ISTITUTO NAZIONALE DI FISICA NUCLEARE

Sezione di Bari

INFN/TC-05/04
18 Febbraio 2005

REALIZZAZIONE DI UN SISTEMA DI AUTENTICAZIONE ED AUTORIZZAZIONE CENTRALIZZATO BASATO SU KERBEROS 5 E OPENLDAP

Domenico Diacono

INFN-Sezione di Bari, Via Orabona 4, I-70126 Bari, Italy

Abstract

In questa nota tecnica si illustra dettagliatamente la procedura per la realizzazione di un servizio di autenticazione ed autorizzazione centralizzata, utilizzando rispettivamente Kerberos V e openLDAP, su sistema operativo GNU/Linux. Le indicazioni sono relative in particolare alla distribuzione Gentoo, ma sono facilmente adattabili a qualsiasi distribuzione.

Come esempio di servizio che fa uso della struttura creata viene descritta l'installazione di una nuova cella AFS direttamente autenticata dai KDC Kerberos, e la preparazione di una macchina di "public login", alla quale è possibile collegarsi usando un utente LDAP con la sua password Kerberos 5 e la home directory nella nuova cella AFS.

*Published by **SIS–Pubblicazioni***
Laboratori Nazionali di Frascati

INDICE

1 INTRODUZIONE

Per aumentare la sicurezza e la disponibilità dei servizi di rete offerti dai centri di calcolo è opportuno distribuire il lavoro su più macchine, ognuna dedicata ad un singolo servizio, piuttosto che accentrare più funzioni su un unico PC. La presenza di diversi servizi su di un'unica macchina se non aumenta la probabilità globale di attacco dall'esterno, dato che anche nel caso di servizi distribuiti questi sono in ogni caso erogati da altre macchine, aumenta però notevolmente l'impatto di una eventuale intrusione, dato che l'intruso avrebbe contemporaneamente sotto controllo più servizi.

Per semplificare la gestione degli utenti di queste macchine e dei relativi permessi di accesso ai servizi erogati può essere utile un servizio centrale di gestione. Questo servizio riveste evidentemente una importanza cruciale nella sicurezza della intera infrastruttura, e quindi la sua sicurezza deve essere estremamente curata.

In questa nota tecnica si illustra la procedura dettagliata per la realizzazione di un servizio di autenticazione ed autorizzazione centralizzata, utilizzando rispettivamente Kerberos V e openLDAP, su sistema operativo GNU/Linux. Come esempio di servizio che fa uso della struttura creata viene descritta l'installazione di una nuova cella AFS direttamente autenticata dai KDC Kerberos; non si segue, infatti, la procedura standard che prevede prima la installazione del servizio Kerberos 4 di AFS e poi la migrazione a Kerberos 5. Inoltre si preparerà una macchina di "public login", sulla quale è possibile collegarsi usando gli utenti definiti in LDAP, aventi la password registrata in Kerberos 5 e la home directory nella nuova cella AFS. Non si danno indicazioni pratiche dettagliate per installare dei server sicuri.

Nel seguito si prende come distribuzione di riferimento la Gentoo [1], ma nonostante ciò le indicazioni riportate, a meno di piccole variazioni, sono valide anche per altre distribuzioni. Nella appendice si riportano i file di configurazione più importanti, in modo da avere un punto di partenza per la propria installazione. Di seguito si riportano le versioni dei software utilizzati:

- MIT Kerberos 5 v.1.3.6
- OpenLDAP v. 2.1.30
- Cyrus SASL v. 2.1.19
- OpenAFS v. 1.2.10
- OpenSSH v. 3.9_p1

1 INSTALLAZIONE DI KERBEROS V

Esiste in rete una ampia documentazione sulla installazione di Kerberos [2], qui si riporta per comodità una breve guida riassuntiva dei passi da effettuare, senza approfondire gli aspetti teorici relativi al funzionamento del software.

1.1 Master

Durante l'installazione iniziale del server si deve installare il pacchetto MIT Kerberos

(app-crypt/mit-krb5); è necessario quindi modificare il file di configurazione */etc/krb5.conf*, un esempio del quale si trova in appendice, che contiene le informazioni di configurazione necessarie al funzionamento del server. Queste includono il realm di default e la posizione del Key Distribution Center (KDC) all'interno dei realm conosciuti. Nella installazione descritta non si è usata la possibilità di impiegare il DNS per distribuire queste stesse informazioni.

Il secondo file da modificare è */etc/krb5kdc/kdc.conf*, che specifica i dati di configurazione per il realm, usati dal servizio di autenticazione Kerberos (AS) e dal KDC. Dato che la cella AFS che verrà in seguito creata ed agganciata a Kerberos 5 dovrà rispondere correttamente sulla porta 7004 alle richieste del tipo *klog*, sarà necessario usare il servizio *fakeka*, presente nella distribuzione MIT: è però necessario cambiare il tipo di cifratura della master key da quella di default a

```
master_key_type = des-cbc-crc
```

Altrimenti *fakeka*, almeno fino alla versione di MIT provata, non funziona.

Si può creare ora il database e lo *stash file*, ossia il file che contiene una copia locale della chiave principale del database. Lo stash file è usato dal KDC per autenticare se stesso prima che i demoni *kadmind* e *krb5kdc* siano partiti, ed è molto importante quindi che sia protetto accuratamente.

```
#/usr/sbin/kdb5_util create -r BA.INFN.IT -s
```

Si crea una ACL di default all'interno del file */etc/krb5kdc/kadm5.acl*:

```
*/admin@BA.INFN.IT     *
*/*@BA.INFN.IT     i
```

In questo modo ogni principal con una istanza di amministratore ha tutti i privilegi, mentre tutti i principal che non hanno istanza di amministratore hanno la possibilità di fare richieste al database.

Può essere utile aggiungere delle policy distinte per gli amministratori e gli utenti, in modo da forzare durata e robustezza delle password per i due ruoli; si aggiungono quindi l'amministratore di sistema e un utente non privilegiato:

```
#kadmin.local
 >:addpol -maxlife "6 months" -minlength 8 -minclasses 3 -history 3 user
 >:addpol -maxlife "3 months" -minlength 10 -minclasses 3 -history 6 admin
 >:addprinc –policy admin +requires_preauth krbadm/admin
 >:addprinc –policy users domenico
 >:ktadd -k /etc/krb5kdc/kadm5.keytab kadmin/admin kadmin/changepw
```

L'ultima istruzione aggiunge la keytab di *kadmind*, ossia la chiave che i demoni di

amministrazione *kadmind4* e *v5passwdd* usano per decifrare i ticket Kerberos di amministratori o client per determinare se questi hanno o no accesso al database. Potrebbe non essere utile se non si farà uso di questi demoni.

E’ possibile ora far partire i demoni sul master, e aggiungerli al runlevel di default:

```
#/etc/init.d/mit-krb5kadmind start
#rc-update add mit-krb5kdc default
#rc-update add mit-krb5kadmind default
```

Per controllare che tutto sia andato bene si devono controllare i file di log */var/log/kadmin.log* e */var/log/krb5kdc.log*. Si ricordi che per fare in modo che il file di log delle autenticazioni sia ruotato è necessario modificare il file */etc/logrotate.d/krb5kdc* aggiungendo le istruzioni seguenti:

```
/var/log/krb5kdc.log {
        sharedscripts
        postrotate
                /bin/killall -HUP krb5kdc
        endscript
}
```

Il servizio ora è operativo: il comando *kinit* deve permettere l’acquisizione delle credenziali, verificabili con il comando *klist*.

