 'xén el campo para el password, incluye una versión encriptada del password y otros datos de administración de contraseñas). Es la opción preferida hoy en día para los password locales; se buscará en: /etc/shadow primero, LDAP si no hay éxito en /etc/shadow.

    a) Anotación: LDAP es necesario no por motivos de autenticación (no hay password alguno allí etc) sino debido al comportamiento de PAM en su tarea de autorización, como se verá luego.

3. Metadatos "*group*". Incluye el gid y el groupname; se buscarán en: /etc/group, LDAP

4. Metadatos "*hosts*": /etc/hosts, DNS

5. ... a continuación se declaran resoluciones para otros 7 grupos restantes contemplados por NSS: no nos interesan.

¿Y el acceso al password en KERBEROS? Aclaremos ésto: NSS provee la búsqueda del metadato password para cuentas locales (/etc/passwd, más común hoy día /etc/shadow) o cuentas centralizadas en LDAP donde el valor de password se encuentra allí en vez de en local. No permite, sin embargo, usar el Kerberos AS para contrastar el password porque el AS no es una base de datos, sino un servicio y protocolo de comunicación para que tenga lugar el proceso completo de autentificación. Dicho lo cual, ocurre:

- En lo relativo al metadato "password", en lugar de PAM usa a NSS, PAM usaría directamente uno de sus módulos (libpam-krb5) para contactar con el AS de KERBEROS etc, como decimos siguiendo el protocolo KERBEROS.

- No obstante si en PAM se prueban varios esquemas de autentificación, como en nuestro caso para permitir el log-in tanto usando cuentas locales como centralizadas, entonces se usa para las locales el método convencional pidiendo a NSS el atributo password. En ese caso, se contempla la posibilidad de que el campo password de

/etc/shadow contenga '*K*'y, tras ser recibido (NSS mediante) por PAM, le indica que sólo es posible autenticar esa cuenta particular a través de KERBEROS luego debiera dirijirse al AS KERBEROS con la contraseña que ya se le haya dado. Así, NSS sí tiene este pequeño papel en el comportamiento del log-in cuando se incluye KERBEROS entre otros mecanismos de autentificación. Detalle éste aparte, todo lo relativo a KERBEROS se configura en PAM y NSS no aporta nada.

Para que NSS utilice a LDAP como fuente de metadatos, necesitamos instalar la librería libnss_ldap (paquete libnss-ldapd), configurar dónde y cómo accederá a la información en LDAP, y usar los atributos que libnss_ldap espera encontar. En concreto, espera encontrar clases de objetos con atributos: uid, uidNumber, gidNumber, homeDirectory y loginShell. Efectivamente estos atributos están permitidos por el objectClass POSIXAccount[4] que utilizamos en su momento para crear a la cuenta ejemplo uid=umea,ou=users,ou=accounts. No obstante, por flexibilidad, incluso se podrían mapear los atributos en ldap con otros necesarios localmente, véase "man nslcd.conf".

1.4. DKLAB1

1.4.1. Instalación de soporte LDAP para el sistema NSS

Con la librería libnss_ldap del paquete libnss-ldapd (es importante la "d" final, pues existe otra implementación empaquetada en libnss-ldap y que nosotros no usamos) se instala por dependencias el demonio nslcd, el componente que se conecta a LDAP. Debconf:

```
apt-get install --no-install-recommends libnss-ldapd nslcd
```

```
-"LDAP server URI:"
DNS                    <- literal, hay que ponerlo
-"LDAP server search base:"
ou=accounts,dc=casafx,dc=dyndns,dc=org
- "LDAP authentication to use"
None
-"Use StartTLS?"
No.
-"Name services to configure (libnss-ldapd):"
group, passwd, shadow
```

[4] Como se explicará, hay que añadir la clase de objeto shadowAccount adicionalmente a la de POSIXAccount.

```
...
Adding new group ‘nslcd’ (GID 111) ...
Adding new user ‘nslcd’ (UID 108) with group ‘nslcd’ ...
Not creating home directory ‘/var/run/nslcd/’.
Starting LDAP connection daemon: nslcd.
Setting up libnss-ldapd (0.7.13) ...
/etc/nsswitch.conf: enable LDAP lookups for group
/etc/nsswitch.conf: enable LDAP lookups for passwd
```

```
ls /lib/i386-linux-gnu/libnss_ldap*


view /etc/nslcd.conf
```

Anotaciones sobre SASL-GSSAPI entre nslcd y slapd

Como se ha comentado en varias ocasiones, la política de acceso a los metadatos en los nodos LDAP que representan las cuentas, es de libre lectura ya que se considera que el material sensible (principal KERBEROS) no se almacena ahí, simplificando significativamente el despliegue. En cualquier caso, si esa condición cambiase slapd fue preparado para autenticaciones usando SASL-GSSAPI. Y reseñar para nslcd que efectivamente tiene soporte para acceder a OpenLDAP usando autenticación SASL-GSSAPI, pero para ello habría igualmente que configurarse a tal efecto, creando un principal y exportándolo en un keytab etc.

- La forma que tendría entonces el fichero /etc/nslcd.conf sería algo como:

```
...
#SASL
use_sasl on
sasl_mech GSSAPI
sals_realm CASAFX.DYNDNS.ORG
krb5_ccname FILE:/tmp/.krb5cc_nscdl
```

- Adicionalmente /etc/default/nslcd permitiría configurar los parámetros de k5start para conseguir el TGT previo a la autenticación SASL-GSSAPI. Así pues, nslcd se encargaría por sí sólo de mantener renovadas sus credenciales lanzando a k5start. El fichero /etc/default/nslcd sería algo como:

```
...
#NSLCD_IFACE=eth0
##Kerberos stuff
KRB5_KEYTAB=/etc/krb5.keytab
KRB5_CCNAME=/tmp/.krb5cc_nscdl
KRB5_PRINCIPAL="host/‘hostname --fqdn‘"
##time in minutes between cache updates
KRB5_CCREFRESH=60 #Adjust the options as necessary.
```

Como es habitual, el ticket TGT se obtiene a partir del principal exportado en un keytab. En el ejemplo se usa el que nosotros crearemos más adelante para OpenSSH.

1.4.2. Configuración del orden de resolución de metadatos en NSS

No existe aún un script "update-nss" que altere /etc/nsswitch, los cambios se hacen editando el fichero[5], si bien como propuesta preliminar, debconf al instalar libnss-ldapd nos preguntó cómo alterar la versión actual del /etc/nsswitch.conf. Debemos revisar los cambios que se hicieron; encontraremos algo como ésto (obbiando los comentarios):

```
view /etc/nsswitch.conf
```

[5]http://bugs.debian.org/cgi-bin/bugreport.cgi?bug%3D496915

```
...
####fx:
#-passwd:       compat
#-group:        compat
#-shadow:       compat
passwd:         compat ldap
group:          compat ldap
shadow:         compat ldap
####endfx

hosts:          files mdns4_minimal [NOTFOUND=return] dns mdns4
...
```

- Anotación:
  - Existen despliegues en que también se hacen búsquedas en LDAP para el grupo shadow; no es nuestro caso (usamos KERBEROS).
  - En la anteriores versiones de Debian (como Lenny), se generaban consultas LDAP para entidades inexistentes y al arranque el sistema se bloqueaba del orden de la media hora. La solución estaba en utilizar "compat" y no "files" (algo que tenemos hecho), así como utilizar las opciones expuestas en "man nslcd.conf", sección TIMING/RECONNECT OPTIONS. Se deja constancia de ello aunque parece estar solucionado en versiones posteriores de Debian.

1.4.3. Instalación del cacheador nscd; test

Cachea los resultados de NSS, él no realiza ninguna conexión. Existen dos implementaciones, nscd y unscd (drop-in-replacement de la anterior).

- Anotación: nscd provee rapidez en las resoluciones; también está pensado para cachear ante desconexiones duraderas, pero ésto no será problema en dklab1 o dklab2 y, por otro lado, hay soluciones más robustas para ello[6].

```
apt-get install nscd
```

[6] Consúltesehttp://people.skolelinux.org/pere/blog/Caching_password__user_and_group_on_a_roaming_Deb

Crearemos una cuenta local específica para autorizar a nscd; configuramos el demonio:

```
invoke-rc.d nscd stop
/usr/sbin/addgroup --system nscd
/usr/sbin/adduser --system --no-create-home \
                  --shell /bin/false --ingroup nscd \
                  --disabled-password --disabled-login \
                  --gecos "Name Service Caching Daemon" nscd
chown nscd:nscd /var/run/nscd/nscd.pid || true

vim /etc/nscd.conf
```

```
...
####fx:
#So'lo usamos logfile si asignamos un valor >0 a debug-level
#logfile                 /var/log/nscd.log
####endfx
...
####fx:
server-user             nscd
####endfx
...
```

```
invoke-rc.d nscd start
grep ^[^#]enable-cache /etc/nscd.conf
```

```
 enable-cache            passwd          yes
 enable-cache            group           yes
 enable-cache            hosts           no
 enable-cache            services        yes
```

... nscd cacheará las consultas de NSS para los grupos passwd group y services.

Test libnss-ldapd

Vamos a preguntar al SO por los metadatos de las cuentas root (local) y umea (centralizada) utilizando las utilidades getent e id. Ambas consultan al sistema NSS y nos devuelven la respuesta; en el caso de getent puede preguntar por todos los metadatos de algún grupo NSS como eran "passwd" o "group". Gracias al cacheador, la segunda resolución (consultando con el programa "id") debiera ser más rápida que la primera:

```
getent passwd root umea
```

```
root:x:0:0:root:/root:/bin/bash
umea:x:31000:31000:umea casafx user:/afs/casafx.dyndns.org/user/u/um/umea:/bin/bash
```

```
getent group  root umea
```

```
root:x:0:
umea:*:31000:
```

```
id umea
```

```
uid=31000(umea) gid=31000(umea) groups=31000(umea)
```

1.4.4. Instalación del soporte KERBEROS para el sistema PAM

Una vez que funcionan los mecanismo de acceso a metadatos, ya podemos integrarlo en el log-in del sistema.

```
cp -a /etc/pam.d /etc/pam.d_preKRB5


apt-get install libpam-krb5 libpam-afs-session
```

```
-"PAM profiles to enable:"
Kerberos authentication, Unix authentication, AFS session management
```

- (puede que aparezcan otros como Ccreds caching o LDAP authentication, pero no los elegimos y más adelante se discutirá parcialmente su utilidad).

Se impone la necesidad de describir cómo se configura PAM:

- libpam-runtime[7] es un paquete instalado en el sistema y que proporciona la infraestructura básica para que configurar un esquema u otro de autentificación sea automático en Debian, sin necesidad de editar los ficheros de configuración de PAM interactivamente. Para conseguir ese objetivo, cada paquete que usa PAM instala un fichero en /usr/share/pam-configs/. En nuestro caso, se acaban de instalar /usr/share/pam-configs/krb5,afs-session.

- Estos ficheros contienen una configuración básica de cada paquete más información adicional para integrarlo con otros (orden, prioridades, etc). Por ejemplo, cuando debconf nos ha preguntado qué plantillas activar y le hemos indicado que todas, ha llamado a la utilidad pam-auth-update y, usando las 3 plantillas correspondientes, ha construído los ficheros de configuración finales, los que realmente leen las librerías del sistema PAM. Estos últimos se crean en /etc/pam.d/ y son "common-auth", "common-account", "common-password", "common-session".

- Cada fichero en /etc/pam.d/ contiene líneas que se interpretan secuencialmente. Podemos analizar su formato:

  ```
  <module-type> <control> <module-path> <arguments>
  ```

  - <module-type>es la tarea (account, auth, password o session) que se pretende configurar con esa línea. Efectivamente, para utilizar el recurso de cuentas de shell, hace falta completar una o varias tareas, y cada una tendrá su módulo: tarea de comprobación de la identidad del usuario (módulo auth), o también tareas de autorización de la cuenta (módulo account), o también tareas que planificar antes de usar el servicio y tras dejarlo libre (módulo session); finalmente el módulo password ayuda a que el usuario pueda cambiar sus credenciales (su contraseña, por ejemplo).
  - <control>del secuenciador; indican que si no se cumple lo que expone la línea, debe fallar la secuencia completa (valor requisite), o puede seguir pero anunciando el error (valor required), o que si sí se cumple ya no es necesario seguir el resto de la secuencia para completar un éxito (sufficient), o que no interfiere (optional); existe una segunda sintaxis [code1=action1 ...] que aporta más precisión al mismo tipo de funcionalidad.
  - <module-path>es el nombre de la librería (absoluto o relativo a /lib/security/) que implementa la funcionalidad de esta tarea, por ejemplo: pam_krb5.so, instalada por el paquete libpam-krb5.
  - <arguments>son los posibles parámetros aceptados por el módulo.
  - Anotación:
    - PAM permite la inclusión de otros ficheros con "substack" e "include" (en Debian es posible llamarlo así: @include file-path). Este mecanismo es el

[7] https://wiki.ubuntu.com/PAMConfigFrameworkSpec

que hace que los ficheros common-<moduletype>actúen como configuración por defecto, ya que si un paquete depende de PAM, puede dejar un fichero (de nombre el de la aplicación o el servicio de internet que ofrece) en /etc/pam.d/ cuyo contenido sea simplemente include's a los ficheros common-<moduletype>que necesite.

- Nota, la autentificación con el KERBEROS AS y la expedición de un determinado tipo de TGT es un proceso configurable en parte aquí, /etc/pam.d/*, en parte en /etc/krb5.conf (por ejemplo, el carácter "forwardable" de los tickets está declarado allí). La fuente de un problema puede estar en ambos lugares, pues.

Si los ficheros anteriores de /etc/pam.d hubiesen sido modificados localmente por el administrador con anterioridad, pam-auth-update se negará a procesarlos y mostrará un mensaje al respecto. En ese caso podemos a continuación ejecutar directamente a pam-auth-update cambiando ese comportamiento en línea de comandos con: "pam-auth-update –force". Volverán a aparecer las preguntas de debconf.

El objetivo final de esta somera introducción a la configuración de PAM, es comprender cómo funcionará el log-in con la configuración por defecto que provee el sistema libpam-runtime. En este despliegue puede ser útil, en otros no y por ello se darán algunas indicaciones.

Veamos ya la configuración de PAM actual. Ante cualquier duda, puede consultarse offline "man pam.d", "man pam_unix", "man pam_krb5", "man pam_afs_session". Ante problemas, añádase el parámetro debug a los módulos que lo soportan, y vigílese /var/log/auth. Véase también "man pam_warn".

Task "auth": autenticación/autorización del usuario

El fichero common-auth podría leerse como sigue:

Prueba con el KERBEROS AS sin permitir autentificación exitosa a identidades con un uid inferior a 1000; no le devuelvas a la aplicación tu estado de salida, si hay éxito salta 2 líneas, si no hay éxito prueba autentificación unix (/etc/passwd|shadow). Obtén credenciales AFS.

```
view /etc/pam.d/common-auth
```

```
auth [success=2 default=ignore] pam_krb5.so minimum_uid=1000
auth [success=1 default=ignore] pam_unix.so nullok_secure try_first_pass
auth requisite                  pam_deny.so
auth required                   pam_permit.so
auth optional                   pam_afs_session.so
```

Task "account": verificación de la cuenta a la que se pretende acceder.

Si los metadatos del grupo shadow resultab en éxito, salta una línea y permite probar después la tarea account de krb5 (básicamente hacer la misma comprobación que al auth). Nosotros no tenemos una cuenta umea local con metadatos en /etc/shadow, pero sí tenemos en LDAP una cuenta uid=umea que usa la clase shadowAccount (aunque no dé valor a ninguno de sus atributos, basta con que la use porque si no fallará, y ésta es la razón por la que se puso en /etc/nsswitch.conf que el grupo "shadow" se consultase también en LDAP: no para la tarea de autentificación sino para que funcione esta configuración por defecto para la tarea de account de PAM).

```
view /etc/pam.d/common-account
```

```
account [success=1 new_authtok_reqd=done default=ignore] pam_unix.so
account requisite                                        pam_deny.so
account required                                         pam_permit.so
account required                                         pam_krb5.so minimum_uid=1000
```

Task "session": tareas requeridas antes de garantizar el servicio y después de retirarlo.