1.2 Slave

La configurazione della parte slave deve essere fatta sia sulla macchina master sia sulla macchina che farà da slave fin dall’inizio, in modo che i due ruoli all’occorrenza possano essere facilmente scambiati. Le operazioni seguenti andranno quindi fatte innanzi tutto sul master. E’ necessario creare le host key per gli slave, e quindi anche per la macchina master, poi estrarre la keytab, che ogni KDC usa per decifrare i ticket:

```
#kadmin.local
 >:addprinc -randkey host/master.ba.infn.it
 >:addprinc -randkey host/slave.ba.infn.it
 >:ktadd host/master.ba.infn.it
```

Si può quindi predisporre il master per la propagazione del suo database allo slave per mezzo del demone *kpropd*. Si modifica il file */etc/krb5kdc/kpropd.acl* inserendo il nome dei KDC del realm:

```
host/master.ba.infn.it@BA.INFN.IT
host/slave.ba.infn.it@BA.INFN.IT
```

Per far partire il servizio di propagazione è necessario servirsi del demone *xinetd*, in appendice si riporta il file di configurazione:

```
#/etc/init.d/xinetd start
#rc-update add xinetd default
```

Se tutto è andato bene nel file di log */var/log/daemonlog* deve trovarsi una riga con:

```
Started working: 2 available services
```

I due servizi sono *kpropd* che gestisce la propagazione del database, e *eklogin* che consente di effettuare un rlogin criptato sul KDC. Nel file */etc/services* si aggiunge la riga relativa al servizio di propagazione, se non è già presente:

```
krb5_prop       754/tcp      # Kerberos slave propagation
```

Fatto ciò si può passare alla configurazione dalla macchina slave vera e propria. Su tale macchina deve essere naturalmente installato Kerberos con gli stessi file di configurazione *krb5.conf* e *kdc.conf* presenti sul master, e vanno ripetuti tutti i passi fatti fino ad ora in questo paragrafo, tranne l'aggiunta dei principal degli host KDC, che ora sono già presenti nel database.

1.3 Propagazione del database e fase finale della installazione

La propagazione del database consiste nell'invio, da parte del master, attraverso il servizio *kprop*, di una copia del suo database al server Kerberos dello slave. Per prima cosa si crea un dump del database, poi si propaga manualmente il dump sullo slave, infine si crea uno script che periodicamente controlla lo stato del database del master e lo propaga automaticamente in caso di sue variazioni:

```
#kdb5_util dump /etc/krb5kdc/slave_datatrans
#kprop -f /etc/krb5kdc/slave_datatrans slave.ba.infn.it
#nano –w /root/bin/push.sh
    #!/bin/bash
    X="`md5sum /etc/krb5kdc/principal`"
    while true; do
        X1="`md5sum /etc/krb5kdc/principal`"
        if [ "$X" != "$X1" ]; then
            X=$X1
            /usr/sbin/kdb5_util dump /etc/krb5kdc/slave_datatrans
            /usr/sbin/kprop -f /etc/krb5kdc/slave_datatrans slave.ba.infn.it
            logger "KERBEROS V database pushed"
        fi
        sleep 1
    done
```

```
#chmod 700 /root/bin/push.sh
#nano –w /etc/conf.d/local.start
    /root/bin/push.sh 2>&1 > /dev/null &
```

Infine si torna sullo slave per ultimare la sua installazione:

```
#kdb5_util stash
#/etc/init.d/mit-krb5kdc start
#rc-update add mit-krb5kdc default
```

Lo stash file serve per autenticare i demoni Kerberos senza usare la master password del database e deve quindi essere presente su ogni KDC.

2 INSTALLAZIONE DI OPENLDAP

Il lavoro fatto finora comprende solo la parte di autenticazione, in altre parole l'analogo del sistema *passwd/shadow* classico di Unix. Manca la autorizzazione, ossia la definizione degli utenti del sistema centralizzato, che sarà definita mediante openLDAP [3], usando tale sistema come un Network Information Service; anche in questo caso si vuole dare una indicazione pratica su come procedere, piuttosto che una trattazione completa. Riassumendo molto brevemente le basi teoriche, LDAP usa uno *schema* per definire quali *attributi* un *oggetto* può o deve possedere, definendo in sostanza delle *classi*. Un oggetto può implementare diverse classi: un utente ad esempio può essere un *posixAccount*, un *inetOrgPerson* e un *pilotPerson*. Ogni oggetto nella directory LDAP è individuato da un *Distinguished Name* (DN).

Nel seguito si è scelto di usare le stesse macchine KDC come server LDAP.

Nella installazione è fondamentale includere le *cyrus-sasl* [4]:

```
#emerge openldap
#emerge cyrus-sasl
#wget http://www.bayour.com/openldap/schemas/krb5-kdc.schema
```

Si è incluso tra gli schemi LDAP anche uno schema che permette di registrare oggetti con attributi legati a Kerberos. Il file va copiato nella directory */etc/openldap/schema*.

La configurazione del demone *slapd*, che risponde alle richieste di autenticazione per la directory LDAP, si trova nel file */etc/openldap/slapd.conf*, la cui versione definitiva si trova in appendice. In questa prima fase della installazione è sufficiente la seguente versione semplificata:

```
include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/inetorgperson.schema
```

```
include         /etc/openldap/schema/nis.schema
include         /etc/openldap/schema/krb5-kdc.schema
pidfile         /var/run/openldap/slapd.pid
argsfile        /var/run/openldap/slapd.args
loglevel        256
threads         8
idletimeout     14400
database        bdb
suffix          "dc=ba,dc=infn,dc=it"
rootdn          "cn=ldapadm,dc=ba,dc=infn,dc=it"
rootpw          {SHA}password
directory       /var/lib/openldap-data
```

La password si ottiene in formato SHA con il comando:

```
#slappasswd -h {SHA} -s "my password"
```

E' importante che il demone sia eseguito con i minimi privilegi necessari, quindi bisogna creare se già non esiste un utente ed un gruppo *ldap*, accertarsi che le directory */var/lib/openldap-data*, */var/lib/openldap-ldbm* e */var/lib/openldap-slurpd* appartengano a tale utente e abbiano i permessi 700, e che lo script di partenza usi le opzioni *–u ldap –g ldap*. Si può quindi far partire il server e verificarne il funzionamento con la richiesta:

```
# ldapsearch –LLL -x -b '' -s base '(objectclass=*)' namingContexts
```

2.1 TLS

Nella installazione standard i dati LDAP sono trasmessi tra client e server in chiaro; è possibile proteggersi da intercettazioni usando delle sessioni cifrate di comunicazione. I server e i client openLDAP possono, infatti, usare il Transport Layer Security (TLS), una evoluzione di SSL, per cifrare le comunicazioni tra gli host a livello TCP.

La prima cosa da fare per poter usare il framework TLS è richiedere alla autorità di certificazione i certificati per il demone *slapd*, usando come CN (common name) del server il FQDN (fully qualified domain name), corrispondente al record PTR del DNS del server. Chiave privata e certificato devono essere posti nella directory */etc/openldap/ssl/*, e bisogna modificare il file */etc/openldap/slapd.conf* inserendo le seguenti opzioni:

```
TLSCipherSuite HIGH
TLSCertificateFile /etc/openldap/ssl/slapd.pem
TLSCertificateKeyFile /etc/openldap/ssl/key.pem
```

E' necessario poi accertarsi che la chiave privata, così come il certificato del servizio, sia leggibile dal server, quindi dall'utente ldap, ma non sia né di proprietà di tale utente né scrivibile da esso, e inoltre non sia leggibile da nessun altro. Una soluzione può essere la seguente:

```
#chown root:ldap key.pem slapd.pem
#chmod 640 key.pem slapd.pem
```

2.2 SASL ed integrazione con Kerberos

OpenLDAP non supporta direttamente l'autenticazione Kerberos 5, ma riesce ad utilizzarla mediante il meccanismo SASL GSSAPI. Il protocollo SASL (Simple Authentication and Security Layer) fornisce un framework per integrare servizi di autenticazione esterni. Kerberos 5 può essere integrato mediante l'interfaccia GSSAPI.

Per rendere operativa questa integrazione bisogna modificare il file */etc/openldap/slapd.conf* includendo le seguenti istruzioni:

```
sasl-host          kerberos.ba.infn.it
sasl-realm         BA.INFN.IT
sasl-secprops      noplain,noactive,noanonymous
```

Come accade per ogni servizio di rete che usa l'autenticazione Kerberos, è necessario ora creare un principal per ldap. Per rendere automatica l'autenticazione del principal all'avvio del servizio si registra la chiave in un file di keytab. Per evitare che sia necessario rendere leggibile all'utente ldap il file di keytab principale, che è proprietà di root con permessi 600, si può usare un file dedicato:

```
#kadmin.local
     >:addprinc -randkey ldap/master.ba.infn.it
     >:ktadd -k /etc/openldap/ldap.keytab ldap/master.ba.infn.it@BA.INFN.IT
#chown root:ldap /etc/openldap/ldap.keytab
#chmod 640 /etc/openldap/ldap.keytab
```

Affinché all'avvio il demone slapd legga quel particolare file di keytab è necessario modificare il file di avvio */etc/init.d/slapd* aggiungendo nella sezione di partenza la riga:

```
export KRB5_KTNAME=/etc/openldap/ldap.keytab
```

Dato che si vuole utilizzare ldap solo mediante il servizio TLS si modifica il file */etc/conf.d/slapd* aggiungendo l'opzione:

```
OPTS="-h 'ldaps://'"
```

Se si usa un'altra distribuzione questa opzione deve essere aggiunta nella riga di partenza del demone.

2.2.1 Configurazione del client LDAP

Per il funzionamento dei comandi di interazione con la directory è necessario infine configurare la parte client di LDAP, modificando il file */etc/openldap/ldap.conf*:

```
BASE            dc=ba, dc=infn, dc=it
URI             ldaps://master.ba.infn.it
TLS_CACERT      /etc/openldap/ssl/INFNCA.pem
```

Nella sezione TLS_CACERT va inserita la chiave pubblica della autorità che ha rilasciato il certificato per *slapd*; LDAP userà questa chiave per verificare, alla connessione del client, la catena di certificazione. Ora si può verificare la funzionalità del server *slapd*: una semplice ricerca con *ldapsearch*, se non si posseggono credenziali Kerberos, darà come risultato un messaggio di errore (No credentials cache found), mentre se si ripete il comando dopo aver acquisito un ticket Kerberos si avrà il risultato

```
#ldapsearch
      SASL/GSSAPI authentication started
      SASL username: domenico@BA.INFN.IT
      SASL SSF: 56
      SASL installing layers
      No such object (32)
```

Il risultato “no such object” è naturalmente da attribuire al fatto che nella directory attualmente non esiste alcun oggetto. Si osservi che la cifratura riportata a 56 bit non corrisponde alla realtà: *klist –e* riporta correttamente il tipo utilizzato.

In sintesi la configurazione adottata permette il binding alla directory solo con le credenziali Kerberos, oltre che con lo username *ldapadm* che andrà in seguito disabilitato, e con connessione cifrata SSL.

2.3 Parametri del database

Il database di backend di LDAP può essere cambiato dal file di configurazione. Quello di default è il Berkley DB (BDB), che permette delle personalizzazioni per migliorare le prestazioni e la affidabilità. Il BDB è un database basato sulle transazioni e usa il logging “write-ahead” per assicurare la consistenza dei dati. Al commit di una transazione i record cambiati sono scritti nei file di log, ma le variazioni effettive al database non sono necessariamente scritte sul disco. Le variazioni che sono parte della transazione ultimata sono scritte nel file di database quando viene effettuato un “*checkpoint*” del file di log: a quel punto il log può essere archiviato e rimosso dal sistema. E’ chiaro che maggiore sarà l’intervallo tra due checkpoint, maggiori le probabilità che le modifiche del database in caso di problemi al sistema saranno non recuperabili.

La directory nella quale risiede il database è indicata in */etc/openldap/slapd.conf* alla voce *directory*, e nella impostazione in esame si tratta di */var/lib/openldap-data/*. All’interno

di tale directory è possibile creare il file DB_CONFIG:

```
set_cachesize      0 8388608 1
set_lg_max         10485760
set_lg_bsize       2097152
```

Queste istruzioni portano ad 8 MB la cache del database, a 2 MB la cache dei file di log e a 10 MB la loro massima dimensione.

Altre opzioni di configurazione del database si possono impostare nel file *slapd.conf*. Le seguenti direttive:

```
cachesize     10000
checkpoint    256  15
```

specificano il numero di entry che *slapd* mantiene in cache, e la frequenza dei checkpoint (ogni 256 Kbyte scritti od ogni 15 minuti).

La velocità delle operazioni del database è notevolmente influenzata dalla definizione degli indici. Per il compito che dovrà svolgere la installazione LDAP qui descritta un insieme di indici da inserire in *slapd.conf* potrebbe essere il seguente:

```
index     uid,uidNumber,gidNumber,memberUid     eq
index     cn,sn                                 pres,eq,sub
index     krb5PrincipalName,objectClass         pres,eq
```

2.4 Installazione del server slave e replica.

Per garantirsi una disponibilità maggiore del servizio slapd ed eventualmente anche una distribuzione del carico tra vari server è possibile configurare un’altra macchina che ospiti un servizio *slapd*. La configurazione iniziale è identica a quella vista finora per il server LDAP principale. Nel caso in esame si usa la macchina KDC slave come secondo server LDAP, sostituendo quindi in tutti i file di configurazione descritti nel paragrafo precedente “master” con “slave”. La conferma che i due server funzionano simmetricamente si ha lanciando il comando:

```
host1#ldapsearch –H ldaps://host2.ba.infn.it/
```

con host1 e host2 alternativamente “master” e “slave”. Una volta fissata la configurazione di base ci si porta sul master per configurare la replica del database. Il demone *slurpd* periodicamente controlla un logfile, generato ad ogni aggiornamento di LDAP, e lo propaga sullo slave. Le richieste di lettura della directory possono quindi essere soddisfatte da ambedue i server, gli aggiornamenti possono essere fatti solo sul master: una richiesta di aggiornamento allo slave produrrà un reindirizzamento della richiesta al master. La distribuzione di carico non è però implementata nativamente tra i vari server ldap.

2.4.1 Master

Per configurare il server LDAP come master è necessario modificare il file *slapd.conf*, aggiungendo una direttiva *replogfile* nella sezione generale:

```
replogfile      /var/lib/openldap-data/master-slapd.replog
```

Questo sarà il file di log letto da *slurpd*, nel quale *slapd* scrive i cambiamenti apportati alla directory. Per ogni server slave si aggiunge inoltre una direttiva *replica*, nella sezione del database dello stesso file:

```
replica          uri=ldaps://slave.ba.infn.it:636
     authcId=”host/master.ba.infn.it@BA.INFN.IT”
     bindmethod=sasl  saslmech=GSSAPI
```

Si noti che è il server master che fa il “push” del database allo slave, identificandosi con la sua identità di server. Il vantaggio di tale approccio sta nel fatto che l’autenticazione avviene automaticamente attraverso il keytab principale, senza la necessità di mantenere attive in qualche modo le credenziali di un utente apposito. L’autenticazione necessaria per autorizzare la replica avviene, infatti, tramite SASL GSSAPI, quindi è comunque necessario avere un principal Kerberos valido per poter connettersi allo slave e replicare il database LDAP.