Básicamente crea o destruye la caché para el tgt, informa que se ha producido el log-in o el log-out usando syslog, crea o destruye el token AFS.

- Anotación: existe también un /etc/pam.d/common-session-noninteractive idéntico al que viene a continuación y poco reelevante.

```
view /etc/pam.d/common-session
```

```
session     [default=1]                  pam_permit.so
session     requisite                    pam_deny.so
session     required                     pam_permit.so
session     optional                     pam_krb5.so minimum_uid=1000
session     required                     pam_unix.so
session     optional                     pam_afs_session.so
```

Task "password": tareas de actualización del mecanismo de autenticación.

Esta configuración pide el pass actual y tiene como condición que exista la cuenta en /etc/passwd, de forma que el pass sea el mismo en local y central. Por tanto, un usuario sólo centralizado no puede cambiar su pass en KERBEROS. Ésto puede ser aceptable como configuración por defecto, y no impide modificar /etc/pam.d/passwd para que, en lugar de incluir este archivo, funcione de otra manera (passwd es una utilidad estándar, en modo texto, para cambiar la contraseña de una cuenta).

```
view /etc/pam.d/common-password
```

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

```
password [success=2 default=ignore]    pam_krb5.so minimum_uid=1000
password [success=1 default=ignore]    pam_unix.so obscure use_authtok \
try_first_pass sha512
password requisite                     pam_deny.so
password required                      pam_permit.so
```

Y hasta aquí la configuración de PAM a través del sistema libpam-runtime de Debian. Opcionalmente:

```
invoke-rc.d <servicio_que_usa_PAM> restart  # No debiera ser necesario, pero
                                            # se puede efectuar.
```

Configuración de PAM sin libpam-runtime

```
dpkg-reconfigure libpam-runtime
```

```
 PAM profiles to enable: Unix authentication
```

Ésto deja los ficheros /etc/pam.d/common-* tal como estaban antes de que instalásemos libpam-krb5 y libpam-afs-session. Ya podemos editarlos según convenga y, una vez hechos cambios a mano, libpam-runtime lo detectará y no volverá a llamar a pam-auth-update para regenerarlos a partir de las plantillas en /usr/share/pam-configs. Las alternativas que presentamos a continuación se comportan parecido que la configuración por defecto que deja libpam-krb5, si bien algunas diferencias especialmente relevantes:

- La autenticación ofrece nuevas posibilidades como la de intentar casar una contraseña tanto en KERBEROS AS como en /etc/shadow.
- Esta vez sí se puede cambiar el password a través de PAM aunque la cuenta sea sólo centralizada.

Veamos:

```
cp -a /etc/pam.d /etc/pam.d_preKRB5noruntime
```

- Task "auth": autentificación/autorización del usuario:

```
vim /etc/pam.d/common-auth
```

```
####fx:
##Krb + shadow auth: Si pass en ambas db =, entonces 1er prompt shadow,
#                    si en blanco, 2o prompt krb.
#                    Si pass en ambas db !=,  entonces 1er prompt intenta
#                    casar con uno u otro.
#                    Si *K* en shadow, el 1er prompt es el de krb
#                    pues pass en ambas db = y quieres krb.
auth    sufficient          pam_unix.so nullok_secure
auth    sufficient          pam_krb5.so use_first_pass minimum_uid=30000
#si afs kerberizado para homes, tb anado la inmediatamente siguiente:
auth    optional            pam_afs_session.so program=/usr/bin/aklog
auth    required            pam_deny.so
####endfx
```

- Task "account": verificación de la cuenta a la que la identidad pretende acceder.

```
vim /etc/pam.d/common-account
```

```
####fx:
account sufficient        pam_unix.so
account sufficient        pam_krb5.so minimum_uid=30000
account required          pam_deny.so
##endfx
```

- Task "password": tareas de actualización del mecanismo de autenticación.

```
vim /etc/pam.d/common-password
```

```
####fx:
password  sufficient   pam_unix.so nullok obscure md5
password  sufficient   pam_krb5.so use_first_pass minimum_uid=30000
password  required     pam_deny.so
####endfx
```

- Task "session": tareas requeridas antes de garantizar el servicio y después de retirarlo.

```
vim /etc/pam.d/common-session
```

```
####fx:
session required pam_limits.so
session optional pam_krb5.so minimum_uid=30000
session optional pam_unix.so
#Con AFS kerberizado para homes, tambie'n an~adimos la siguiente:
session optional pam_afs_session.so program=/usr/bin/aklog
#y suponemos que podemos an~adir (nota: umask=0022 es igual a mask=0644)
session required pam_mkhomedir.so skel=/etc/skel/ umask=0022
# Nota: en debian existe tambie'n un /etc/pam.d/common-session-noninteractive,
#       pero no se relaciona con servicios de cuentas de shell POSIX.
####endfx
```

Bien a través de libpam-runtime o la configuración alternativa que acabamos de exponer, tenemos una muestra de cómo entender PAM. Si la primera opción nos es válida, ahorramos en tiempo de administración; en otro caso, siempre podemos editar a mano y utilizar las amplias posibilidades que ofrece PAM.

1.4.5. No instalación de cacheador de credenciales en dklab1 y dklab2

Efectivamente, nscd hizo de cacheador de los metadatos de cuentas uid, gid etc. Cuando las cuentas están centralizadas, también puede ser conveniente instalar un software cacheador de credenciales (por ejemplo libpam-ccreds).

Sin embargo, dklab1 es una máquina que sólo participa de nuestra organización, la cual utiliza a KERBEROS como protocolo específico de autenticación y, como sabemos, tiene su propio sistema de caché de credenciales (al igual que los token AFS). Así, libpam-ccreds no aporta ni caché ante ciclos desconexión-conexión, ni aumenta la celeridad de las autenticaciones ni, en definitiva, nada. Se suele usar cuando las cuentas están centralizadas y utilizan no KERBEROS sino LDAP para autenticaciones.

Aún así, esta pequeña discusión sobre la caché de credenciales nos hace pasar cerca de un asunto relacionado. Al contrario que dklab1, una computadora portátil de un usuario de nuestra organización puede moverse por distintas redes cada una con su esquema de autenticación. Una de esas redes será la nuestra y estará basada en KERBEROS, otra puede estarlo en LDAP y una tercera red quizás no ofrezca cuentas centralizadas.

Por tanto, para los clientes hipotéticos de esta infraestructura como dispositivos portátiles, sí será conveniente tener un sistema más dinámico con posibilidad de utilizar varios

perfiles (algunos con caché ante desconexiones, aludiendo a la discusión que nos trajo aquí) y se optará por una solución más robusta: libpam-sss y su servicio acompañante sssd [8], los cuales sustituyen a libpam-ldapd, nslcd y libpam-ccreds. En la sección sobre los hipotéticos clientes de nuestra infraestructura se desarrollará este asunto.

1.4.6. Test log-in

Preliminarmente comprobamos que NSS, el principal KERBEROS y la identidad AFS son usables:

```
id umea

kinit umea; aklog

klist

tokens


unlog; kdestroy

tail /var/log/auth.log
```

Utilizamos la utilidad "login" para probar PAM; ésta permite al usuario utilizar el servicio de cuentas de shell POSIX en modo texto[9]; login utiliza PAM y configura su comportamiento en el /etc/pam.d/login, el cual, básicamente, hace include's a los ficheros /etc/pam.d/common-* donde reside nuestra configuración.

```
login umea
```

[8] http://people.skolelinux.org/pere/blog/Caching_password__user_and_group_on_a_roaming_Debian_laptop
[9] IEEE P1003.2

```
pwd
ls
klist; tokens

#Antes de salir, podemos copiar algunos ficheros ba'sicos a nuestro home:
SKELDIR=/etc/skel
USUARIO=`id -n -u`
GRUPO=`id -n -g`
HOMEDIR=${HOME}
cd /etc/skel;
for i in `find . -maxdepth 1 -type d|grep \/`; do
 cp -dr --no-preserve=ownership $i ~/
done
cd ~
for i in `ls -A ${SKELDIR}`
    do install -o ${USUARIO} -g ${GRUPO} -m 0664 ${SKELDIR}/${i} ${HOMEDIR}
done
echo "export HISTSIZE=10000000; export HISTFILESIZE=100000000" >> .bashrc
ls -A

exit
```

Llegados a este punto, nos podemos preguntar qué programas en Debian utilizan PAM de la misma manera que lo ha hecho "login"; ésto es posible comprobarlo con:

```
apt-cache rdepends libpam0g


# Ejemplos ti'picos:
apt-cache rdepends libpam0g | egrep '(login|dm$|su)'
```

1.4.7. SSH

Log-in SSH utilizando KERBEROS AS (entrada al sistema)

No es infrecuente que, por cualquier causa, el log-in en el sistema se produzca remotamente usando protocolos como SSH. Es decir el usuario utiliza un cliente SSH desde alguna máquina ajena a la organización, y el servidor SSH en, por ejemplo, dklab1 media el log-in en el sistema. Buscamos entonces que ese proceso especial de log-in se comporte también como el desarrollado hasta ahora.

Efectivamente el servidor sshd está configurado para usar PAM, luego ya está todo hecho:

```
grep UsePAM /etc/ssh/sshd_config
```

```
UsePAM yes
```

Por tanto, usará su configuración en /etc/pam.d/sshd que básicamente hace include's a nuestra configuración en /etc/pam.d/common-*; PAM además utiliza como se expuso a NSS, que está configurado convenientemente para obtener los metadatos en LDAP, etc.

No hay, pues, que configurar nada en dklab1, dklab2, o cualquier máquina hipotética configurada para usar nuestra infraestructura. Si sshd está instalado con "UsePAM yes", se puede usar ya al cliente SSH.

Clientes SSH típicos serían:

- ssh (proyecto OpenSSH) suele encontrarse preinstalado en casi cualquier SO POSIX; en Windows se puede encontrar igualmente[10] (incluso, sobre cygwin, existen instaladores con el par cliente/servidor[11]); no obstante en Windows se recomienda putty.
- putty (proyecto Putty), es un cliente SSH más emulador gráfico de terminal, creado para Windows y portado a SO POSIX. Las opciones pueden pasársele en un panel gráfico de configuración o en la línea de comando, soportando en este caso una interfaz común a la del cliente de openssh (-l, -X ...).

Supongamos que el usuario umea quiere entrar en el dklab1 desde una máquina cualquiera utilizando SSH, entonces haría:

```
maquinacualquiera> ssh -l umea dklab1.casafx.dyndns.org.
```

- *Anotación: "maquinacualquiera" puede ser dklab1 también, sin problemas podemos hacer log-in ssh en la misma máquina por motivos de comprobaciones.*

[10] http://es.sourceforge.jp/projects/sfnet_mingw/downloads/MSYS/BaseSystem/msys-core/msys-1.0.11/MSYS-1.0.11.exe/

[11] http://www.itefix.no/i2/copssh

```
umea@dklab1.casafx.dyndns.org's password:
umeapa55
```

```
hostname --fqdn
pwd; klist; tokens;
# mostrar'ia que estamos en nuestro home bajo /afs/ y que
# al entrar fueron expedidos efectivamente el TGT KERBEROS
# y el token AFS.
exit
```

- Anotación: Alternativa a usePAM de OpenSSH. OpenSSH lo desarrolla gente del proyecto OpenBSD, SO que no utiliza PAM; por este motivo existe implementada la siguiente alternativa, menos flexible pero suficiente, en que sshd se encarga de todo el proceso; en algunos casos puede ser útil, y es trivial su configuración:

```
vim /etc/ssh/sshd_config
```

```
####fx:
#-usePAM yes
KerberosAuthentication yes    #Hace auth en el kerberos AS (lo autodescubre)
KerberosOrLocalPasswd yes     #Alternativamente, utiliza /etc/shadow.
KerberosGetAFSToken yes       #Consigue token AFS tras krb5 TGT... pero tiene
                              # un bug en Debian 6 y 7:
                              # http://bugs.debian.org/607238
KerberosTicketCleanup yes     #Borra las anteriores credenciales al logout
####endfx
```

Log-in SSH aprovechando funcionalidad SSO (cambio de máquina)

Supongamos que el usuario umea ha hecho log-in (por ejemplo utilizando el programa login, o conectándose a través del acceso mediado por sshd que se acaba de exponer). Trabajando en su máquina, llega el momento de ejecutar software que consume cuantiosos recursos de tiempo de CPU, memoria RAM, ambos... El usuario puede hábilmente decidir

conectarse por ssh a una máquina más potente y ejecutarlos allí. Nos preguntamos ahora si, puesto que ya hicimos un log-in en el que se expidió un TGT KERBEROS, podríamos usar ese ticket para la fase de autenticación de este segundo log-in, gracias al esquema SSO que provee KERBEROS, y conseguir el servicio sin que se nos interrumpa para pedir la contraseña, etc.

Efectivamente, sshd implementa también autenticación KERBEROS usando GSS-API. Además de permitir su uso desde el archivo de configuración, debemos completar los pasos típicos a la kerberización de cualquier servicio: crear principal del servicio, exportar su llave, indicar dónde está.

Principal para sshd en dklab1 El formato para el servicio ssh es host/<FQDN>@<REALM>. Es habitual dejar exportar este principal a /etc/krb5.keytab, donde sshd lo buscará por defecto. Así pues, no lo exportamos a /etc/keytab.d/openssh.keytab como venía siendo nuestro política de keytabs.

```
kadmin -p root/admin
> addprinc -policy service \
           -randkey host/dklab1.casafx.dyndns.org@CASAFX.DYNDNS.ORG
> ktadd -k /etc/krb5.keytab host/dklab1.casafx.dyndns.org@CASAFX.DYNDNS.ORG
```

Configuración de sshd Editamos /etc/ssh/sshd_config:

```
vim /etc/ssh/sshd_config
```

```
...
####fx:
GSSAPIAuthentication yes       # Ofrecera' gssapi auth
GSSAPICleanupCredentials yes   # Aseguramos que hara' kdestroy;unlog al log-out
####endfx
```

```
invoke-rc.d ssh reload
```

Configuración del cliente SSH ssh Le indicamos que utilice, cuando se le ofrezca, GSSAPI. Hay varias formas de indicar ésto:

- En línea de comandos: ssh dispone de -o "opción" donde las opciones relevantes ahora son GSSAPIAuthentication, PreferredAuthentications, GSSAPIDelegateCre-

dentials. En debian, la única opción realmente necesaria, dados los valores por defecto, es "GSSAPIDelegateCredentials yes" para que utilice el token AFS conseguido en la primera máquina, también en la segunda.

```
maquinacualquiera# login umea
maquinacualquiera$ ssh -v -o "GSSAPIDelegateCredentials yes" \
                        umea@dklab1.casafx.dyndns.org.
```

- *"maquinacualquiera" puede ser dklab1 también.*

```
...
debug1: Authentications that can continue:
        publickey,gssapi-keyex,gssapi-with-mic,password
#...es decir, el servidor ofrece todos esos me'todos, y el cliente
#   prefiere gssapi:
debug1: Next authentication method: gssapi-with-mic
...
```

```
hostname --fqdn
pwd; klist; tokens
# de nuevo, mostrar'ia que estamos en el home de umea bajo /afs/ y que
# al entrar fueron transferidas las credenciales TGT KERBEROS
# y el token AFS para poder utilizarlas tambie'n aqui'.
exit
```

- Configurando ~/.ssh/config, es decir, el usuario umea podría:

```
login umea
```

```
mkdir -p .ssh
vim .ssh/config
```

```
####fx:
Host dklab1
 HostName dklab1.casafx.dyndns.org.
 User umea
 GSSAPIDelegateCredentials yes
 ForwardX11 yes
####endfx
```

```
history -a # si queremos conservar el historial de comandos tras ssh
           # almacenamos lo que llevamos introducido en esta sesio'n
ssh dklab1
# mismo resultado que el caso de los flags en li'nea de comandos
exit
```

- Configurando globalmente /etc/ssh/ssh_config. En dklab1 nos parece aceptable, así:

```
vim /etc/ssh/ssh_config
```

```
...
GSSAPIAuthentication yes
...
####fx:
#-GSSAPIDelegateCredentials no
GSSAPIDelegateCredentials yes
####endfx
```

```
login umea
```

```
ssh dklab1
# mismo resultado que el caso de los flags en li'nea de comandos
exit
```

Configuración del cliente SSH putty Putty también soporta GSSAPI, tanto la versión 0.58 modificada del 2005[12] como la últimas versiones oficiales (hasta hace poco con bugs, por éso se ha indicado también la del 2005).

La configuración se hace gráficamente (y no parecen existir flags en CLI), concretamente en el nodo:

```
Connection -> SSH -> Auth -> GSSAPI
```

Anotación Queremos enfatizar el resultado práctico que se consigue en nuestro despliegue al cambiar de máquina con SSH:

Gracias a la infraestructura desplegada en capítulos anteriores, la cuenta está centralizada, el log-in es único y el sistema de archivos es ubicuo con un espacio de nombres invariable. Por ello, tras ejecutar ssh: *todo queda igual.*

Efectivamente, si se mantiene la identidad, la autenticación pasada y el árbol de ficheros, sólo cambian los recursos de la máquina (CPU, RAM, retardo de red...).

Posibilidades para el usuario

SSH se concibió como un protocolo para ofrecer una sesión (remota y segura) en modo texto, que permite también la transferencia de archivos o incluso la creación de túneles y vpn's. Además, el usuario puede salir del modo texto y ejecutar aplicaciones multimedia si se pasa a ssh o putty el flag -X y se dan las siguientes circunstancias:

- El SO origen (donde se ejecuta ssh o putty) dispone de un servidor gráfico (en SO's POSIX Xorg[13] -o su antecesor Xfree86[14]-, en Windows el porte Xming[15]).
- El SO origen (donde se ejecuta ssh o putty) disponde de un servidor de sonido (como Pulseaudio[16]).
- El SO destino tiene una aplicación con salida gráfica (como un visor PDF).
- El SO destino tiene una aplicación con salida sonora (como un reproductor de música).

[12] http://ftp.cs.stanford.edu/pub/Windows/putty/
[13] http://www.x.org/wiki/
[14] http://www.xfree86.org/
[15] http://sourceforge.net/projects/xming/
[16] http://www.pulseaudio.org/wiki/AboutPulseAudio#SupportedOperatingSystems

Entonces, el visor PDF ejecutado en la sesión SSH puede mandar su salida gráfica al servidor gráfico del SO origen. También el reproductor de música ejecutado en la sesión SSH puede mandar su salida sonora al servidor de sonido en el SO origen. Internamente, la aplicación gráfica busca su servidor gráfico en la variable de entorno DISPLAY, que ssh/putty (gracias al flag -X) ajusta convenientemente y tuneliza hacia el SO origen; por su lado la aplicación sonora se ha de configurar para que utilice pulseaudio como subsistema de salida y, entonces, volverá a utilizar la variable DISPLAY que le lleva al SO origen y allí preguntará al servidor gráfico por el servidor sonoro local: efectivamente una extensión de Xorg permite a un servidor sonoro registrarse en éste para facilitar su localización.

No podemos dedicar más tiempo a estas cuestiones, pero queda claro que el log-in remoto usando SSH, aún no siendo la primera opción, puede ser una opción bastante completa.

1.5. DKLAB2

El despliegue en dklab2 puede ser idéntico al mostrado en dklab1.

1.5.1. Instalación de soporte LDAP para el sistema NSS

```
apt-get install --no-install-recommends libnss-ldapd nslcd
```

```
-"LDAP server URI:"
DNS
-"LDAP server search base:"
ou=accounts,dc=casafx,dc=dyndns,dc=org
- "LDAP authentication to use"
None
-"Use StartTLS?"
No
-"Name services to configure (libnss-ldapd):"
group, passwd, shadow
```

```
ls /lib/i386-linux-gnu/libnss_ldap*
```

Gracias a las preguntas de debconf, nslcd está configurado:

```
view /etc/nslcd.conf
```

1.5.2. Configuración del orden de resolución de metadatos en NSS

Ya hecho por debconf; podemos revisarlo con:

```
view /etc/nsswitch.conf
```

1.5.3. Instalación del cacheador nscd; test

Instalamos, creamos una cuenta local no privilegiada que declaramos en su fichero de configuración:

```
apt-get install nscd


invoke-rc.d nscd stop
/usr/sbin/addgroup --system nscd
/usr/sbin/adduser --system --no-create-home --shell /bin/false --ingroup nscd \
                  --disabled-password --disabled-login \
                  --gecos "Name Service Caching Daemon" nscd
chown nscd:nscd /var/run/nscd/nscd.pid
```

```
vim /etc/nscd.conf
```

```
...
####fx:
#So'lo usamos logfile si asignamos un valor >0 a debug-level
#logfile                 /var/log/nscd.log
####endfx
...
####fx:
server-user             nscd
####endfx
...
```

```
invoke-rc.d nscd restart
```

A partir de ahora, cacheará las consultas de NSS para los grupos passwd group y services:

```
grep ^[^#]enable-cache /etc/nscd.conf
```

Test libnss-ldapd

Comprobamos que utilidades que hacen uso del sistema NSS, funcionan correctamente y que la primera resolución es más lenta que las posteriores gracias al cacheador.

```
getent passwd root umea
```

```
root:x:0:0:root:/root:/bin/bash
umea:x:31000:31000:umea casafx user:/afs/casafx.dyndns.org/user/u/um/umea:/bin/bash
```

```
getent group  root umea
```

```
root:x:0:
umea:*:31000:
```

```
id umea
```

```
uid=31000(umea) gid=31000(umea) groups=31000(umea)
```

1.5.4. Instalación del soporte KERBEROS para el sistema PAM

Una vez que funcionan los mecanismo de acceso a metadatos, ya podemos integrarlo en el log-in del sistema.

```
cp -a /etc/pam.d /etc/pam.d_preKRB5


apt-get install libpam-krb5 libpam-afs-session
```

```
-"PAM profiles to enable:"
Kerberos authentication, Unix authentication, AFS session management
```

Para una configuración alternativa sin libpam-runtime, revísense las indicaciones y posibilidades que se mostraron en dklab1.

1.5.5. No instalación del cacheador de credenciales en dklab1 y dklab2

KERBEROS (y AFS) ya tienen su propio sistema caché y no es necesario.

1.5.6. Test log-in

Primero comprobamos que NSS, el principal KERBEROS y la identidad AFS son usables:

```
id umea
kinit umea; aklog


klist; tokens


unlog; kdestroy
tail /var/log/auth.log
```

Utilizábamos la utilidad "login" para probar PAM; ésta permite al usuario utilizar el servicio de cuentas de shell POSIX en modo texto[17]; login utiliza PAM y configura su comportamiento en el /etc/pam.d/login, el cual, básicamente, hace include's a los ficheros /etc/pam.d/common-* donde reside nuestra configuración.

```
login umea
```

```
pwd
id
klist; tokens


ls -A
cat .bashrc


exit
```

1.5.7. SSH

Login SSH utilizando KERBEROS AS (entrada al sistema)

El servidor sshd ya está plenamente operativo puesto que usa PAM. OpenSSH ofrecía una alternativa a PAM para conseguir parecida funcionalidad, consúltese lo que se expuso en dklab1.

[17] IEEE P1003.2

Log-in SSH aprovechando funcionalidad SSO (cambio de máquina)

Principal para sshd en dklab2 Creamos y exportamos el principal host/dklab2.casafx.dyndns.org

```
kadmin -p root/admin
> addprinc -policy service \
           -randkey host/dklab2.casafx.dyndns.org@CASAFX.DYNDNS.ORG
> ktadd -k /etc/krb5.keytab host/dklab2.casafx.dyndns.org@CASAFX.DYNDNS.ORG
```

Configuración de sshd Hacíamos que sshd ofrezca autenticación KERBEROS:

```
vim /etc/ssh/sshd_config
```

```
...
####fx:
GSSAPIAuthentication yes       # Ofrecera' gssapi auth
GSSAPICleanupCredentials yes   # Hara' kdestroy; unlog al logout
####endfx
```

```
invoke-rc.d ssh reload
```

Configuración del cliente SSH ssh Nos quedamos con el último escenario que vimos en dklab1, hacer que por defecto usase autenticación KERBEROS y delegación de credenciales.

```
vim /etc/ssh/ssh_config
```

```
...
GSSAPIAuthentication yes
...
####fx:
#-GSSAPIDelegateCredentials no
GSSAPIDelegateCredentials yes
####endfx
```

```
login umea
```

```
ssh -l umea dklab2.casafx.dyndns.org.
# ... entra sin mostrar prompt alguno.
hostname --fqdn
pwd; klist; tokens
exit
```

1.6. Nueva máquina en nuestro parque informático: despliegue en máquinas clientes de nuestra infraestructura

Supongamos que una nueva máquina es incorporada a nuestro parque informático. Para ser útil precisa ser configurada como cliente de toda la infraestructura desplegada.

- Anotación: suponemos que la nueva máquina va a hacer de cliente porque es la situación que nos queda por analizar. Evidentemente, si por el contrario lo que queremos es que la nueva máquina haga de servidor y ayude a reducir la carga de trabajo en dklab1 y dklab2, simplemente debemos proceder como se ha hecho hasta ahora, y se hará después para otros servicios, con dklab2. En concreto, será así siempre que las bases de datos utilizadas soporten un esquema Multi-Máster de replicación y los servicios pueden ser encontrados a través de registros DNS SRV RR; si se cumplen estas premisas, el rol de servidor es sistemáticamente idéntico en todas las máquinas.

- Anotación: las indicaciones de instalación del sofware cliente de todos los servicios *finales* -Pidgin para IM, VoIP; Mozilla Thunderbird para E-Mail- son absolutamente triviales (la configuración de esos clientes sí que no es trivial, pero ésta reside en AFS, luego no es pues asunto de las máquinas con rol de cliente, en verdad). Por su lado, las indicaciones del resto del software cliente -clientes de servicios no finales: cliente MIT Kerberos, cliente OpenAFS etc- no son tan triviales, son del orden de dificultad que hemos venido palpando al instalar la parte de cliente de dklab1 y dklab2. Incidimos una vez más, es preferible tratarlas ahora todas juntas a cuando se hable del despliegue de esos servicios del lado del servidor.

1.6.1. Esquema general

Sea cual sea el equipo, debemos desplegar los siguientes componentes:

1. Cliente OpenVPN.
2. Cliente DNS.
3. Cliente NTP.
4. Cliente KERBEROS.
5. Cliente AFS.
6. Sistema NSS o equivalente para conseguir metadatos en LDAP.
7. Sistema PAM o equivalente para autenticación ante el KERBEROS AS y expedición del TGT KERBEROS y el token AFS.
8. Instalación de clientes para el resto de servicios (IM, VoIP, E-mail).

Supondremos que el SO de la máquina sólo puede ser o bien GNU/Linux (nos centramos en Debian 7 de nuevo), o bien Windows, o bien otros POSIX (MAC OSX etc). En cada caso, y con distinto grado de profundidad, procuramos ahora dar las claves de cada punto del esquema anterior.

1.6.2. GNU/Linux (Debian 7 "Wheezy")

Evidentemente, el despliegue debiera ser casi idéntico a como se hizo en dklab1 y dklab2 para su rol de clientes. Sin embargo, siempre que se pueda, se harán modificaciones para adaptar opciones o instalar software alternativo orientados a máquinas portátiles que tiendan a establecerse nómadamente en distintos tipos de redes[18].

[18]Podemos adelantar que con dispositivos portátiles no nos referimos a smart-phones o tablets: Si tomamos a Android como referencia, OpenAFS no está portado y KERBEROS no lo estaba hasta muy recientemente, ref. Por otro lado Dalvik (Android) utiliza otros sistemas de resolución de metadatos y autenticación diferentes de NSS y PAM por lo que aunque se pudiese utilizar un servicio KERBERIZADO, no está claro que se pudiese integrar en el log-in como se hará aquí. OpenVPN sí está portado a Android.

- Anotación sobre otras distribuciones de Linux: la forma de instalar será distinta y la de configurar puede variar algo, el software debiera ser el mismo.

Cliente OpenVPN

```
apt-get install openvpn


# Podemos aprovechar tambie'n para asegurarnos la instalacio'n de OpenSSH:
apt-get install openssh-client sshfs
```

A continuación se deberían seguir las siguientes secciones del capítulo sobre la VPN:

- Certificados para el rol de OpenVPN tls-client. Además en la máquina cliente creamos el directorio

  ```
  mkdir /etc/openvpn/ovpn-keys-casafx.dyndns.org
  ```

  Entonces transferimos (por ejemplo utilizando un pendrive, SCP u otro medio seguro) el material criptográfico (<host>.casafx.dyndns.org.{key,p12,crt}, ca.crt) desde dklab1 hasta ese directorio (sólo root debería poder leer el .key y .p12.

- Configuración de OpenVPN en modo tls-client. Adaptamos la configuración al nombre de la máquina cliente y la almacenamos en /etc/openvpn/casafx-tls-client.conf

- Configuración del OpenVPN en modo tls-server, es decir en dklab1 y, para que la máquina cliente reciba una IP fija a través del sistema CCD, hemos de seguir el último párrafo de esa sección.

Cliente DNS

Consistía en configurar el fichero /etc/resolv.conf, idealmente gracias al flexible resolvconf:

```
apt-get install resolvconf
```

```
-"Prepare /etc/resolv.conf for dynamic updates?"
Yes
-"Append original file to dynamic file?"
No
```

A continuación se utilizarían en las definiciones de las redes, las opciones que son posteriormente utilizadas por resolvconf:

```
vim /etc/network/interfaces
```

```
...
iface <iface-name> inet manual # manual, a no ser que se usase dhcp.
 ...
 dns-options rotate             # uso round-robin de los servidores DNS
 dns-nameservers 10.168.1.1 10.168.1.2
 dns-search casafx.dyndns.org.
...
```

Puede ser conveniente, especialmente para portátiles, permitir búsquedas utilizando Multicast-DNS. Estamos entonces expresando que queremos una nueva fuente de nombres de dominio junto a DNS y /etc/hosts, editamos por tanto el nsswitch.conf:

```
vim /etc/nsswitch.conf
```

```
...
####fx:
#
#-hosts:          files dns
hosts:          files mdns4_minimal [NOTFOUND=return] dns mdns4
####endfx
...
```

```
invoke-rc.d resolvconf reload
```

Evidentemente, deberíamos plantearnos insertar un registro DNS RR de tipo A para la nueva máquina, de forma que éste tenga su propio nombre en la red. Consúltese en el capítulo sobre DNS, la sección:

- Vista "vpn", resolución directa e inversa. Un ejemplo allí de registro A es el de la máquina nowhere (que era ficticia, precisamente un ejemplo).