2.4.2 Slave

Sullo slave non vanno in nessun caso utilizzate le direttive precedenti, la creazione di catene di repliche è sconsigliata, e quindi non è necessario scrivere i cambiamenti in un file di log. Nel file *slapd.conf* invece bisogna includere la seguente direttiva nella sezione del database:

```
updatedn "uid=host/master.ba.infn.it,cn=ba.infn.it,cn=gssapi,cn=auth"
```

L’uid usato è quello autorizzato a scrivere gli aggiornamenti nel database, deve coincidere quindi con quello della direttiva *replica* nel master e deve avere i permessi di scrittura nel database. Dato che di default tutti gli uid hanno solo la possibilità di leggere la directory bisogna impostare la seguente ACL:

```
access to *
     by dn="uid=host/master.ba.infn.it,cn=ba.infn.it,cn=gssapi,cn=auth" write
     by * read
```

Si sottolinea che è importante verificare che l’uid usato nella ACL e in *updatedn* sia coincidente con quello della direttiva *replica* del master, espresso nel namespace LDAP.

Infine è necessaria nel file *slapd.conf* dello slave una direttiva che indichi ai client a quale server rivolgersi per gli aggiornamenti della directory:

```
updateref ldaps://master.ba.infn.it/
```

Il server *slurpd* è un client ldap, e quindi al contrario di quanto accade per *slapd* che è un servizio registrato nel database Kerberos e ha un suo file di keytab, necessita di un ticket aggiornato per autenticarsi. Dato che è eseguito con i privilegi di root può essere configurato per utilizzare il principal host/bakdcmaster.ba.infn.it, la cui chiave è già nel file di keytab di default, e si deve quindi solo aggiungere un cronjob che rinnovi il ticket. Per determinare l'intervallo da utilizzare per rinnovare il ticket si può eseguire il comando:

```
#kadmin.local –q "getprinc host/master.ba.infn.it" | grep ticket
```

Nel caso in esame la "Maximum ticket life" è stata impostata a 24 ore, quindi è sufficiente rinnovare il ticket ogni 23 ore. Sul master:

```
#nano –w /etc/cron.d/updateslurpd.sh
      #!/bin/bash
      KRB5CCNAME="FILE:/var/run/slurpd.krbenv" /usr/bin/kinit –k host/master.ba.infn.it
#chmod u+x /etc/cron.d/updateslurpd.sh
#crontab –e
      0  */23 * * *  /etc/cron.d/updateslurpd.sh 2>&1 > /dev/null
```

Lo script di partenza di *slurpd* sul master dovrà essere modificato per assicurarsi che richieda il TGT del pricipal che viene usato per la replica, e punti alla corretta cache delle credenziali:

```
start() {
      ebegin "Starting slurpd"
      export KRB5CCNAME="FILE:/var/run/slurpd.krbenv"
      /usr/bin/kinit -k host/master.ba.infn.it
      start-stop-daemon --start --quiet \
      --exec /usr/lib/openldap/slurpd
      eend $?
}
```

Ora si può far partire il server *slurpd*. Per verificare che la parte Kerberos funzioni si può usare il seguente comando per leggere la cache delle credenziali usate da *slurpd*:

```
#klist -c /var/run/slurpd.krbenv
```

Si deve porre attenzione al fatto che la *kvno* del principal usato deve coincidere con quella presente nella keytab, pena la impossibilità di ottenere il ticket e autenticarsi, solitamente con errori difficilmente interpretabili. Per verificare che le *kvno* siano corrette:

```
#kadmin.local
     >:getprinc host/bakdcmaster.ba.infn.it
#ktutil
     >:rkt /etc/krb5.keytab
     >:l
```

Una verifica della funzionalità di replica si può effettuare con il seguente file di prova *start.ldif*:

```
dn: dc=ba,dc=infn,dc=it
objectclass: organization
objectclass: dcObject
objectClass: top
o: INFN
dc: ba
description: INFN sezione di Bari
```

Per inserirlo nel database si usa il comando:

```
#ldapadd -x -D cn=ldapadm,dc=ba,dc=infn,dc=it -w passwd -f start.ldif
     adding new entry "dc=ba,dc=infn,dc=it"
```

Per cancellare l’oggetto:

```
#ldapdelete -r -x -D cn=ldapadm,dc=ba,dc=infn,dc=it -w passwd -dn
dc=ba,dc=infn,dc=it
```

Si può verificare l’avvenuta propagazione nel file */var/log/syslog* dello slave, ma anche interrogando i due server separatamente con *ldapsearch –LLL*.

2.5 Popolazione del database

La struttura scelta prevede la divisione in gruppi di ricerca e servizi. Ogni corrispondente unità organizzativa include sia server host che persone, e potrà contenere unità organizzative di livello inferiore per una eventuale divisione per esperimenti. Il file *base.ldif* in appendice contiene un esempio della struttura LDIF necessaria, che può essere replicata per i gruppi voluti copiando le ultime 4 unità organizzative e sostituendo ou=gruppo1 con la nuova unità. L’inserimento avviene usando i comandi visti nel paragrafo precedente. Il file non

contiene utenti, per aggiungerli si può usare il file *user.ldif*, che contiene un utente amministratore di sistema, due utenti comuni appartenenti a gruppi differenti ed un amministratore di gruppo. Questi stessi utenti devono ovviamente essere aggiunti al database Kerberos.

Una volta impostata la struttura si può rimuovere dal file *slapd.conf* di ambedue i server la possibilità di autenticarsi con l'utente *rootdn*, commentando le righe relative e facendo ripartire il server. Gli utenti che si sono inseriti sono sufficienti per provare la funzionalità della directory.

La divisione effettuata è funzionale alla futura delega della amministrazione di ogni unità organizzativa ai rispettivi amministratori. Si vuole ottenere il seguente comportamento:

1-L'amministratore di sistema può cambiare ogni caratteristica di ogni utente

2-L'amministratore del gruppo*n* può aggiungere, modificare o cancellare utenti nel gruppo*n*, ma non negli altri gruppi

3-L'utente può cambiare alcune sue caratteristiche, come e-mail e shell di login, che devono restare private e non accessibili agli utenti anonimi. Infatti, affinché sia possibile autenticare un utente di rete nella struttura descritta deve essere possibile il binding anonimo su TLS, quindi anche se cifrate le informazioni presenti nella directory sono disponibili in lettura a chiunque se non si filtra l'accesso opportunamente.

Questo tipo di suddivisione dei compiti si ottiene impostando le ACL nel file */etc/openldap/slapd.access*, poi incluso in *slapd.conf* dalla direttiva

```
include         /etc/openldap/slapd.access
```

Naturalmente questa costituisce solo la parte LDAP del problema: una analoga delega dei compiti deve essere realizzata anche per AFS, ma il problema non sarà affrontato in questa nota. Una strada che si potrebbe seguire, ma che non si è analizzata in dettaglio, consiste nel permettere la connessione via ssh sui server amministrativi solo tramite autenticazione GSSAPI. In questo modo si conoscerebbe l'identità dell'utente prima di eseguire il comando richiesto, e si potrebbe dunque negare l'esecuzione basandosi su ACL simili a quelle in *slapd.access*.