Cliente NTP

```
apt-get install ntp
```

Recordemos que dklab1 y dklab2 proveen un esquema de seguridad Autokey v2 IFF que era opcional[19] utilizar por los clientes. Por tanto vamos a ver la configuración simple sin autenticación y la compleja con ella:

Sin Autokey Añadimos como pool a los servidores ntp:

```
vim /etc/ntp.conf
```

```
...
####fx:
pool ntp.casafx.dyndns.org
####endfx
...
restrict -4 default ignore
restrict -6 default ignore
restrict 127.0.0.1
restrict ::1
####fx:
restrict ntp.casafx.dyndns.org notrust nomodify notrap ntpport
restrict 10.168.1.1 notrust nomodify notrap ntpport
restrict 10.168.1.2 notrust nomodify notrap ntpport
####endfx
...
```

```
invoke-rc.d ntp restart
```

Con Autokey Implica la creación de un par de llaves privada y pública (certificado X.509). No implica crear parámetros IFF esta vez, ni el certificado no será "Trust Root"

[19] http://www.linuxcertif.com/man/5/ntp_auth/#IDENTITY_SCHEMES_AND_CRYPTOTYPES_2998h

y, como no pertenecerá al grupo IFF, no se le transfiere la clave de grupo. En definitiva este sistema es un "leaf node" (hoja) en lo que respecta al árbol de conexiones entre ntpd's, no tiene que autentificarse como servidor a un cliente pues no tiene clientes. La llamada a ntp-keygen no lleva los flags -T ni -I:

```
mkdir -p /etc/ntp/crypto

chown ntp:root /etc/ntp/crypto

cd /etc/ntp/crypto

ntp-keygen -H -s <domainname>.casafx.dyndns.org -p passleafnode


chown -R ntp:root /etc/ntp/crypto

chmod -R o-rwx /etc/ntp/crypto


chown ntp:root /etc/ntp.conf

chmod o-rwx /etc/ntp.conf
```

```
vim /etc/ntp.conf
```

```
####fx:
keysdir /etc/ntp/crypto
crypto pw passleafnode
crypto randfile /dev/urandom  # si syslog: crypto_setup:
                              #  random seed file not found
crypto host leafnode.casafx.dyndns.org
#crypto ident casafx.dyndns.orgw  # No se debe usar ident esta vez,
#                                 # pues creeri'a que pertenece al grupo.

pool ntp.casafx.dyndns.org autokey

#Se pueden usar ACL como:
restrict -4 default ignore
restrict -6 default ignore
restrict 127.0.0.1
restrict ::1
restrict ntp.casafx.dyndns.org notrust nomodify notrap ntpport
restrict 10.168.1.1 notrust nomodify notrap ntpport
restrict 10.168.1.2 notrust nomodify notrap ntpport
####endfx
...
```

```
invoke-rc.d ntp restart
```

Cliente KERBEROS

Sígase lo dicho para dklab1:

- Instalación de MIT Kerberos, sólo paquetes krb5-config krb5-user.
- Modificaciones sobre clientes KERBEROS. Al editar /etc/krb5.conf, aunque no lo hemos comprobado, pensamos que en [logging] no deberían aparecer referencias a

kdc o admin-server (tampoco después al definir el rotador de logs), y la sección [dbdefaults] y [dbmodules] deben omitirse.

Cliente AFS

Sígase lo expuesto para dklab1:

- Instalación del soporte en el kernel Linux
  - Con DKMS
- Instalación de OpenAFS, rol cliente.

Sistemas NSS y PAM (despliegue "Roaming Setup" con SSSD)

Como se discutió con anterioridad, en dispositivos portátiles existe un planteamiento alternativo que permite la configuración de diversos perfiles para las distintas redes a las que eventualmente tenga que integrarse. El proyecto SSSD implementa ese esquema "Roaming Setup" y sustituye por tanto a libnss-ldapd, nscd, libnss-krb5, libpam-afs-session, libpam-ccreds.

SSSD provee un conjunto de demonios para administrar el acceso a directorios remotos así como intermediación en los procesos de autenticación, son:

- Librería para NSS
- Librería para PAM
- Funcionalidades de caching para las dos anteriores.
- Servicio que se encarga de las conexiones, sssd.

```
cp -R /etc/pam.d /etc/pam.d_preSSSD
cp /etc/nsswitch.conf /etc/nsswitch.conf_preSSSD
```

Instalación del soporte LDAP para el sistema NSS y KERBEROS/AFS para el sistema PAM; caching en ambos En el caso de Sssd, el soporte para NSS y PAM que necesitamos y la funcionalidad de caching se resuelven instalando el demonio sssd, libnss-sss, libpam-sss y libpam-afs-session. Podemos instalar también las sssd-tools, pero son de gestión de usuarios y no las vamos a usar.

```
# Preliminarmente:
apt-get purge -y nscd libnss-ldapd
apt-get purge -y libpam-ccreds libpam-krb5

# Sssd:
apt-get install sssd libnss-sss libpam-sss libpam-afs-session # sssd-tools

ls /lib/i386-linux-gnu/libnss_sss*
```

```
 -"PAM profiles to enable:"
 SSS authentication, Unix authentication, AFS session management
```

Situación de NSS El script /var/lib/dpkg/info/libnss-sss.postinst de instalación de libpnss-sss se encargó de proponerse a sí mismo como segunda fuente de metadatos para los grupos "passwd", "shadow" y "group" de NSS (también "netgroup").

```
view /etc/nsswitch.conf
```

```
...
####fx:
#-passwd:       compat
#-group:        compat
#-shadow:       compat
passwd:         compat sss
group:          compat sss
shadow:         compat sss
####endfx
...
```

...vamos entonces a instalar el soporte para PAM que provee SSSD.

Situación de PAM libpam-sss y libpam-afs-session utilizan el sistema libpam-runtime para la configuración automática de /etc/pam.d/common-* a través de las plantillas que dejan en /var/lib/pam-configs/. El esquema es muy parecido al que se explicó tras instalar libpam-krb5 en dklab1 y dklab2, inspecciónese con:

```
view -o /etc/pam.d/common-*
```

Configuración de Sssd Vamos a configurar el servicio sssd. Afortunadamente la versión en Debian 7 está mucho más pulida que las anteriores, que resultaron difíciles de configurar debido a la presencia de bugs. La configuración que presentamos a continuación funciona correctamente, y puede servir de ejemplo para otras redes adicionales a la de nuestra organización. Principalmente hemos de declarar información sobre la localización nuestros servidores y su tipo ("proveedor" en terminología de Sssd).

```
cp /usr/share/doc/sssd/examples/sssd-example.conf /etc/sssd/sssd.conf
chown root:root /etc/sssd/sssd.conf
chmod 0600 /etc/sssd/sssd.conf


vim /etc/sssd/sssd.conf
```

```
[sssd]
...
####fx:
#-domains = LOCAL
domains = casafx.dyndns.org
####endfx:
[nss]
####fx:
#podemos o no modificar la caducidad en segundos de la
#cache con entry_cache_timeout = 600
####endfx
```

```
[pam]
...
####fx:
[domain/casafx.dyndns.org]
min_id = 1000
enumerate = true
cache_credentials = true
## Ldap nss
id_provider = ldap
;ldap_uri = _srv_,dklab1.casafx.dyndns.org.,dklab2.casafx.dyndns.org.
ldap_uri = _srv_
ldap_search_base = ou=accounts,dc=casafx,dc=dyndns,dc=org
```

```
## Krb5 pam
auth_provider = krb5
chpass_provider = krb5
;krb5_kdcip = _srv_,dklab2.casafx.dyndns.org.,dklab1.casafx.dyndns.org.
krb5_kdcip = _srv_
krb5_changepw_principal = kadmin/changepw
krb5_realm = CASAFX.DYNDNS.ORG
krb5_ccachedir = /tmp
krb5_ccname_template = FILE:%d/krb5cc_%U
####endfx
```

```
invoke-rc.d sssd start
ps aux | grep sss
```

Se lanzó sssd y, atendiendo a la opción de configuración "services", lanzará a otros demonios:

/usr/lib/sssd/sssd/sssd_nss, sssd_pam y sssd_be. Todos ellos se ejecuta por defecto

como root, queda pendiente la investigación sobre si es posible utilizar una cuenta no privilegiada creada a tal efecto.

En el fichero de configuración sssd.conf se ofrecen ejemplos para Active Directory. Consúltese "man sssd.conf" (junto a "man sssd-krb5" y "man sssd-ldap") como la fuente definitiva para conocer el tipo de autenticación de cuentas que soporta, o los atributos y mapeos que puede recabar de LDAP etc.

- Nota: sssd-ldap soporta SASL-GSSAPI para la autenticación ante el servidor LDAP, al igual que lo soportaba libnss-ldapd tal como se discutió en su momento. Además, también soporta un modo "proxy" para dejar paso a otro módulo NSS, parece ser.

Acciones adicionales al caching: libpam-mklocaluser Cuando un usuario hace log-in la primera vez, un usuario local es creado en /etc/passwd, un grupo primario es creado en /etc/group y un directorio local como home es creado bajo /home. Ésto es útil para cuando el servidor LDAP queda desconectado, para poder seguir utilizando la cuenta. La contraseña al log-in puede ser cacheada por sssd.

Puede ser interesante dependiendo de los defectos que podamos encontrar en las redes a las que pertenezcamos, utiliza el sistema libpam-runtime luego es trivial su instalación y se expone aquí:

```
apt-get install libpam-mklocaluser
```

```
-"PAM profiles to enable:"
SSS authentication, Unix authentication, AFS session management
-"Create local accounts and home directory on first time login"
Yes
```

Otra posibilidad es simplemente crear un directorio casa temporal; también existe módulo de PAM al respecto, véase man pam_mkhomedir. Este módulo no tiene soporte aún[20] para el sistema libpam-runtime, sin embargo.

Test log-in Primero comprobamos que NSS, el principal KERBEROS y el usuario AFS son usables:

```
id umea
kinit umea; aklog
klist; tokens
unlog; kdestroy
```

[20] http://bugs.debian.org/568577

```
login umea
```

```
pwd;
ls -A;
klist; tokens
exit
```

Ante problemas, podemos pedir a sssd que emita información de depuración:

```
invoke-rc.d sssd stop
sssd -i -d 0x4000
```

- -i interactive mode
- -d debug level, 0x4000 es de muy bajo nivel, hay otros consultables en man sssd.

Instalación de clientes para el resto de servicios (IM, VoIP, E-mail)

```
apt-get install pidgin finch libsasl2-modules-gssapi-mit libgsasl7 \
                icedove icedove-l10n-es-es xul-ext-sieve enigmail  \
                        xul-ext-nostalgy iceowl-extension          \
                mutt-patched urlview lbdb libnet-ldap-perl         \
                sieve-connect libgssapi-perl
```

- Si se quiere aportar autocompletado para mutt, quedaría aún configurar lbdb o un script alternativo; revísese en el capítulo sobre el Servicio de Correo la subsubsección:
  - Soporte de autocompletado de direcciones en LDAP para mutt.
- Si queremos en icedove (Thunderbird) soporte de libreta de direcciones usando CARDDAV, debemos instalar por el sistema de extensiones de éste a sogo-connector y moreFunctionsForAd pero no es interesante en nuestra infraestructura pues ésta no provee una interfaz CARDDAV, los contactos se obtienen directamente consultando a LDAP (no hace falta extensione para ésto).

1.6.3. Windows

En el caso de Windows, no se puede integrar nuestra infraestructura con su secuencia de log-in. Lo que se suele hacer es incluir otro elemento más: un controlador de dominio Samba que, usando los scripts del proyecto Smbldap-tools y el esquema samba.schema, consiga utilizar LDAP como almacén de metadatos de cuentas; el cliente usaría el sistema de "Profiles" para conseguir indirectamente (sin ticket KERBEROS ni token AFS) acceder al árbol AFS exportado como un directorio remoto CIFS. Quizás la próxima versión de Samba (Samba4) mejore algo estos despliegues pero, en general, el proceso es complicado e incompleto.

Lo que sí permite la distribución para Windows de MIT Kerberos y OpenAFS, una vez instalados, es utilizar el log-in exitoso a una cuenta local para, reutilizando la contraseña contra el KERBEROS AS, conseguir el TGT KERBEROS y el token AFS, aportando además algunas utilidades gráficas para administración de credenciales (NIM). Los únicos requisitos para este escenario serían:

- Que el nombre de la cuenta local coincida con el primario del principal (puede que el software incorpore alguna solución a ésto, necesitamos investigarlo más).
- Que la contraseña de la cuenta local coincida con la contrastada en el KERBEROS AS para ese principal.

Preámbulo: cuenta local de la que dependemos

Deberíamos crear, si no existiese, la cuenta local de la que, como se ha explicado, depende la cuenta centralizada (por ejemplo umea):

```
rem [Comprobado en Windows XP]


rem Crea con defaults, ve'ase  net user /?
net user umea /add
rem Ingresa una contrasen~a, con '*' nos ofrecera' un prompt para escribirla
net user umea *
rem Para eliminarla usamos /del:
rem net user umea /del


rem ---


echo USERDOMAIN %USERDOMAIN%
echo USERNAME %USERNAME%
echo USERPROFILE %USERPROFILE%
echo HOMEPATH %HOMEPATH%
```

Cliente OpenVPN

Nuestro testbed ofrece los servicios a través de una VPN realizada con OpenVPN. Habría que configurar la interfaz de red para conectarnos a la red preexistente y, con salida ya a internet, bajar e instalar la versión de OpenVPN para Windows:

http://openvpn.se/development.html[21]

A continuación se deberían seguir las siguientes secciones del capítulo sobre la VPN de la organización:

- Certificados para el rol de OpenVPN tls-client. Además en la máquina con Windows creamos el directorio <ruta instalación OpenVPN>\ovpn-keys-casafx.dyndns.org. Entonces transferimos (por ejemplo utilizando un pendrive, SCP u otro medio seguro) el material criptográfico (<host>.casafx.dyndns.org.{key,p12,crt}, ca.crt) desde dklab1 hasta ese directorio.
- Configuración de OpenVPN en modo tls-client. Adaptamos la configuración al nombre de la máquina con Windows y la almacenamos en <directorio de instalación OpenVPN>\config\casafx-tls-client.conf

[21] Los paquetes de ese enlace incorporan una GUI, los últimos paquetes oficiales se encuentran en http://openvpn.net/index.php/open-source/downloads.html

- Configuración de OpenVPN en modo tls-server, es decir en dklab1 y, para que la máquina con Windows reciba una IP fija a través del sistema CCD, hemos de seguir el último párrafo de esa sección.

Anotamos en cualquier caso que, dada una interfaz de red, en Windows se puede configurar manualmente las interfaces a través de la utilidad netsh de la siguiente manera:

```
rem [Comprobado en Windows XP]


rem Configuracio'n de red:

rem Nombre de las interfaces disponibles, y si activas:

netsh interface show interface


rem En el caso de configuracio'n esta'tica, por ejemplo:

netsh interface ip set address name="<nombre interfaz>" ^

     source=static addr=10.168.1.9 mask=255.255.255.0 gateway=10.168.1.1 ^

     gwmetric=0


rem En el caso de configuracio'n dina'mica con DHCP:

netsh interface ip set address "<nombre interfaz>" dhcp
```

- Anotación: podemos aprovechar también e instalar un cliente SSH/SCP tal como putty:
  - http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
  - http://winscp.net

Cliente DNS

```
rem [Comprobado en Windows XP]


rem Si configuracio'n de red esta'tica:
netsh interface ip set dns name="<nombre interfaz" ^
      source=static addr=10.168.1.1 register=PRIMARY
netsh interface ip add dns name="<nombre interfaz> addr=10.168.1.2 index=2


rem Si configuracio'n dina'mica
netsh interface ip set dns "<nombre interfaz>" dhcp
```

Evidentemente, deberíamos plantearnos insertar un DNS RR de tipo A para el host con Windows, de forma que éste tenga su propio nombre en la red. Consúltese en el capítulo sobre DNS, la sección:

- Vista "vpn", resolución directa e inversa. Un ejemplo allí de registro A es el de la máquina nowhere (que era ficticia, precisamente un ejemplo).

Cliente NTP

La implementación de NTP nativa de Windows no parece que soporte Autokey v2 IFF[22]. En cualquier caso no es obligatorio que el cliente lo utilice, así:

Sin Autokey La opción nativa:

```
rem [Comprobado en Windows XP]


net time /setsntp:"ntp.casafx.dyndns.org
                   0.es.pool.ntp.org 1.europe.pool.ntp.org"
net time /querysntp
net stop w32time && net start w32time
```

[22]http://technet.microsoft.com/en-us/library/cc773263%28WS.10%29.aspx
La búsqueda "ntp autokey" en el sitio technet.microsoft.com tampoco devuelve nada.

Para que los cambios sean permanentes debemos modificar el registro. En Windows XP puede consultarse, desde la utilidad regedit, el nodo[23]:
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\

Con Autokey Si queremos utilizar Autokey en Windows, debemos instalar Ntp.org para Windows:
http://doc.ntp.org/4.2.6/hints/winnt.html
En esa página, además, efectivamente confirma que Autokey está implementado también para Windows. A continuación seguiremos y adaptaremos, en este mismo capítulo, la sección:

- Linux (Debian 7 "Wheezy")
  - Cliente NTP
    - Con autokey
- Anotación: Nosotros no hemos comprobado esta posibilidad; no obstante sabemos que se aconseja utilizar en todas las máquinas implicadas la misma versión de OpenSSL[24].

Cliente KERBEROS

SecureEndpoints desarrollaba la distribución de MIT Kerberos para Windows (llamada "KfW" y recomendada entonces desde el sitio de MIT Kerberos) así como la aplicación de usuario para la administración de credenciales ("Network Identity Manager", NIM).

- https://www.secure-endpoints.com/#kfw
- En https://www.secure-endpoints.com/#Network%20Identity%20Manager podremos hacernos una idea de este software y su sistema de módulos, además ahí se apunta a (pues la versión que viene con KfW puede ser antigua):
  - https://www.secure-endpoints.com/netidmgr/v2/

Desafortunadamente, durante el año 2012 la situación del soporte MIT Kerberos en Windows ha estado cambiando. Básicamente el porte de MIT Kerberos a Windows se estancó en 2007 con la versión 3.2.2 de KfW, que nunca tuvo soporte para X86-64 ni versiones de Windows más allá de Vista. Es cierto que en 2013 se actualizó dicho porte, se implementó soporte para 64 bits y las últimas versiones de Windows con el lanzamiento de la versión 4.0.x de KfW, pero en ese tiempo SecureEndpoints decidió ofrecer Heimdal Kerberos para Windows[25], otra implementación, y dejar de ofrecer su versión de KfW.

[23] http://support.microsoft.com/kb/q223184/
http://nj-secure.blogspot.com/2006/02/windows-as-ntp-client.html
[24] http://lists.ntp.org/pipermail/bugs/2011-February/013145.html
[25] http://www.h5l.org/
https://www.secure-endpoints.com/#heimdal
Tabla comparativa con KfW 3.2.2 (ya superada) https://www.secure-endpoints.com/heimdal/

Como decimos, la situación está cambiando y la versión 4.0.x de KfW que se distribuye desde el sitio del MIT tiene su propio gestor de credenciales distinto a NIM, y dice no soportar (el gestor, entendemos ¿?) a AFS.

- http://web.mit.edu/kerberos/kfw-4.0/kfw-4.0.html

NIM funciona con ambas implementaciones de KERBEROS. Nosotros no hemos probado a Heimdal, daremos información sólo de nuestra experiencia con KfW 3.2.2 y NIM. Conforme pase el tiempo la situación se volverá a estabilizar.

Instalamos KfW con NIM, nuestras opciones se resumen en:

- Elegimos "custom install".
- Seleccionamos que se instalen KfW, Cliente, documentación. No SDK, ni debug.
- Seleccionamos autostart para NIM en cada log in.
- Seleccionamos para que se asegure de que el TGT KERBEROS esté disponible al iniciar la sesión en Windows ("Ensure krb tgt are available throughout win login session").

El instalador no nos ha pedido el krb5.ini (que es el krb5.conf que conocemos de MIT Kerberos en Debian), pero hay que crearlo y transferirlo a C:\WINDWOS\krb5.ini. Puede usarse como referencia la sección, de este mismo capítulo:

- Los hipotéticos usuarios de la infraestructura
    - Linux (Debian 7 "Wheezy")
        - Cliente Kerberos
            - Modificaciones

Tras la instalación no aparece más que NIM para configurar MIT Kerberos (no hay nada más ni en menú inicio ni en panel control). Este gestor, por cierto, no necesita principal alguno para realizar su cometido.

Si abrimos la aplicación NIM, se puede ver que aún no tiene instalado el plug-in que le permite automáticamente conseguir el token AFS a partir del TGT KERBEROS. Éste viene con el instalador de OpenAFS para Windows. Por otro lado, podemos comprobar que su REALM es el correcto.

Cliente AFS

SecureEndpoints es el principal desarrollador del porte a Windows de OpenAFS, así como del módulo para NIM.

Nosotros tuvimos oportunidad de comprobar el funcionamiento de la versión 1.6.x que utiliza una tecnología "Microsoft's SMB redirector network" para implementar AFS encima de la infraestructura de SMB/CIFS. La nueva versión, 1.7.5, implementa por

fin AFS como un sistema de ficheros nativo de Windows entre otras mejoras críticas y está disponible para Windows XP SP3 en adelante. Se recomienda utilizar Heimdal KERBEROS con la rama 1.7.x, si bien no leemos que sea incompatible con KfW.

Con los datos anteriores debiéramos poder elegir y descargar la versión 1.6.x o la 1.7.x desde:

- https://www.secure-endpoints.com/openafs-windows.html

Puede consultarse una guía de instalación en la wiki oficial de OpenAFS[26], no obstante nuestras opciones al instalador interactivo 1.6.x se resumen en:

- Elegimos instalación "Custom".
- Seleccionamos el cliente, loopbackAdapter (no en 1.7.x) y el módulo para NIM. No servidor AFS ni soporte de depuración, por tanto.
- Obtener una lista de servidores para otras celdas en http://grand.central.org/dl/cellservdb/Cell
- Default cell: casafx.dyndns.org
- Sí a que el log in esté integrado con la expedición del token AFS
- Sí a que el flujo de datos esté encriptado. Recordemos que es el cliente el que pide encriptar o no.
- Freelance Mode, se elige si el equipo es un portátil que va a pasar probablemente por períodos de desconexión respecto de su célula.
- Sí utilizo a DNS para encontrar otras células junto al CellServDB.
- Detecta cambios de IP y vuelve a pedir las credenciales en tal caso.
- Si el nombre de una cuenta en windows no coincidiese con el del principal Kerberos, aún puedodemos integrar AFS en el log-in, activando "Auto initialize AFS Credentials".
- Se asegura de que el mapeo de una unidad de windows y AFS sea permanente.
- Sí a modo silencioso, tal que no notifique que el administrador de credenciales no está cargado.
- Sí a que quede en la bandeja del sistema al inicio sin informar de nada más.
- Se abrirán los puertos TCP y UDP del 7000 al 7009 en el cortafuegos de Windows.

[26] http://wiki.openafs.org/AFSLore/WindowsEndUserQuickStartGuide/

A partir de aquí cualquier otro cambio[27] se hará en Inicio->Panel de Control->AFS Client Config. Imprescindible será, por ejemplo, el mapeo[28] de células AFS a unidades, de tal forma que nuestra celda se monte en, por ejemplo, la unidad "S:". El menú de inicio también permite hacer algunos cambios.

Gracias a la labor del NIM, tras autenticarnos tendremos un token AFS que nos permitirá acceder a los contenidos AFS montados según el mapeo de unidades mencionado. Echamos en falta que el home del usuario esté, además, en AFS en lugar de "C:\Document and Settings\<username>". Puede emplearse el uso de Logon-Scripts[29] genéricos o cambiando localmente el metadato home de cada usuario para que apunte a AFS:

```
rem net user /? para ayuda.


net user umea /homedir:<ruta>


rem Existe una aplicacio'n gra'fica, lusrmgr.msc.
```

La ruta a AFS puede expresarse a través de la sintaxis <unidad>:<path>, o a través del UNC \\AFS\<path>, si bien el segundo ha estado sujeto a errores en algunas versiones de Windows[30].

Resolución de Metadatos

No podemos hacer que busque los metadatos relativos a la cuenta en LDAP, dependíamos por ello de una cuenta local intermediaria.

Autenticación

No podemos modificar el sistema de autenticaciones para que ésta se realice directamente contra KERBEROS AS; al menos NIM conseguía que si la autenticación local era exitosa, se reutilizase el password para conseguir el TGT KERBEROS y token AFS.

Instalación de clientes para el resto de servicios (IM, VoIP, E-mail)

- Pidgin (IM, VoIP): http://pidgin.im/download/windows/
- Mozilla Thunderbird (E-mail): www.mozilla.org/en-US/thunderbird/all.html

[27] http://wiki.openafs.org/AFSLore/WindowsConfigurationReferenceGuide/
[28] http://kb.iu.edu/data/adyv.html
[29] http://technet.microsoft.com/en-us/library/cc758918%28WS.10%29.aspx
[30] http://rt.central.org/rt/Ticket/Display.html?id=50864

1.6.4. Otros SO POSIX

Existen distribuciones OpenVPN[31], Ntp.org[32], MIT Kerberos[33], OpenAFS[34], Pidgin[35] y Mozilla Thunderbird[36] para otros SO POSIX como Solaris o Mac OSX, cada uno con su propia metodología de instalación de dicho software pero con una configuración que no diferirá en líneas generales de la que se dio para GNU/Linux. Los principales escollos pueden estar en que no utilicen NSS y PAM, sino otros sistemas equivalentes en funcionalidad. A este respecto, es conocido que:

- NSS: GNU/Linux, FreeBSD, NetBSD, HP-UX, IRIX, AIX y Solaris implementan el sistema NSS en su librería C. A su vez el módulo libnss-ldapd sólo funciona con la librería C de GNU/Linux, sin embargo el módulo libnss-ldap del que deriva el primero y de similar configuración, soporta también a AIX, Solaris, FreeBSD, HP-UX; IRS y ONC+. El equivalente a NSS de MAC sería su software no libre Apple Open Directory, el cual soporta LDAP etc.

- PAM es utilizado en GNU/Linux; DragonFly BSD, FreeBSD, Mac OS X, NetBSD; Solaris; AIX; HP-UX. Sin embargo cada uno tiene su implementación PAM y el módulo libpam-krb5 (y por tanto, entendemos, parcialmente la configuración dada aquí), se afirma que funcionaría en GNU/Linux, AIX, Mac OSX y HP-UX (al igual que libpam-afs-session, del mismo autor). En los demás habría que comprobar si su sistema PAM tiene soporte KERBEROS v5 y cómo se configura.

Mac OSX (algunas claves y enlaces)

Básicamente, Mac OSX viene con Ntp.org y el cliente MIT Kerberos preinstalados, no así OpenAFS[37], OpenVPN[38], Pidgin[39] o Mozilla Thunderbird[40]. Suponiendo que instaláramos a éstos últimos, la configuración sería similar a la que se expuso para GNU/Linux matizada por el grado de integración en su equivalente NSS/PAM. No disponemos de Mac OSX para hacer pruebas, por ello daremos algunas claves y enlaces a otras guías. Existen dos posibilidades de integración:

- Como ocurría en Windows, permite que si una cuenta local coincide en nombre de usuario y contraseña con el principal, se reutilice la contraseña expida el TGT KERBEROS y el token AFS durante la secuencia de log-in.

[31] http://openvpn.net/index.php/open-source/documentation/miscellaneous/porting-notes.html
[32] http://www.ntp.org/ntpfaq/NTP-s-def-impl.htm#Q-NTP-IMPL-UNIX
[33] http://k5wiki.kerberos.org/wiki/Supported_platforms
[34] http://git.openafs.org/?p=openafs.git;a=blob_plain;f=README;hb=HEAD
[35] http://developer.pidgin.im/wiki/Installing%20Pidgin#Compiling
http://developer.pidgin.im/wiki/FAQssl (mención Solaris, *BSD)
http://www.filehippo.com/download_pidgin/changelog/3963/ (mención AIX)
[36] https://developer.mozilla.org/en/Supported_build_configurations
[37] http://www.openafs.org/pages/macos.html
[38] http://code.google.com/p/tunnelblick/
[39] http://www.pidgin.im/download/mac/
[40] http://www.mozilla.org/en-US/thunderbird/all.html

- http://tig.csail.mit.edu/twiki/bin/view/TIG/OpenAFSAndMacOSX

- La funcionalidad completa parece ser que es posible también. Es decir, para que no intermedien cuentas locales, sino que se utilice directamente la identidad centralizada en LDAP, el log in sea único en el KERBEROS AS y el home sea en AFS, tal como hacemos en GNU/Linux. aquí daremos simplemente las siguientes pistas para seguir investigando:

  - Apple Open Directory, como se ha dicho, es el sistema NSS/PAM nativo en MAC, con soporte LDAP, KERBEROS, SASL.
  - /etc/authorization controla si se expiden tickets KERBEROS al log-in[41].
  - Algunas guías:
    - https://wiki.inf.ed.ac.uk/DICE/DiceMac
    - http://www.stanford.edu/group/macosxsig/blog/2008/03/connecting_105_to_stan
    - http://www.ibiblio.org/macsupport/kerberos/kerberos10_4.html

- Anotación: es conocido que Mac OSX puede integrarse también en un entorno Active Directory.

1.7. Nuevo usuario en nuestra organización: creación de los recursos para una cuenta nueva

Supongamos que un nuevo usuario se incorpora a nuestra organización. Entonces, precisa una nueva cuenta propia que le permita usar el parque informático desplegado. El hecho de plantearnos esta cuestión en este punto no es casualidad. Puesto que los servicios que restan por desplegar son de tipo persona-a-persona, necesitamos una cuenta adicional a la de umea (y que podemos llamar, especularmente, aemu). Sólo con dos cuentas podremos hacer pruebas de servicios como Mensajería Instantánea (IM), Telefonía (VoIP), Correo Electrónico (E-Mail).

- Anotación: La siguiente secuencia de comandos crea todos los recursos necesarios, incluyendo aquellos para los servicios que aún no se han desplegado. En concreto se crean los recursos necesarios en OpenLDAP, MIT Kerberos y OpenAFS así como los ficheros de configuración de Pidgin (IM, VoIP), Mutt y Mozilla Thunderbird (E-Mail). Así, esta sección (que es necesaria ejecutar ahora para, como hemos explicado, crear la cuenta alternativa aemu) será sólo comprensible cuando se haya leído el resto de capítulos. No queríamos repetirnos y por éso aprovechamos la creación de aemu para exponer cómo se crea una cuenta nueva con soporte para *todos* los servicios. Sí será umea la cuenta para la que se crearán los recursos posteriores conforme se vayan necesitando en el resto de capítulos, tal como se ha hecho hasta ahora.

[41] http://developer.apple.com/library/mac/#documentation/Security/Conceptual/authorization_concepts/01intro

1.7.1. Variables

Procuraremos utilizar variables de shell siempre que nos sea posible, con idea de que las secuencias de comandos que se dan puedan ser fácilmente adaptables a scripts.

```
# Nota: el rango para uid 60000-64999 no deberi'a usarse sin
#       leer previamente /usr/share/doc/base-passwd/README

NOMBRE=aemu
APELLIDO=aladyn
UIDu=31001
GIDu=31001
DEPARTAMENTO=sedeB
ACPASS=${NOMBRE}pa55    #read -p 'krb5 AC pass: ' ACPASS
DONDE_VOL=dklab1
DONDE_REP=dklab2
QUOTA_VOL=15000         # kB
SUBRUTA2NIVELES=`printf ${NOMBRE}|cut -c -1`/`printf ${NOMBRE}|cut -c -2`

DIT="dc=casafx,dc=dyndns,dc=org"
REALM="CASAFX.DYNDNS.ORG"
DOMAIN=`printf $REALM | awk '{print tolower($0)}'` #REALM en minu'sculas
dklabX=dklab$(echo `date +%S`%2 +1 | bc). # dklab1. o dklab2.

MAILSIZEMAX=10000       # kB
MAILQUOTASIZE=5000      # kB
MAILQUOTACOUNT=0
SPAMASSASSINUSERPREFS="required_score 5.00"
MAILALTERNATEADDRESS=""
MAILFORWARDINGADDRESS=""
DELIVERYMODE=''         #DELIVERYMODE=noprogram
```

```
#Nota: En el siguiente texto, se ha de sustituir
#      "Estare"->"Estare'", "procurare->"procurare'":
(echo '
Thank you for your email.
I am out of the office, I will respond upon my return.

Regards.'
 echo
 echo '---'
 echo '
Gracias por su correo.
Estare lejos de la oficina, pero procurare responder a mi regreso.

Saludos.'
 echo)\
 > /tmp/generic_vacation.txt
```

1.7.2. LDAP (metadatos de todos los servicios)

Anotamos que el fichero LDIF que se construye a continuación contiene los metadatos de todos los servicios y por tanto, necesita que en el DIT cn=config estén cargados todos los esquemas. Hasta el momento están cargados (vienen por defecto):

- nis.schema
- inetorgperson.schema

necesarios para las cuentas de shell POSIX. El servicio de correo necesita:

- qmail-ldap.schema

y el servicio de telefonía (aún pendiente para un futuro):

- asterisk.schema

Así, en el fichero LDIF que confeccionamos a continuación, se indican en mayúsculas el comienzo de los objetos y atributos relativos a cada servicio, de forma que se puedan eliminar las partes para las que cn=config no tenga aún cargado el esquema.