Le ACL, come si vede in Appendice, sono differenti tra master e slave, dato che nello slave l'unico utente che può scrivere è quello abilitato alla replica del database. La corretta interpretazione delle ACL non può prescindere nel caso in esame dalla applicazione di una direttiva sasl-regexp. Il meccanismo di autenticazione del server slapd userà le chiamate della libreria SASL per ottenere lo *username* dell'utente nello spazio dei nomi del meccanismo di autenticazione, in questo caso Kerberos 5, ma non in quello di LDAP. E' quindi necessario che l'amministratore dica al server slapd come trasformare un DN di una richiesta di autenticazione (che sarà del tipo uid=<username>, cn=<realm>, cn=<mechanism>, cn=auth)

in un DN LDAP.

```
sasl-regexp    uid=(.*),cn=BA.INFN.IT,cn=GSSAPI,cn=auth
        ldaps:///dc=ba,dc=infn,dc=it??sub?uid=$1
```

Questo tipo di direttiva, da impostare nel file *slapd.conf* sia sul server master che sullo slave, esegue una vera e propria query su LDAP alla ricerca dell'uid: se la ricerca restituisce esattamente un oggetto è accettata, altrimenti l'autenticazione fallisce.

Per verificare i diritti degli utenti e la corretta replicazione si può usare *ldapmodify*. Creando un file *change.ldif*, ad esempio, come segue:

```
dn: uid=utente1,ou=people,ou=gruppo1,dc=ba,dc=infn,dc=it
changetype: modify
replace: loginShell
loginShell: /bin/tcsh
```

si può cambiare la shell con il comando

```
#ldapmodify –f change.ldif
```

L'esecuzione del comando deve fallire a meno che non si posseggano le credenziali Kerberos di amministratore del sistema o del gruppo dell'utente, o dell'utente stesso.

3 INSTALLAZIONE DI OPENAFS

In questo capitolo si illustrerà come installare la prima macchina della nuova cella AFS [5], sia come server che come client. Per la cella in esame, che è costruita da zero, non si è installato il kaserver, ma si è usato direttamente il servizio Kerberos 5 per registrare le password degli utenti. La macchina deve avere spazio su disco sufficiente per i volumi AFS. Per sfruttare in seguito le caratteristiche di AFS è conveniente installare i file utente in volumi AFS, in modo che poi quando saranno aggiunte macchine addizionali nella cella questi volumi possano essere distribuiti tra le macchine. Si configurerà la macchina iniziale sia come *file server* che come *database server*.

3.1 Installazione del client

Per essere in grado durante la procedura di installazione del server di verificarne il funzionamento per prima cosa si installa il client AFS.

Nella distribuzione Gentoo si usano i transarc-path, quindi la cache si trova per default nella directory */usr/vice/cache*, e la sua dimensione può essere cambiata nel file */etc/afs/cacheinfo*. La directory */usr/vice/etc* è un link a */etc/afs* e contiene i file relativi alla

configurazione del client e il modulo del kernel, i binari in user space sono in */usr/afsws* e i file di configurazione del server sono in */usr/afs/etc*. La directory di cache deve essere montata su una partizione ext2:

```
#mkdir /usr/vice/cache
#nano -w /etc/fstab
      /dev/sda#      /usr/vice/cache        ext2    defaults        0 0
#fdisk /dev/sda
#mke2fs /dev/sda#
#echo “infn.it” > /usr/vice/etc/ThisCell
#nano -w /usr/vice/etc/CellServDB
      >infn.it #Istituto Nazionale di Fisica Nucleare (INFN), Italia
      141.108.3.252          #afsrm1.roma1.infn.it
      192.84.134.75          #afsna.na.infn.it
      131.154.1.7           #afscnaf.infn.it
```

Bisogna poi aprire il firewall creando una catena dedicata af AFS da inserire tra quelle in uscita, un esempio può essere il seguente:

```
iptables -N allow-afs-traffic-out
iptables -F allow-afs-traffic-out
iptables -A allow-afs-traffic-out -p udp --dport 7000 -j ACCEPT
iptables -A allow-afs-traffic-out -p udp --dport 7001 -j ACCEPT
iptables -A allow-afs-traffic-out -p udp --dport 7002 -j ACCEPT
iptables -A allow-afs-traffic-out -p udp --dport 7003 -j ACCEPT
iptables -A allow-afs-traffic-out -p udp --dport 7004 -j ACCEPT
iptables -A allow-afs-traffic-out -p udp --dport 7005 -j ACCEPT
iptables -A allow-afs-traffic-out -p udp --dport 7006 -j ACCEPT
iptables -A allow-afs-traffic-out -p udp --dport 7007 -j ACCEPT
iptables -A allow-afs-traffic-out -p udp --dport 7008 -j ACCEPT
iptables -A allow-afs-traffic-out -p udp --dport 7021 -j ACCEPT
iptables -A allow-afs-traffic-out -p udp --dport 750 -j ACCEPT
iptables -A allow-afs-traffic-out -p tcp --dport 750 -j ACCEPT
iptables -A allow-afs-traffic-out -p udp --dport 88 -j ACCEPT
iptables -A allow-afs-traffic-out -p tcp --dport 88 -j ACCEPT
iptables -A allow-afs-traffic-out -p udp --dport 464 -j ACCEPT
iptables -A output -j allow-afs-traffic-out
```

In seguito volendo si potrà rimuovere la funzionalità client dalla macchina. Una volta fatto partire il servizio AFS con

```
#/etc/init.d/afs start
```

si potrà esaminare il contenuto della cella infn.it e delle altre celle da questa visibili.

3.2 Configurazione del server

Un file server AFS deve possedere almeno una partizione o un volume logico dedicato ai volumi AFS. Ogni partizione deve essere montata su directory chiamate ***/vicep***xx, dove *xx* è formato da una o due lettere minuscole. Le directory ***/vicep***xx devono essere sottodirectory della principale. Dato che su Linux viene usato il fileserver *namei*, che usa le normali strutture di file e directory per immagazzinare i dati, è possibile formattare le partizioni **/vicep***xx* con filesystem avanzati, come XFS.

```
#mkdir /vicepa
#nano -w /etc/fstab
    /dev/sdb1 /vicepa reiserfs defaults 0 0
#mkfs.xfs -f /dev/sdb1
#mount -a
```

Ora si è pronti ad installare i servizi AFS. Innanzitutto si ferma il client e lo si rende parte della cella in costruzione modificando i file seguenti:

```
#echo "ba.infn.it" > /usr/vice/etc/ThisCell
#echo ">ba.infn.it #Cella ba.infn.it" >> /usr/vice/etc/CellServDB
#echo ″193.206.185.235  #baafsserver.ba.infn.it″>>                    \
/usr/vice/etc/CellServDB
```

Si avvia quindi il processo bosserver con l’opzione noauth:

```
#/usr/afs/bin/bosserver -noauth &
```

La cella manca di meccanismo di autorizzazione, perciò bisogna impedire l’accesso dall’esterno, ad esempio con il firewall o staccando il cavo di rete, e non lasciare la macchina incustodita con il processo *bosserver* attivo. La prima attivazione di bosserver crea i file *ThisCell* e *CellServDB* nella directory */usr/afs/etc/*, che sono relativi alle operazioni del server. Si tratta di file generici che mancano del nome della cella reale, che deve essere impostato mediante il comando seguente:

```
#bos setcellname -server baafsserver.ba.infn.it -cell ba.infn.it -name
ba.infn.it -noauth
```

Il comando *listhosts* permette di verificare che la procedura sia andata a buon fine:

```
#bos listhosts baafsserver.ba.infn.it -noauth
     Cell name is ba.infn.it
              Host 1 is baafsserver
```

Ora si possono creare i processi server. Dato però che non sarà utilizzato il Kerberos 4 nativo di AFS, non si crea il processo *kaserver*.

```
#bos create baafsserver.ba.infn.it ptserver simple
/usr/afs/bin/ptserver -cell ba.infn.it -noauth
#bos create baafsserver.ba.infn.it vlserver simple
/usr/afs/bin/vlserver -cell ba.infn.it -noauth
#bos create baafsserver.ba.infn.it buserver simple
/usr/afs/bin/buserver -cell ba.infn.it -noauth
#bos create baafsserver.ba.infn.it fs fs /usr/afs/bin/fileserver
/usr/afs/bin/volserver /usr/afs/bin/salvager -cell ba.infn.it -noauth
```

3.2.1 Interazione kerberos5 - openAFS

All’interno del realm Kerberos è necessario creare un principal AFS. Per evitare malfunzionamenti si consiglia di usare per questo principal solo la chiave compatibile con AFS, quindi per creare il principal ed estrarre la sua keytab si devono utilizzare i comandi:

```
KDC#kadmin.local -e des-cbc-crc:v4 -q "ank -randkey afs/ba.infn.it"
KDC#kadmin.local -e des-cbc-crc:v4 -q "ktadd -k /etc/krb5.keytab.afs
afs/ba.infn.it"
```

Affinché il login funzioni il realm deve supportare *krb524*, ossia deve essere in grado di rispondere alle richieste del tipo Kerberos versione 4 e convertire i TGT versione 5 in token AFS. Il servizio krb524 usa un nome di applicazione particolare (afs_krb5) nella sezione appdefaults di *krb5.conf* per sapere se è possibile usare il nuovo formato dei token AFS basati su Kerberos 5 invece del vecchio formato che utilizza una versione convertita dei ticket Kerberos 4. Il nuovo formato assicura la possibilità di autenticazione cross-realm senza introdurre falle nella sicurezza ed è usato per default.

Naturalmente anche sulla macchina AFS server deve essere presente il file *krb5.conf*. Su ogni KDC va installato uno script di avvio, se si usa lo stash file e i servizi krb5kdc e k5admind partono al boot, in modo da avviare anche il servizio krb524d. Inoltre è necessario aprire sia sui KDC che sul file server AFS la porta udp 4444.

Si torni ora sul file server AFS e si copi il file */etc/krb5.keytab.afs* creato sul KDC in modo sicuro. E’ necessario inserire la chiave des-cdc-crc del principal afs@BA.INFN.IT, presente nel keytab appena copiato, nel KeyFile necessario alla operatività del server AFS. Sono di aiuto per questo compito i programmi del migration kit AFS-krb5. Non ne esiste una versione inclusa in portage, ma un semplice script di ebuild può essere scritto in modo da

permettere la loro compilazione, come dall'esempio in appendice, o si possono compilare a mano, scaricandoli dal sito indicato nello script stesso, avendo cura di effettuare le stesse modifiche qui descritte. Per utilizzare lo script deve essere creata la directory */usr/portage.local/app-crypt/afs-krb5*. Si inserisce quindi nel file */etc/make.conf* la riga

PORTDIR_OVERLAY="/usr/portage.local"

E' necessario procedere come segue:

```
#cd /usr/portage.local/app-crypt/afs-krb5
#ebuild afs-krb5-2.0.ebuild fetch
#ebuild afs-krb5-2.0.ebuild unpack
```

Ora si può modificare il file `/usr/tmpportage/portage/afs-krb5-2.0/work/afs-krb5/src/configure`: si devono eliminare tutti i riferimenti alla libreria kr524, che è stata incorporata nella krb5, sostituendo la riga

KRB524LIB=-lkrb524

con

KRB524LIB=-lkrb5

Quindi si può proseguire:

```
#ebuild afs-krb5-2.0.ebuild compile
#ebuild afs-krb5-2.0.ebuild install
#ebuild afs-krb5-2.0.ebuild qmerge
```

Con il migration kit installato la chiave */usr/afs/etc/KeyFile* per il funzionamento del principal AFS si ottiene con

```
#asetkey add n krb5.keytab.afs afs
```

dove n è il numero di versione della chiave, ossia l'output del comando

```
KDC#kadmin.local -q "getprinc afs/ba.infn.it" | grep ^Key
```

Se tutto è andato bene deve essere possibile far ripartire i demoni AFS con:

```
#bos restart baafsserver.ba.infn.it -all -cell ba.infn.it -noauth
```

La successiva analisi dello stato del server deve dare il seguente risultato:

```
#bos status baafsserver.ba.infn.it  -noauth
    Instance ptserver, currently running normally.
    Instance vlserver, currently running normally.
    Instance buserver, currently running normally.
    Instance fs, currently running normally.
      Auxiliary status is: file server running.
```

Dopo aver rimosso il file *krb5.keytab.afs* è necessario creare un amministratore per AFS. Innanzitutto si crea il principal Kerberos, che ne conterrà la password:

```
KDC#kadmin.local -q “addprinc -policy admin afsadm/admin”
```

Naturalmente impostare i diritti di amministratore in Kerberos non influisce sui diritti di amministratore in AFS. Per creare l’utente afsadm in AFS e aggiungerlo al gruppo system:administrators si usano i seguenti comandi:

```
#pts createuser -name afsadm.admin -cell ba.infn.it -id 2001 -noauth
#pts adduser afsadm.admin system:administrators -cell ba.infn.it -noauth
#bos adduser -server baafsserver.ba.infn.it -cell ba.infn.it -user afsadm.admin -noauth
```

Si è dato l’ID 2001 in modo da permettere di far eventualmente coincidere in futuro l’ID di utenze unix o ldap con l’id di AFS. Per verificare che l’utente sia stato correttamente aggiunto ai database rispettivamente del gruppo di protezione system:administrators e della lista dei SuperUtenti del server si eseguono i comandi:

```
#pts mem system:administrators -cell ba.infn.it -noauth
#bos listusers baafsserver.ba.infn.it -cell ba.infn.it -noauth
```

Infine si completa l’inizializzazione creando il volume di partenza *root.afs* nella partizione */vicepa*. Per verificare che il processo volserver abbia correttamente riconosciuto la partizione:

```
#vos partinfo baafsserver.ba.infn.it -cell ba.infn.it -noauth
```

Quindi si procede alla creazione del volume root.afs e si fa ripartire il client:

```
#vos create baafsserver.ba.infn.it /vicepa root.afs -cell ba.infn.it -
noauth
#/etc/init.d/afs start
```

Il ramo */afs* è inaccessibile agli utenti normali, solo l'amministratore della cella può accedervi. Per poter operare come amministratore in AFS bisogna prima avere le credenziali di afsadm/admin, poi richiedere il token:

```
#kinit afsadm/admin
#aklog
```

Se tutto è andato bene l'output dei comandi di controllo klist e tokens deve essere come segue:

```
#klist
    Default principal: afsadm/admin@BA.INFN.IT
    Valid starting     Expires           Service principal
    11/23/04 18:14:31  11/24/04 04:14:32  krbtgt/BA.INFN.IT@BA.INFN.IT
        renew until 11/30/04 18:14:31
    11/23/04 18:14:41  11/24/04 04:14:32  afs/ba.infn.it@BA.INFN.IT
        renew until 11/30/04 18:14:31
    Kerberos 4 ticket cache: /tmp/tkt0
    klist: You have no tickets cached

#tokens
    Tokens held by the Cache Manager:
    User's (AFS ID 2001) tokens for afs@ba.infn.it [Expires Nov 24 04:14]
      --End of list--
```

Si noti come non è presente alcun ticket Kerberos 4, aklog ottiene il token AFS a partire dal ticket kerbero5 mediante il servizio krb524. Il comando *aklog -d* fornisce informazioni di debug utili per la risoluzione di eventuali problemi.