Evidentemente, la primera vez que leamos este documento, llegamos aquí con sólo los esquemas "nis" e "inetorgperson" cargados, y por tanto sólo los objetos y atributos del servicio de cuentas de shell POSIX debieran formar parte del fichero LDIF (más tarde, en el resto de los capítulos, se expondrá cómo incluir el resto de esquemas y cómo modificar el directorio para incluir los objetos y atributos restantes). Pero, si llegamos aquí tras completar toda la infraestructura, el fichero LDIF, tal cual está, sirve al propósito de crear nuevas cuentas con todos los metadatos en LDAP que el despliegue precisa.

```
cd ~
mkdir -p ldif
cat <<EOF >ldif/cuenta_${NOMBRE}.ldif
# P O S I X   S H E L L   A C C O U N T
# (necesita que *.schema este'n cargados previamente en cn=config)
dn: cn=${NOMBRE},ou=groups,ou=accounts,${DIT}
cn: ${NOMBRE}
gidNumber: ${GIDu}
objectClass: top
objectClass: POSIXGroup

dn: uid=${NOMBRE},ou=users,ou=accounts,${DIT}
uid: ${NOMBRE}
uidNumber: ${UIDu}
gidNumber: ${GIDu}
cn: ${NOMBRE}
sn: ${APELLIDO}
objectClass: top
objectClass: POSIXAccount
objectClass: shadowAccount
objectClass: inetOrgPerson
loginShell: /bin/bash
homeDirectory: /afs/${DOMAIN}/user/${SUBRUTA2NIVELES}/${NOMBRE}
userPassword: {CRYPT}*
#
```

```
## Inetorgperson Class
#jpegPhoto:< file://var/tmp/${NOMBRE}.jpg #Algunos srv recomiendan <= 16kB
departmentNumber: ${DEPARTAMENTO}
givenName: givenname ${APELLIDO}
displayName: ${NOMBRE} ${APELLIDO} (displayname)
mail: ${NOMBRE}@${DOMAIN}
labeledURI: http://${DOMAIN}/~${NOMBRE}/ Home page
#labeledURI: http://${DOMAIN}/pics/${NOMBRE}.jpg [photo]
#userSMimeCertificate:
#homePhone
#homePostalAddress
```

```
#
# J A B B E R
# (usa los objetos y atributos anteriores)
#
# M A I L
# (necesita que qmail-ldap.schema este' cargado previamente en cn=config)
objectClass: qmailUser
accountStatus: active
mailSizeMax: ${MAILSIZEMAX}
mailQuotaSize: ${MAILQUOTASIZE}
mailQuotaCount: ${MAILQUOTACOUNT}
spamassassinUserPrefs: ${SPAMASSASSINUSERPREFS}
deliveryMode: ${DELIVERYMODE}
#deliveryMode: nolocal
#deliveryMode: noforward
#deliveryMode: noprogram
#deliveryMode: replay
mailAlternateAddress: ${MAILALTERNATEADDRESS}
mailForwardingAddress: ${MAILFORWARDINGADDRESS}
deliveryProgramPath: |/usr/bin/procmail
mailMessageStore: Maildir
mailReplyText:< file:///tmp/generic_vacation.txt
# ...
```

```
#
# V O I P
# (necesita que asterisk.schema este' cargado previamente en cn=config)
# ...
EOF


kinit root/admin; aklog
ldapadd -Qf ldif/cuenta_${NOMBRE}.ldif
ldapsearch -LLL  \
   -b  "uid=${NOMBRE},ou=users,ou=accounts,${DIT}" \
   -s base
```

1.7.3. Kerberos (Principal)

```
kadmin.local -p root/admin -q \
 "addprinc -policy user -pw ${ACPASS} ${NOMBRE}@${REALM}"
kadmin.local -p root/admin -q \
 "getprinc ${NOMBRE}@${REALM}"
```

1.7.4. AFS (Id, Home)

```
pts createuser ${NOMBRE} -id ${UIDu}
pts examine ${NOMBRE}
 #Volumen AFS para el home
vos create ${DONDE_VOL}.${DOMAIN}. a user.${NOMBRE} \
    -maxquota ${QUOTA_VOL}
vos listvol ${DONDE_VOL}.${DOMAIN}. -localauth
vos examine user.${NOMBRE}
 #Montaje volumen en a'rbol afs
mkdir -p /afs/${DOMAIN}/user/‘
         printf ${NOMBRE}|cut -c -1‘/‘printf ${NOMBRE}|cut -c -2‘
cd /afs/${DOMAIN}/user/‘
         printf ${NOMBRE}|cut -c -1‘/‘printf ${NOMBRE}|cut -c -2‘
fs mkmount ${NOMBRE} user.${NOMBRE} -rw
fs setacl  ${NOMBRE} ${NOMBRE} all
fs setacl  ${NOMBRE} system:anyuser ""
fs setacl  ${NOMBRE} system:authuser ""
fs listacl ${NOMBRE}
fs lsmount ${NOMBRE}
 #Re'plica volumen afs
vos addsite -server ${DONDE_REP}.${DOMAIN}. -partition a \
            -id user.${NOMBRE} -verbose
vos release user.${NOMBRE} -verbose
vos listvol ${DONDE_REP}.${DOMAIN}.
```

```
 #Backup diario volumen AFS
bos create -type cron -instance respaldovolusers \
           -server ${DONDE_VOL}.${DOMAIN}. \
           -cmd "/usr/bin/vos backupsys -prefix user -localauth" "03:00" \
&& \
bos status -server ${DONDE_VOL}.${DOMAIN}. -long
vos backupsys -prefix user "03:00" -dryrun



 #Test:
kdestroy

kinit ${NOMBRE}
klist
ldapwhoami -Q

aklog -d
tokens
cd ${NOMBRE}
echo RW > test
cat test
fs listacl test
ls -l test

klist

login ${NOMBRE}
```

```
SKELDIR=/etc/skel
USUARIO=‘id -n -u‘
GRUPO=‘id -n -g‘
HOMEDIR=${HOME}
for i in ‘ls -A ${SKELDIR}‘
    do install -o ${USUARIO} -g ${GRUPO} -m 0664 ${SKELDIR}/${i} ${HOMEDIR}
done
echo "export HISTSIZE=1000000; export HISTFILESIZE=1000000" >> .bashrc
ls -A
exit
```

1.7.5. XMPP (Pidgin)

```
cd ~
mkdir -p /afs/${DOMAIN}/user/${SUBRUTA2NIVELES}/${NOMBRE}/.purple
cat <<EOF > \
/afs/${DOMAIN}/user/${SUBRUTA2NIVELES}/${NOMBRE}/.purple/accounts.xml
<?xml version='1.0' encoding='UTF-8' ?>

<account version='1.0'>
        <account>
                <protocol>prpl-jabber</protocol>
                <name>${NOMBRE}@${DOMAIN}/</name>
                <statuses>
                   <status type='available' name='Available' active='true'>
                           <attributes/>
                   </status>
                   <status type='mood' name='Feeling' active='false'>
                           <attributes/>
                   </status>
                   <status type='freeforchat' name='Chatty' active='false'>
                           <attributes/>
                   </status>
                   <status type='away' name='Away' active='false'>
                           <attributes/>
                   </status>
```

```
          <status type='extended_away' name='Extended away' active='false'>
                          <attributes/>
                  </status>
                   <status type='dnd' name='Do Not Disturb' active='false'>
                          <attributes/>
                  </status>
                  <status type='offline' name='Offline' active='false'>
                          <attributes/>
                  </status>
                </statuses>

                <settings>
                 <setting name='check-mail' type='bool'>0</setting>
                 <setting name='old_ssl' type='bool'>0</setting>
                 <setting name='auth_plain_in_clear' type='bool'>0</setting>
                 <setting name='require_tls' type='bool'>1</setting>
     <setting name='ft_proxies' type='string'>proxy.eu.jabber.org</setting>
                <setting name='use-global-buddyicon' type='bool'>1</setting>
                 <setting name='custom_smileys' type='bool'>1</setting>
                 <setting name='port' type='int'>5222</setting>
                </settings>
                <settings ui='gtk-gaim'>
                        <setting name='auto-login' type='bool'>1</setting>
                </settings>
                <current_error/>
        </account>
</account>
EOF
```

1.7.6. Mail (AFS Storage)

```
DONDE_MAILST=${DONDE_REP}
vos create ${DONDE_MAILST}.${DOMAIN}. a service.mail.${NOMBRE} \
    -maxquota 5000

cd /afs/${DOMAIN}/service
mkdir -p mail/${SUBRUTA2NIVELES}
fs mkmount mail/${SUBRUTA2NIVELES}/${NOMBRE} service.mail.${NOMBRE} -rw
fs listacl mail/${SUBRUTA2NIVELES}/${NOMBRE}
fs setacl  mail/${SUBRUTA2NIVELES}/${NOMBRE} mailgroup all
fs setacl  mail/${SUBRUTA2NIVELES}/${NOMBRE} system:anyuser ""
fs setacl  mail/${SUBRUTA2NIVELES}/${NOMBRE} system:authuser ""
mkdir mail/${SUBRUTA2NIVELES}/${NOMBRE}/Maildir
mkdir ../user/${SUBRUTA2NIVELES}/${NOMBRE}/.PublicMailboxIndexes
fs setacl ../user/${SUBRUTA2NIVELES}/${NOMBRE}/.PublicMailboxIndexes \
   mailgroup rlidwk
fs setacl ../user/${SUBRUTA2NIVELES}/${NOMBRE}/.PublicMailboxIndexes \
   mailgroup l

cd /afs/${DOMAIN}/user
mkdir ${SUBRUTA2NIVELES}/${NOMBRE}/Maildir
fs setacl ${SUBRUTA2NIVELES}/${NOMBRE}/Maildir umea all
fs setacl ${SUBRUTA2NIVELES}/${NOMBRE}/Maildir mailgroup rwklid
fs setacl ${SUBRUTA2NIVELES}/${NOMBRE}/Maildir system:anyuser ""
fs setacl ${SUBRUTA2NIVELES}/${NOMBRE}/Maildir system:authuser ""
fs setacl ${SUBRUTA2NIVELES}/${NOMBRE} mailgroup l
cd ~
```

1.7.7. Mail (Mutt/Thunderbird)

Mutt

```
cat <<EOF \
    > /afs/${DOMAIN}/user/${SUBRUTA2NIVELES}/${NOMBRE}/.muttrc
# Do'nde llega el correo:
set spoolfile = imap://${NOMBRE}@\`
 dig @${dklabX}${DOMAIN}. _imap._tcp.${DOMAIN} SRV +short
 | awk '{print \$4}'\`
set folder = ~/Maildir
set mbox_type = Maildir

# Buzones relacionados con la edicio'n de correos: borradores, copias...
# no'tese que si disponemos estos buzones en imap y no localmente,
# podra'n ser compartidos por todos los clientes, siendo e'sto deseable:
set copy = yes
set record = !Sent      # imap://${NOMBRE}@${dklabX}${DOMAIN}/Sent
set postponed = !Drafts
#... imap://${NOMBRE}@${dklabX}${DOMAIN}/Drafts
set trash = !Trash
#... imap://${NOMBRE}@${dklabX}${DOMAIN}/Trash

# Los buzones sobre los que queremos navegar sera'n los anteriores
# ma's subscritos:
# de esta forma se an~aden/suprimen dina'micamente, sin declararlo aqui',
# todos los buzones a los que estemos subscritos (comando LSUB de IMAP).
# '!' equivale al spoolfile
# '=' equivale al folder
mailboxes ! !Sent !Drafts !Trash =
set imap_check_subscribed = yes
```

```
set imap_list_subscribed  = no
#... e'sta permite que al pulsar la "y" y entrar en modo mailboxes browser,
#    liste todas las mailboxes incluso no subscritas, asi' que puedas
#    subscribirte con "s" ("u" para unsubscribe). "T" llama a
#    toggle-subscribed.
#    nota: no hemos acertado a utilizarlo, pero asi' esta' descrito en el
#          manual.

# Smarthost LMTP
set smtp_url = smtp://${NOMBRE}@\`
 dig @dklab1.${DOMAIN}. _submission._tcp.${DOMAIN} SRV +short|
 awk '{print \$4}'\`:587

# Para'metros de autentificacio'n y encriptacio'n SMTP/IMAP:
set smtp_authenticators=gssapi
set imap_authenticators=gssapi
set ssl_ca_certificates_file=/etc/ssl/certs/casafx-ca.crt

# Composicio'n de correos:
set editor="/usr/bin/vim +8"
set edit_headers

# Declara contenido de la cabecera "From:" (y otras) en los
# correos enviados:
my_hdr From: ${NOMBRE}@${DOMAIN}
my_hdr Jabber-ID: ${NOMBRE}@${DOMAIN}
#my_hdr X-PGP-Key: http://${DOMAIN}/~${NOMBRE}/pubkey.asc
```

```
# Correo recibido, criterio de ordenacio'n:
set sort=reverse-threads

# Tratamiento de adjuntos:
# "v" permite listarlos y, ahi', "s" permite salvarlos separadamente. Pero
# adema's si un correo lleva adjuntos de ciertos tipos MIME, podemos hacer
# que se procesen como texto simple para poder ser enviados al terminal
# y visualizarlos nada ma's abrir el correo en cuestio'n. Los programas
# responsables de ese procesamiento son los del sistema MIME/mailcap
# en /etc/mailcap. Recomendamos modificar el comportamiento para html asi':
# echo 'text/html; /usr/bin/w3m -I %{charset} -o display_link_number=1'   \\
#     '-T text/html -cols 68 '%s'; copiousoutput; description=HTML Text;'\\
#     'nametemplate=%s.html' > ~/.mailcap
# Si esta' instalado url-view, Control-b debiera presentarnos un menu'
# con todas las url presentes y poder lanzar un navegador sin copiar+pegar.
auto_view text/html
auto_view application/pdf
```

```
# Algunos atajos de teclados u'tiles, ba'sicamente para que n,p,i
# nos permitan movernos y seleccionar por el panel lateral de mailboxes,
# y k,j nos permitan movernos sobre la lista de correos. Por su lado, 'b'
# conmuta el panel lateral.
bind index k previous-entry
bind index j next-entry
bind index p sidebar-prev
bind index n sidebar-next
bind index i sidebar-open
bind pager p sidebar-prev
bind pager n sidebar-next
bind pager i sidebar-open
macro pager b '<enter-command>toogle sidebar_visible<enter><redraw-screen>'

# Las posibilidades de configuracio'n de mutt son ingentes,
# ma's informacio'n en:
# man muttrc
EOF
```

Mozilla Thunderbird

```
cd /afs/${DOMAIN}/user/${SUBRUTA2NIVELES}/${NOMBRE}/
mkdir -p .icedove/w5s1nswf.default
cat <<EOF > .icedove/profiles.ini
[General]
StartWithLastProfile=1

[Profile0]
Name=default
IsRelative=1
Path=w5s1nswf.default
EOF
```