3.2.2 Creazione della cella

Per proseguire nella installazione ci si deve autenticare come amministratore. Il processo inizia con la creazione del volume di ingresso root.cell, che è montato come */afs/ba.infn.it* e al quale bisogna garantire accesso in lettura a tutti:

```
#vos create baafsserver.ba.infn.it a root.cell
#fs mkm /afs/ba.infn.it root.cell
#fs sa /afs system:anyuser read
```

```
#fs sa /afs/ba.infn.it system:anyuser read
```

Entrambi i volumi root.cell e root.afs devono avere repliche Read-Only; inoltre si deve creare il punto di ingresso nell'albero Read-Write attraverso un mount-point speciale:

```
#vos addsite baafsserver.ba.infn.it a root.afs
    Added replication site baafsserver.ba.infn.it /vicepa for volume root.afs
#vos addsite baafsserver.ba.infn.it a root.cell
#fs mkm /afs/.ba.infn.it root.cell -rw
#vos release root.afs
    Released volume root.afs successfully
#vos release root.cell
```

Dopo aver modificato il file */etc/afs/afs.conf* inserendo la riga

```
AFS_SERVER=on
```

si consiglia un reboot della macchina. Riavviata la macchina si può far partire AFS ed aggiungerlo al runlevel, contollando poi lo stato dell'albero AFS tramite il comando

```
#fs exam /afs
    Volume status for vid = 536870913 named root.afs.readonly
```

Per aggiungere altre celle sotto il ramo /afs si deve innanzitutto disporre degli IP dei loro fileserver nel file */usr/vice/etc/CellServDB*. Dato che si è replicato il volume root.afs, il Cache Manager è programmato per accedere alla sua versione read-only (root.afs.readonly) ogni volta che questo è possibile. Per modificare il contenuto del volume root.afs (ad esempio appunto quando si sta montando un volume root.cell di un'altra cella al secondo livello del filespace AFS), è necessario montare il volume root.afs temporaneamente, effettuare i cambiamenti, rilasciare il volume e rimuovere il mount point temporaneo:

```
#fs mkm /afs/.ba.infn.it/TEMP root.afs
#fs mkm /afs/.ba.infn.it/TEMP/caspur.it root.cell -cell caspur.it
#fs mkm /afs/.ba.infn.it/TEMP/infn.it root.cell -cell infn.it
#fs mkm /afs/.ba.infn.it/TEMP/cern.ch root.cell -cell cern.ch
#vos release root.afs
#fs rmm /afs/.ba.infn.it/TEMP
```

3.2.3 Creazione di un nuovo utente

La creazione di un nuovo utente AFS non può più prescindere dalla creazione di un principal Kerberos 5: si deve innanzitutto aggiungere un principal nel KDC, poi procedere

come segue per la creazione del volume dedicato agli utenti e della home directory:

```
#vos create baafsserver.ba.infn.it a user
#fs mkmount /afs/.ba.infn.it/user user
#fs setacl /afs/.ba.infn.it/user system:anyuser read
#vos addsite baafsserver.ba.infn.it a user
#vos release user
#vos release root.cell
#fs checkv

#pts createuser domenico
#vos create baafsserver.ba.infn.it a user.domenico
#fs mkmount /afs/.ba.infn.it/user/domenico user.domenico
#vos release user
#fs checkv
#mkdir /afs/ba.infn.it/user/domenico/public
#mkdir /afs/ba.infn.it/user/domenico/private
#cd /afs/ba.infn.it/user/domenico/
#chown -Rh 2002:5000 .
#fs setquota /afs/ba.infn.it/user/domenico 300000
#fs setacl /afs/ba.infn.it/user/domenico domenico all system:anyuser l
-clear
#fs sa /afs/ba.infn.it/user/domenico/public domenico all system:anyuser
read system:administrators i -clear
#fs sa /afs/ba.infn.it/user/domenico/private domenico all -clear
```

Il nuovo utente una volta autenticato con kinit e aklog possiede tutte le credenziali necessarie per operare sulla sua directory.

3.3 Interazione con i client

Così come è stata finora configurata la cella permette l’interazione solo con client che abbiano kinit e aklog installati, escludendo così il meccanismo standard di klog. Naturalmente è necessario fornire tale meccanismo, altrimenti si perderebbe il vantaggio della accessibilità via rete del filesystem AFS.

Sono possibili due scenari: nel primo il server AFS usa il processo ka-forwarder per indirizzare al server Kerberos le richieste indirizzate al kaserver. Il server Kerberos usa un processo fakeka per fornire tali informazioni nel formato richiesto, prelevandole dal database Kerberos. Nel secondo scenario, che è quello che si realizzerà, sul server AFS è presente un server Kerberos slave, che risponde direttamente tramite fakeka alle richieste indirizzate al kaserver.

L’installazione dello slave Kerberos deve essere eseguita compilando i sorgenti con

l’opzione “--enable-fakeka”. Per modificare la configurazione è necessario sfruttare la directory di portdir_overlay, già introdotta per la compilazione di aklog e asetkey, e modificare lo script di ebuild introducendo l’opzione voluta come segue:

```
local myconf
myconf="--enable-fakeka"
```

Per l’installazione del KDC slave devono essere seguiti i passi indicati al paragrafo 1.2, e la configurazione del demone *krb524d* introdotto al paragrafo 3.2.1.

Infine deve essere avviato il servizio *fakeka*, su tutti i KDC in modo da ottenere maggiore affidabilità, con uno script simile in tutto a quello usato per *krb524d*. Anche in questo caso deve essere usata l’opzione di avvio “-m”.

Fatto ciò sarà possibile ottenere il token AFS mediante il comando:

```
#klog -principal <nome> -cell <nome cella>
```

anche non possedendo ticket Kerberos 5, e da un client appartenente ad una cella qualsiasi che abbia la cella ba.infn.it montata sotto root.afs.

Per quanto riguarda il client windows i test sono stati condotti installando OpenAFSforWindows versione 1.3.770 e KerberosforWindows versione 2.6.5. L’installazione dei due pacchetti è stata ultimata accettando le opzioni di default. E’ necessario modificare le impostazioni del client AFS per inserire la cella tra quelle conosciute. Dato che è Kerberos a richiedere il token AFS, si può anche abilitare alla partenza solo questo servizio, dopo aver configurato AFS in modo da mappare la directory personale su una unità di rete. Si noti che in Windows Kerberos usa il servizio kr524d per ottenere il token AFS alla maniera di aklog, cioè anche qui senza passare per un ticket Kerberos 4.

3.4 Volumi di backup

Un volume di backup è un volume clone che risiede sullo stesso sito della sua versione scrivibile. Creare volumi di backup dei volumi utente ha due scopi:

- Convenzionalmente il primo passo per immagazzinare un volume su un sistema di backup a nastro è quello di creare proprio un volume di backup. Infatti il dump su nastro di un volume lo rende inaccessibile, quindi è conveniente in tale operazione usare il volume di backup.
- Permette agli utenti di ripristinare dei file cambiati o cancellati per errore senza interpellare l’amministratore di sistema. Naturalmente è necessario montare il volume di backup in modo che sia accessibile agli utenti.

Prima di iniziare l’operazione di backup sul primo utente di prova bisogna verificare che l’utente usato sia un amministratore con

```
#bos listusers baafsserver
```

e che gli amministratori posseggano, sulla directory dove si vuole creare il mount point di accesso al backup, i diritti per inserire una nuova directory:

```
#fs listacl /afs/ba.infn.it/user/domenico
```

A questo punto è possibile creare il volume di backup e montarlo.

```
#vos backup user.domenico
    Created backup volume for user.domenico
#fs mkmount /afs/.ba.infn.it/user/domenico/backup user.domenico.backup
#vos release user
#fs checkv
```

Il volume di backup non può ovviamente essere modificato dall'utente. Per verificare che il punto di mount si riferisca al volume corretto si può usare il comando

```
#fs lsmount /afs/ba.infn.it/user/domenico/private/backup
```

La procedura di clonazione del volume utente può essere ripetuta automaticamente nelle ore notturne, così che ogni giorno l'utente abbia la possibilità di recuperare i file del giorno precedente.

```
#bos create baafsserver backupusers cron -cmd "/usr/afs/bin/vos backupsys -prefix user -localauth" "1:00"
```

Per verificare che il cron sia stato correttamente impostato:

```
#/usr/afs/bin/bos status baafsserver
    Instance ptserver, currently running normally.
    Instance vlserver, currently running normally.
    Instance buserver, currently running normally.
    Instance fs, currently running normally.
        Auxiliary status is: file server running.
    Instance backupusers, currently running normally.
        Auxiliary status is: run next at Wed Feb  2 01:00:00 2005.
```

Non viene trattata in questa nota la creazione di un backup su nastro.