(Una amplia proporción de las líneas que vienen a continuación aparecen divididas en 2 o 3 partes delimitadas por \ y cambio de línea. Realmente deben aparecer en una sóla línea, sin \ y cambio de línea:)

```
cat <<EOF > .icedove/w5s1nswf.default/prefs.js
# Mozilla User Preferences

/* Do not edit this file.
 *
 * If you make changes to this file while the application is running,
 * the changes will be overwritten when the application exits.
 *
 * To make a manual change to preferences, you can visit the
 * URL about:config
 * For more information, see
 * http://www.mozilla.org/unix/customizing.html#prefs
 */

user_pref("app.update.lastUpdateTime.addon-background-update-timer",\
1323801628);
user_pref("app.update.lastUpdateTime.background-update-timer",\
1323801549);
user_pref("app.update.lastUpdateTime.blocklist-background-update-timer"\
, 1323801549);
```

```
user_pref("extensions.enabledItems",\
"{847b3a00-7ab1-11d4-8f02-006008948af5}:1.0.1,langpack-es-ES@\
thunderbird.mozilla.org:3.0.522,nostalgy@alain.frisch:0.2.22,\
sieve@mozdev.org:0.1.13,{972ce4c6-7e08-4474-a285-3208198ce6fd}:3.0.11");
user_pref("extensions.enigmail.configuredVersion", "1.0.1");
user_pref("extensions.lastAppVersion", "3.0.11");
user_pref("extensions.sieve.account.${NOMBRE}@\
${dklabX}${DOMAIN}.TLS", true);
user_pref("extensions.sieve.account.${NOMBRE}@\
${dklabX}${DOMAIN}.TLS.forced", true);
user_pref("extensions.sieve.account.${NOMBRE}@\
${dklabX}${DOMAIN}.activeAuthorization", 1);
user_pref("extensions.sieve.account.${NOMBRE}@\
${dklabX}${DOMAIN}.activeHost", 1);
user_pref("extensions.sieve.account.${NOMBRE}@\
${dklabX}${DOMAIN}.activeLogin", 1);
user_pref("extensions.sieve.account.${NOMBRE}@\
${dklabX}${DOMAIN}.compatibility.tls", 0);
user_pref("extensions.sieve.account.${NOMBRE}@\
${dklabX}${DOMAIN}.enabled", true);
user_pref("extensions.sieve.account.${NOMBRE}@\
${dklabX}${DOMAIN}.firstRunDone", true);
user_pref("extensions.sieve.account.${NOMBRE}@\
${dklabX}${DOMAIN}.hostname", "${dklabX}${DOMAIN}");
user_pref("extensions.sieve.account.${NOMBRE}@\
${dklabX}${DOMAIN}.port", 4190);
```

```
user_pref("extensions.sieve.account.${NOMBRE}@\
${dklabX}${DOMAIN}.port.type", 0);
user_pref("extensions.sieve.account.${NOMBRE}@\
${dklabX}${DOMAIN}.proxy.type", 1);
user_pref("extensions.sieve.account.${NOMBRE}@\
${dklabX}${DOMAIN}.sasl.forced", false);
user_pref("idle.lastDailyNotification", 1324081550);
//user_pref("intl.charsetmenu.mailview.cache", "ISO-8859-1");
user_pref("intl.charsetmenu.mailview.cache", "UTF-8");
user_pref("ldap_2.autoComplete.directoryServer",\
"ldap_2.servers.dklab1");
user_pref("ldap_2.autoComplete.useDirectory", true);
user_pref("ldap_2.servers.dklab1.auth.dn", "");
user_pref("ldap_2.servers.dklab1.auth.saslmech", "");
user_pref("ldap_2.servers.dklab1.description", "dklab1");
user_pref("ldap_2.servers.dklab1.filename", "ldap.mab");
user_pref("ldap_2.servers.dklab1.maxHits", 100);
user_pref("ldap_2.servers.dklab1.uri",\
"ldap://dklab1.${DOMAIN}/\
ou=users,ou=accounts,${DIT}??sub?(objectclass=*)");
user_pref("ldap_2.servers.dklab2.auth.dn", "");
user_pref("ldap_2.servers.dklab2.auth.saslmech", "");
user_pref("ldap_2.servers.dklab2.description", "dklab2");
user_pref("ldap_2.servers.dklab2.filename", "ldap-1.mab");
user_pref("ldap_2.servers.dklab2.maxHits", 100);
user_pref("ldap_2.servers.dklab2.uri",\
"ldap://dklab2.${DOMAIN}/\
ou=users,ou=accounts,${DIT}??sub?(objectclass=*)");
```

```
user_pref("mail.account.account1.identities", "id1");
user_pref("mail.account.account1.server", "server1");
user_pref("mail.account.account2.server", "server2");
user_pref("mail.account.account3.server", "server3");
user_pref("mail.accountmanager.accounts", "account1,account2,account3");
user_pref("mail.accountmanager.defaultaccount", "account1");
user_pref("mail.accountmanager.localfoldersserver", "server2");
user_pref("mail.append_preconfig_smtpservers.version", 2);
user_pref("mail.attachment.store.version", 1);
user_pref("mail.folder.views.version", 1);
user_pref("mail.identity.id1.archive_folder",\
"imap://${NOMBRE}@${dklabX}${DOMAIN}/Archives");
user_pref("mail.identity.id1.doBcc", false);
user_pref("mail.identity.id1.draft_folder",\
"imap://${NOMBRE}@${dklabX}${DOMAIN}/Drafts");
user_pref("mail.identity.id1.drafts_folder_picker_mode", "0");
user_pref("mail.identity.id1.fcc_folder",\
"imap://${NOMBRE}@${dklabX}${DOMAIN}/Sent");
user_pref("mail.identity.id1.fcc_folder_picker_mode", "0");
user_pref("mail.identity.id1.fullName", "${NOMBRE}");
user_pref("mail.identity.id1.smtpServer", "smtp1");
user_pref("mail.identity.id1.stationery_folder",\
"imap://${NOMBRE}@${dklabX}${DOMAIN}/Templates");
user_pref("mail.identity.id1.tmpl_folder_picker_mode", "0");
user_pref("mail.identity.id1.useremail", "${NOMBRE}@${DOMAIN}");
user_pref("mail.identity.id1.valid", true);
```

```
user_pref("mail.openMessageBehavior.version", 1);
user_pref("mail.preferences.advanced.selectedTabIndex", 4);
user_pref("mail.preferences.compose.selectedTabIndex", 1);
user_pref("mail.rights.version", 1);
user_pref("mail.root.imap",\
"/afs/${DOMAIN}/user/${SUBRUTA2NIVELES}/${NOMBRE}/\
.icedove/w5s1nswf.default/ImapMail");
user_pref("mail.root.imap-rel", "[ProfD]ImapMail");
user_pref("mail.root.none",\
"/afs/${DOMAIN}/user/${SUBRUTA2NIVELES}/${NOMBRE}/\
.icedove/w5s1nswf.default/Mail");
user_pref("mail.root.none-rel", "[ProfD]Mail");
user_pref("mail.server.server1.capability", 101474849);
user_pref("mail.server.server1.check_new_mail", true);
user_pref("mail.server.server1.directory", \
"/afs/${DOMAIN}/user/${SUBRUTA2NIVELES}/${NOMBRE}/\
.icedove/w5s1nswf.default/ImapMail/${dklabX}${DOMAIN}");
user_pref("mail.server.server1.directory-rel",\
"[ProfD]ImapMail/${dklabX}${DOMAIN}");
user_pref("mail.server.server1.hostname", "${dklabX}${DOMAIN}");
user_pref("mail.server.server1.login_at_startup", true);
user_pref("mail.server.server1.max_cached_connections", 5);
user_pref("mail.server.server1.name", "${NOMBRE}@${DOMAIN}");
user_pref("mail.server.server1.namespace.personal", "\"\"");
user_pref("mail.server.server1.namespace.public", "\"Public/\"");
user_pref("mail.server.server1.socketType", 2);
user_pref("mail.server.server1.timeout", 29);
user_pref("mail.server.server1.type", "imap");
user_pref("mail.server.server1.useSecAuth", true);
```

```
user_pref("mail.server.server1.userName", "${NOMBRE}");
user_pref("mail.server.server1.using_subscription", false);
user_pref("mail.server.server2.directory",\
"/afs/${DOMAIN}/user/${SUBRUTA2NIVELES}/${NOMBRE}/\
.icedove/w5s1nswf.default/Mail/Local Folders");
user_pref("mail.server.server2.directory-rel", \
"[ProfD]Mail/Local Folders");
user_pref("mail.server.server2.hostname", "Local Folders");
user_pref("mail.server.server2.name", "Local Folders");
user_pref("mail.server.server2.type", "none");
user_pref("mail.server.server2.userName", "nobody");
user_pref("mail.server.server3.directory",\
"/afs/${DOMAIN}/user/${SUBRUTA2NIVELES}/${NOMBRE}/\
.icedove/w5s1nswf.default/Mail/smart mailboxes");
user_pref("mail.server.server3.directory-rel", \
"[ProfD]Mail/smart mailboxes");
user_pref("mail.server.server3.hidden", true);
user_pref("mail.server.server3.hostname", "smart mailboxes");
user_pref("mail.server.server3.name", "Smart Folders");
user_pref("mail.server.server3.type", "none");
user_pref("mail.server.server3.userName", "nobody");
```

```
user_pref("mail.smtpserver.smtp1.auth_method", 1);
user_pref("mail.smtpserver.smtp1.hostname", "${dklabX}${DOMAIN}");
user_pref("mail.smtpserver.smtp1.port", 587);
user_pref("mail.smtpserver.smtp1.try_ssl", 2);
user_pref("mail.smtpserver.smtp1.useSecAuth", true);
user_pref("mail.smtpserver.smtp1.username", "${NOMBRE}");
user_pref("mail.smtpservers", "smtp1");
user_pref("mail.spam.version", 1);
user_pref("mail.startup.enabledMailCheckOnce", true);
user_pref("mailnews.quotingPrefs.version", 1);
user_pref("mailnews.send_default_charset", "UTF-8");
//user_pref("mailnews.view_default_charset", "UTF-8");
user_pref("mailnews.start_page_override.mstone", "3.0.11");
user_pref("mailnews.tags.$label1.color", "#FF0000");
user_pref("mailnews.tags.$label1.tag", "Important");
user_pref("mailnews.tags.$label2.color", "#FF9900");
user_pref("mailnews.tags.$label2.tag", "Work");
user_pref("mailnews.tags.$label3.color", "#009900");
user_pref("mailnews.tags.$label3.tag", "Personal");
user_pref("mailnews.tags.$label4.color", "#3333FF");
user_pref("mailnews.tags.$label4.tag", "To Do");
user_pref("mailnews.tags.$label5.color", "#993399");
user_pref("mailnews.tags.$label5.tag", "Later");
user_pref("mailnews.tags.version", 2);
user_pref("network.cookie.prefsMigrated", true);
user_pref("pref.ldap.disable_button.edit_directories", false);
user_pref("security.disable_button.openCertManager", false);
EOF
cd ~
```

Importación de certificados en Mozilla Thunderbird La importación no manual requiere ser investigada: inicialmente parece ser que los certificados están en .icedove/<profile>/cert8.db, una base de datos tipo Berkeley DB v1.85:

```
cd /afs/${DOMAIN}/user/${SUBRUTA2NIVELES}/${NOMBRE}/
file .icedove/<profile>/cert8.db
apt-get install libdb1-compat
db_dump185 -p .icedove/<profile>/cert8.db | sed G | less
cd ~
```

Siendo la v1.85, no tenemos del todo claro cómo cargar datos desde un volcado (existe db_dump185 pero no db_load185). Nos decantamos por codificar cert8.db (con todos sus certificados más el de nuestra CA) tras haberlo importarlo desde la interfaz en base64 para poder ser decodificado ahora. Evidentemente, si ya tenemos una copia del fichero (por ejemplo de una decodificación anterior) simplemente bastará con copiarlo de un sitio a otro.

- Se usó 'cat cert8.db | gzip -9 | base64 -w 74'para codificar base64:

```
vim /var/tmp/cert8.db.b64
```

```
H4sIALIP9k4CA+3dB1hTV/sA8JtpkCGyRZDIENknCVtAEBBciDJUFDCEANFAIAlTCzLEhbNaEB
WjouDe4KwoKqJYHLROqigIrYNqtSp+6v/eBCmi9f/1aZ9qv+f91YTcc+899z13vCf3VK4YXZuL
YRgZw6iXMMwT/4ipYRgJU7wwRucHcudPivynfBamQXxekfDVXUyB3Pn6xzFIGAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA8K+mjz0f+nRo8tAnzg3Osc6tTis5MznN7ED2
HJaAdR7tRpZovZ2ZdaPVXXOaeTOzkvnMyMlo4IBWw4ED3gIAAAAAAAAAAAAAAAAAAAAAPg3YV
CIJ75bY1goXywIEsQmML2FXImEyWH6pkn5CdH8aGYoVyiI5koFogRmUNBoprcXhjVi2OTUSad1
85oiasyr2CssBk9H2WdZykiJTgmh0jXJIUEsXaRNTPTWVHtXtTVzRALPlmWEDIkZypo6XdsMFi
dLpMwAvjRVJJ7OckXOigVYg3kWTDZCjsz3q2DaMIeLxExusjROJBZk4DEmS/hMUYIwneWLvIl1
KZpuHzQoMDlKKOAxA8WCeK44nenNF0sFMQKeomVeirqk6Xjdfg4UlH3o72rNEOSiWIAdzBfHS5
iiGHm0XCkzTipNlLja2aWmptqm4CtL8JVteaJ4O3Eil4k3HTmy7BFb0RyrP3F8GGQMO/qvbgH5
z5yNDBLGkP9Ha8OsUQ6tGeVQ62QUMolMVv/gLFWlK1nm++c/VyWRaDTsf+qkRQN0VfEdzkLOSC
5MV5VFTDqxOQ4uDi5h/+ozAuWQjLsfPBINo+SQ+mB4eW9yDomEFd2Qhdxf/Vo3/W5xgpbh7slF
```

```
JnN0EuakrN7vPGS/LGNfL55Sv3uWc/03t66pQGGMQpUHW5p1BOIok+JXGcGDi06p2FReMa4489
JZt6bxOmt48XKtRSWP0mSVxg0jggvHtEgnLzvyQGh9q/V4RX/9kOkxDEqW785izmP1KuHKkctM
WYz4RrIHNyUtreJhptMO4a/zmiqbbmrXVYaunSXbTcsKcNJ+W2k3YDHdeWhgpWDwhv0xkY764/
29Y8bfWr9s7GW9qd/e/TmiXxLtWImBME5sJto7Q1Ibf1Tj7SovdsTTtsxTyhtGR18MODKIwX/V
r3BiWhPn2MM4j93mvqKqOw8KBc+nXH4+jkwhYaQNOaRL+B75Tn78DNWoOlSt/8wPPFRy0CxsRs
jssSsw3uKfJk73RhrEbE0S6S2Vgej4DzIJQ+5EGZPqiIjjQ8U/YsgSDaYzrOj4ldKLTNIx+MNj
y0uUKFY3UqwuQzLbfGticXxp39AgPi9ZzLfhiYXvr5bI43JsYh1s8RlIjVi9DxERFb9y6agvXW
lqvj8p/8UwEqmzKP5dLCQVKhdFrguXTUaT0AQUoqOEXxyxfLtYQQwaiIxQLzrNSo1C1qdqLW65
rJ+/beGi6dX1s6Y1+OtZz+hni8x0TDpDE4piRe+HlCKRlxE1WRAR9aUaI2bpAGTA0kd6irO2r/
xk5Xh7sZG9sw3Lxt4Jb/u7yEhUFrJDNu+mESl/YM/dIOJJEt/bKDIitmRC1UM6s7SyLvPLTjQ+
OPwLUvrV26WDvJPD4vRIWhTidP9mfUfWPfeHDVbfVWoPrRJTIhcc0lli9vxE7tWaZHtenEaQSt
C8GbWhSmXVjj4Baheiovbmd7StW2UfuzDIWNfj9bZHsbbed8KzOHlP1idWJeQUVtm/NjttkvXw
```

```
3q5TE7zz9vxo8/jCb/5GWkkT9mxjNRWamtRqhktbhrcrzSxpWsnqzUjvvzhprW7MGef6RwWFFf
SDkuHNl5cfb5m63yzZ5pqA9eD8N9UbUl63SX22aRwcc2xc28EGnwDzczuYFzMtZ1bw/EpSDhVe
0GlVFsY/9xZxlC1Qaseal88CDqTsPa+37yWz3f3GvbPbV10IuDRns+omH0lgY+Lkh9szRx4x/r
58UPOJ2UzVa3+mgyD9T39X6er+KI2YOsqhXEM55OOK7o9CdrzQs8cb9n7LdZCWouWqfnyRomny
hvdH/RSxaXSV+wlFUVwhkYvxjoaFEJvlwmZzXNiOeEfDJiadOyeR5x/t3Xd1de5d/IxXbKRP10
b+y2y/ZO+s480tw7cnTnU/1KLVWDI/YkFy2L2z8V4bTdpcZg/KKAtJLfZftyzTdYD+Y/OtG0a7
1J1z99uZvUNn2hSndl+v16+KC91k3mtO39IdeHH++tATlBF5Gtketb2Dp8YcLAhufnnv5fLHty
OT+z6xONicv373vO2bBuRa8Pb8EmwwWzmjtclWdWfp0fOtu8OuFNDC1EZJBi9IDre/mqrcJI26
k4Sirh6ZT2th8UYGz5vGSSsLCDE1LGzXb9Y43Qfp7bMf03TVZMjAyTaon5ma9HnpwFGyUmat8H
xRaNWT8kiPQcky1ReOc+dbh30Vizb3ubgy0TatadygnSEZpt8WuKimrvmRXK1y1EiR7bOvoewf
Pkidv+f+YenB/bnnQ6w8eN+7+U94tvaV73cj38s2VRlFcQvnvdpKi1bp+9WWr/gHztYFfLRzcF
Vkdw7C05vMRmaVb9GZ1oikHssXSYmDp+gLxEKJXaw0Vn6yyFO7fbfsOBgNQqbdsqNeZzXypNi9
```

```
no/lvIb2oLw2/aPrFlCrNj+LXKee5hu+Z8K0ANNz/ABW4dLiBtMXd8xjElmrj0cF01sLDV2Lq0
ot/HvvNK0pP1kcfHzm6LTRVnHW4bqbgtNGD78zq9rIOUnj0pWgu8GzAw+Qzz3iub4InTcm5LBt
1Rm91HyWsnLiL5kOuaWWmtGhsfbhOpGSbTPoptfHBsZbbr0wJsjrxKLDSZQRFvl69QOHjZZGjd
M+JSpvP2ne0p/TGDb3SpbGxTKumdH3t28a1uurX5WdiyzoUGk5lvS2JdzHcJPHHL8O/YDcXI2h
c2NWFKUXz75P9jtvdvcJnRqw8uwUWULG4cYNS/R9Oe3HPC5RT1bctpzN9/JQb+hxkWAk4vL+G6
5n4h/YKsL0MD1KOKaJZ5BJeAY5ocggStjDqQ+W5TtU5vTMIzO7bdc3iKWKlIkJhiZ1pJdvAEsN
qRCTvTRpIVH8DC6rD766PCy6t1eQ1/CJLA2krohHSVGAh8GyQOZd21Ai6ejzuPFciZQv9uRxJd
yYNNvo9IToBImtSBwrz0D4N1v8+60L4rBdiAxETHI6Jz9zdNlreyav7IX4vcY8cnY2dr592kXG
DTudEMG6irYjFWPqHB9fW7Wceflwy4JFR34+Mv1GZvpYWu/8xYnjDI5ev1m76+xXveq3HP5l6e
l59zwnR5g33IwJ+Jasedu/scXTY8nZIn0rC/+9pftME2M4iFTg6HhTU/ZskNtyH6XVyaSnI2/m
2a+t9lwSs0h4X4/R2tSgZf22M3PcQdm3umUKj6Uap25THm5eGXquZgYWdqpETWaKsncqUkV2Gc
ounfXRZdZlzyrN/Lx7O6fbOaoiT11UGqLgeatnCsF3fo3x9ApTttKY+VvrDN64Vbf0NVRzvfb1
```

```
2MhBnjXuN2I2PvpxRG2OQcy0wOSfdtokO50rvd8amvMklz5uSIz48STD4adXu7X957hNdPkW7X
APft70pUsTwh0S24cWOXCuXKxffef6pTN+GYyJRTdN2k6ust2+cer9lJR2Gvf5LYfXSya+UV47
pat9eM+t+IyRfo/+s+5GBh3DMolfGfi8YdD/eBZGhPi1PMTsJX8tRlNkrIixf7Qohc8TSW1Yth
9sj2WF3wN0i9ogUSSR/mHcn5qJKUbZDDDsE5vDsCwM6/N59z4epOEnG0L5Anc9Mdz19RcYF/lT
B7vrWzvVDDNAOdSBeL87X9Hnkvp8gX0ti43/4SBne2d5X8tiI+fOyS9wz3+yA955xGBMVctviy
8azSkWH9peX143ztXCucpT/0YAfxfzzKaclW/qEh3STrKS2T5nwxxGbEqKFE/Itd573SedTZqU
2Vw/tiynst83t55v7FvmMi7XcsPmke1zLxZpVxfdHnLqjHGcm+WDrNfCNSs2TDHZIvnNIcryzq
jykZ43zqiWLLV8ouKx8t1AzWr8ZqZIvt/wHouMsA+GPOie+FfnriJVqrmOmS9Xkm4zPsiL6cdP
4Iu5UvxuMogvTuGLu90h8rt167kXClPG9847lPpstujeW8PRV+d43f0XduuaRMBmVBXU+92tA4
UkD8awD7GjaDI0VjFY44+G5/T9sCq96OlCbtRHzqocXfz+OvEjMz7y3UGf1kvtQj/lxbIJx9tD
naPKtz16+quv5fKtTR19o1ZsjnQ9fuvMrCvbvJpvWu4K7dc8Izp79TAPkwzVA1vPO0x9tC2LYe
RWZ9Tbw6q6wG+ZyU9RsyqHepVbUW3XxDMubGkcHzyXsoBzryk0sG5HjTGb7+d7pcNhfyTl1Kld
NX5Ds07ff5k48VPZhPSZ+w0iBcehmPdvRbTlxw3fhoqPIFZAnKDErQirH9JVjJioE+OI0fgsHj
6LuOdjmSETRRAGXWv4C2LjmF4SSbKYm8Dj41HZcMifmvl7YqVPwgxRDj0E5dDcO/9vAGPcy0nZ
Xr5nj942vaNRe0KtZ54V/tUGWMlHDfEGGP9RjL6hzPEikbRzFAU5IXvE6RquZ7O7Tf5Tu/P/HW
s5xu1toW4U8cbeaRzjxYBXxg+53yqrJo4O5A1K1j7oNbew4heZ9mHG1NvD2mgW6nMc5w/ml7ez
eknjHQvtBg+PodZYpg/KWPn9our4sbKI7Ufcl56YNCPCUVgVVI8x2B7q0RPVNQU1Ko9WDxuHva
T5PDL9LsPoYUHfoxrlv2o/NchgCUj9y+6FVxULM5VKWsf7jFR3NWleVUHZWb563+S1DaO2Lyv4
bn9fyy1Oegv3nThwdQq94MVbk6MOEQ0zPTJz1ZZ+Qyuq3mN09wz//uYZ4TLruv+4XYt/WNcxyr
ugtuHARq7BHs+KwRq3Dr6IfHN5T227+74Bo2YajF7pWOJ+/0FZXIThrdiEiFedCZv8HL8t/rXH
aEs+vpdOKwbOc0hH8YmD+Gs/XVmetV8LSRSMjBeUItdu4+i2nWMePY+XnUQitOElSmzEfLxzE0
hF4nR80Xh8/eiutYnaQgfkkMZjXlgClo4xsWRMgvHxnyIsBn+XYnGYAC9hYt54qRifFuDlAoyH
cfHPxHI8fMkEfAlijhRfmygllufic3j450S8hIsvwetRKzHlg68Ti7/e1U1sJRCzk78HyetIwK
```

```
K7LT8efxfiUQrw8lh8OhBfgliPiNsLLxHj8/lYPP5KkNeWKo+eh78z8fUE+ByBvJz4zMWi8Heh
vCS927a48lqY8m0QbRPjLSDeFe0lloiTR8uXL8HEayHWJqZjOssVbbVFfbrGvTrvH//rgavuRx
FlL1aMlWXPQdl5yFPmIXPLd/19tIzz/jF/d2USF2bXdekbSuQIby/58NkHNdj/2Rq6j/jt9qgW
UF4c8yM1FA3SZ5DbedFW1d2+MAT+LJl3w+KV+uKSFhKz4cf0lf65zz82FBc+Nu9x+dzLG/YcWR
3fYjEitan1h5ubrzrsqE262XLj5hrTgUs21VeSDFY1TChceiVZZ6PGlYPutvPcXQTzf3Bwevl4
4MnJntsnbbaLdMUsvM2TKM1HETtk05s8NeVVzv2svWdu8z6WzGoxMfANqRQwlu6vv/L97ohHa5
c3/5BxUK/6wA/jeQbqhxbYxYfX0ZyfhtLi20setHEdVbRGzdYKU8kdazxl0rmC+UnXO06p502s
7/Xo7pITz15a1ISZJL2ctC/aau6uMseaYib3fkfA/h3T+zfffXDnmpvXiHJmafCzZQsPV4UwqZ
uu7tLVqbt74cw+95Zreacv/5YQn1ab7t2W4fR2isOIY+tOfbILIn3Qz/yjHQsDk/9tDolAlID1
xgqHtnvc8Jjocc1d053sftgtzU3LrWOIlhs8GwEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

```
AAAAAAAAAAAIAvgeJhVOoY1vN5gpgfhv1NjxSk/PVngBLPFBn216sh92ylovmaGPaRh7Fhc/D9
82U8jw2Psf8n5lI+/wNfPvMDgsgfOX6KY2uIYZ/8/XziOXef91f0Kf/Uw11ew79wAQAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAfHb/B+CM0q8AAAEA
```

```
cat /var/tmp/cert8.db.b64 | base64 -d | gunzip -c -d \
   | cat >  /var/tmp/cert8.db

cd /afs/${DOMAIN}/user/${SUBRUTA2NIVELES}/${NOMBRE}/
cp /var/tmp/cert8.db .icedove/w5s1nswf.default/cert8.db
cd ~
```

1.8. Comentarios a las consultas DNS a la luz de NSS

1.8.1. Cómo se resuelve un nombre de dominio

Sólo queremos recalcar cómo se resuelven los nombres en el sistema, conociendo ya cómo NSS hace de interfaz entre las fuentes de resolución y los programas, justificando de paso las acciones que se tomaron cuando instalamos bind9 y resolvconf. Esta información se ha obtenido de las páginas man y de pruebas efectuadas, no obstante, la última palabra

la tendría el código fuente de la implementación actual.

Cuando se pide la resolución de un host o una IP, las aplicaciones harán una llamada al sistema que corresponde a una función en la librería(s) C, tal como gethostbyname()/gethostbyaddr() o, más recientemente, getaddrinfo()/getnameinfo(). Entonces se inspecciona el /etc/nsswitch.conf:

```
hosts:          files mdns4_minimal [NOTFOUND=return] dns mdns4
```

- Anotación: en dklab1 y dklab2 sólo aparecían "files" y "dns", pero vamos a analizar esta solución mejor, por incluir la nuestra y ser muy típico en portátiles hoy en día.

Establece como primera fuente el fichero /etc/hosts. El formato allí es el siguiente:

```
#  ip             hostname              aliases list
#  --             --------              -------------
#1.1.1.1     nombre.example.com    nombre nombre2.example.org ...
```

El fichero existe, pero no tenemos nada allí, resultando la búsqueda en un NOTFOUND. Ante esta situación, la acción[42] por defecto es continuar con el siguiente sistema. El /etc/nsswitch.conf apuntaba entonces a Multicast-Dns v4 [43] mínimo [44] (bonjour/zeroconf/avahi[45]) que al no estar incluído en nuestro despliegue (pues disponemos directamente de DNS con Bind9) el resultado será UNAVAIL y por ello no tendrá en cuenta la acción [NOTFOUND=return] que le obligaría a parar de buscar, sino que continuará intentándolo con el siguiente sistema, DNS. La librería /lib/libnss_dns.so.2 utiliza a "Resolver", que es un conjunto de funciones de la librería C de comportamiento configurable. En concreto usa a /etc/host.conf (obbiado en parte, en debian sólo se aprovecha para indicar que se pueden devolver múltiples resultados del /etc/hosts por consulta, "multi on"). El otro fichero de configuración, el fundamental, es /etc/resolv.conf:

```
nameserver 10.168.1.1

nameserver 10.168.1.2

search casafx.dyndns.org.

#options rotate
```

Si la aplicación pidió al sistema NSS resolver un nombre y éste no está cualificado (no acaba en ".") y tiene sólo una componente, como por ejemplo "dklab1", ha de cualificarse.

[42] http://www.gnu.org/s/hello/manual/libc/Actions-in-the-NSS-configuration.html#Actions-in-the-NSS-configuration

[43] http://lists.alioth.debian.org/pipermail/pkg-utopia-maintainers/2008-February/002365.html

[44] http://0pointer.de/lennart/projects/nss-mdns/#documentation

[45] http://avahi.org/wiki/AboutAvahi

Si la variable HOSTALIASES existiese y apuntase a un fichero con pares "<nombre-de-un-componente><nombrecompleto>.", se tomaría la primera coincidencia de ahí. Si no, se concatena con los argumentos de la opción search en el resolv.conf, en el ejemplo se convertiría en "dklab1.casafx.dyndns.org.". A continuación, se lanza por fin una petición de resolución DNS a los servidores indicados por la directiva "nameserver". Si "options rotate" no está activo, se sigue el orden en que aparecen en el fichero. Si falla la primera pasada, podría cualificarse bajando un nivel y buscando por tanto a "dklab1.dyndns.org." (en general se baja mientras se conserven al menos los dos primeros niveles). Si hubiese más argumentos para la directiva "search", se repetiría el proceso anterior con todos ellos.

En algún momento puede haber éxito, la búsqueda para y el resultado se mandaría a la aplicación. Si nada diese resultado, nsswitch.conf aún indica, como última posibilidad, hacer una búsqueda multicast dns en cualquier versión de IP y en cualquier dominio, no sólo .local etc. Si se da otro NOTFOUND, se informa a la aplicación del error.

La resolución inversa es básicamente similar.

1.8.2. Cómo autoresuelve una máquina con Debian su propio FQDN

Vamos a explicar cómo forma una máquina su propio nombre y cómo explorarlo, ya que nuestro sistema es sensible a que las máquinas resuelvan correctamente su FQDN.

La máquina, en un contexto de redes de comunicaciones o no, tiene un nombre consultable con el comando "uname -n" o "hostname", y que es almacenado entre arranques en /etc/hostname. Cuando forma parte de una red, el nombre que debiera almacenar el administrador en /etc/hostname debiera tener alguna relación con su nombre en la red. Efectivamente se almacena su nombre sin el dominio al que pertenece (sin puntos); por ejemplo para dklab1.casafx.dyndns.org., el administrador debiera almacenar "dklab1" sin las comillas.

Para construir el nombre completo, Resolver procedería teóricamente así:

getaddrinfo(gethostname())

Es decir:

Lee su /etc/nsswitch.conf. La primera fuente es "files", es decir /etc/hosts, y el uso que se hace de este fichero es buscar la primera entrada (más largo, más preferencia) prefijada con la salida de "hostname". Si no hay coincidencia, nsswitch.conf apunta a MDNS, y después a DNS. El uso de DNS es, precisamente, buscar en el /etc/resolv.conf el argumento a la directiva "domain", o el primer argumento de la directiva "search", de forma que el FQDN se forma finalmente de unir la salida de "hostname" con el nombre de dominio apuntado por la directiva domain o search.

```
hostname              #...devuelve nombre de host.
hostname --fqdn       #...devuelve el nombre completamente cualificado
                      #   segu'n el proceso expuesto.
dnsdomainname         #...devuelve la diferencia de los dos anteriores
hostname --ip-address #...devuelve la resolucio'n inversa de
                      #   la salida de hostname.
```

LyX