4 UTENTI E PUBLIC LOGIN

La creazione di un nuovo utente della struttura fin qui illustrata prevede quindi la creazione innanzitutto della utenza AFS. Questa fornisce un ID, da utilizzarsi per la creazione della entità LDAP che ne gestirà le autorizzazioni. Infine viene creato il principal Kerberos, che conterrà la password di autenticazione dell'utente.

La struttura può essere usata per consentire agli utenti LDAP il login su di una macchina priva di utenti locali.

La macchina in questione deve essere resa innanzitutto client Kerberos e AFS, installando i relativi pacchetti, e deve essere disponibile il comando *aklog*, quindi deve essere installato anche il migration kit AFS-krb5. Fatto ciò un utente locale può richiedere un token AFS o tramite il meccanismo kinit+aklog, o semplicemente con klog. Per fare in modo che il meccanismo di autenticazione faccia riferimento al database Kerberos durante il login è necessario usare dei moduli pam appositi. Il modulo pam_krb5 fornito con la distribuzione Gentoo sebbene permetta l'autenticazione non permette la modifica della password con il comando unix standard, e nel caso in esame è stato sostituito dal modulo omonimo reperibile all'indirizzo [6]. Questo modulo permetterebbe anche l'acquisizione automatica del token AFS, ma tramite la generazione di un token k4, cosa non desiderabile se la cella dovrà essere cross-autenticata con altre celle.

Si noti che affinché sia possibile la modifica della password dalla macchina client deve essere autorizzato, tra tale macchina e il server Kerberos, il traffico sulla porta UDP 464. Una volta fatto ciò è possibile cambiare la password con il comando standard *passwd*.

L'acquisizione del token AFS tramite ticket Kerberos 5 è possibile utilizzando il modulo *libpam_openafs-session* distribuito con Debian [7], partendo naturalmente dal codice sorgente. Questo modulo riesce ad ottenere automaticamente il token AFS e a creare un nuovo PAG (Process Autentication Group), di fatto usando *aklog*. Il PAG è un numero che serve al Cache Manager AFS per identificare in modo univoco l'utente, ed evita che il superuser sul client possa guadagnare i privilegi AFS di un utente con un semplice *su*. Inoltre tutti i processi partiti dalla shell iniziale ereditano automaticamente il token senza perdere i privilegi acquisiti nello spazio AFS, anche nel caso in cui l'userid venga cambiato. Al contrario di quanto accade per il modulo *pam_aklog* distribuito da RedHat il modulo qui usato provvede anche ad effettuare un unlog alla chiusura della sessione.

Affinché l'autenticazione faccia quanto descritto deve essere modificato il file */etc/pam.d/system-auth*, che è riportato in appendice.

Manca ancora la possibilità di accedere alla macchina con un utente non locale ma registrato su LDAP. E' necessario innanzitutto configurare la parte client LDAP, come già descritto al paragrafo 2.2.1, importando il certificato della CA utilizzata e modificando il file */etc/openldap/ldap.conf*. Bisogna fare attenzione a che i nomi dei server ldap siano i FQDN e non degli alias, in quanto devono coincidere con i nomi presenti nei certificati. Se il client ldap funziona deve essere possibile dalla macchina in questione eseguire *ldapsearch*, una volta

richieste ed ottenute le credenziali Kerberos. Per permettere il login degli utenti della directory bisogna installare il pacchetto *nss_ldap*, aggiornare opportunamente il file */etc/ldap.conf* (può bastare copiare il file */etc/openldap/ldap.conf*) e modificare il file */etc/nsswitch.conf* come segue:

```
passwd:     compat ldap
shadow:     compat
group:      compat ldap
```

Si modifica infine l'utente LDAP in modo che abbia uidNumber e gidNumber corrispondenti a quelli AFS, e si modifica l'attributo homeDirectory in modo che contenga la propria directory AFS /afs/ba.infn.it/user/[nomeutente]. A questo punto deve essere possibile accedere alla macchina con l'utente LDAP e password Kerberos 5, e trovarsi automaticamente sulla propria directory AFS, possedendo un ticket K5 ed un token AFS.

4.1 OpenSSH

Con la versione attuale di openssh (3.9p1) la autenticazione tramite pam avviene in un processo separato, non correlato alla shell finale dell'utente, con la quale dunque non condivide i dati relativi alle credenziali ottenute. Nelle versioni precedenti invece il processo di autenticazione finiva per essere un *ancestor* della shell. In sostanza quel che succede è che è possibile autenticarsi con la password Kerberos, ma un successivo *klist* mostra la cache delle credenziali vuota, ed ovviamente non si riesce ad ottenere il token AFS. Con le versioni attuali dei software usati l'unica soluzione consiste nell'abilitare l'uso dei POSIX threads all'interno di openSSH: i dati delle credenziali sono così conservati nello stesso spazio di indirizzamento del processo sshd principale, e sopravvivono alla chiusura del thread di autenticazione, diventando così disponibili al processo che origina la shell.

Per abilitare l'uso dei threads in openssh l'applicazione va ricompilata con le opzioni:

```
--with-cflags=-DUSE_POSIX_THREADS --with-ldflags=-lpthread
```

In Gentoo si tratta di usare la directory PORTDIR_OVERLAY, come descritto in precedenza, e modificare la variabile locale my_conf nella funzione src_compile() dell'ebuild.

Dopo la ricompilazione, al login tramite ssh la cache delle credenziali Kerberos contiene il corretto *default principal*, e il ticket per *krbtgt*, ma non contiene il ticket AFS come avviene per il login da console. Il token AFS viene però correttamente ottenuto dal modulo PAM. Si è verificato che il meccanismo di richiesta del token usa aklog e non klog, per questo dato che il ticket AFS serve solo per ottenere il token, si ritiene questa soluzione un buon compromesso ai fini pratici.

5 RIFERIMENTI E BIBLIOGRAFIA

- [1]-Distribuzione Gentoo http://www.gentoo.org
- [2]-MIT Kerberos 5 http://web.mit.edu/kerberos/
- [3]-OpenLDAP http://www.openldap.org/
- [4]-SASL http://asg.web.cmu.edu/sasl/
- [5]-OpenAFS http://www.openafs.org/
- [6]-Pam_krb5 http://sourceforge.net/projects/pam-krb5/.
- [7]-Pam_afs http://packages.debian.org/stable/net/libpam-openafs-session
- LDAPv3 How-to http://www.bayour.com/LDAPv3-HOWTO.html
- Migrazione kaserver kdc http://www.ts.infn.it/events/ccr2002/talks/ricciardi1.ppt
- Centralized User Management with Kerberos and LDAP

http://www.samag.com/documents/s=9494/sam0502a/0502a.htm

- Replacing NIS with Kerberos and LDAP How-To

http://www.ofb.net/~jheiss/krbldap/howto.html

- Andrei Maslennikov - AFS, fileystem distribuito e multipiattaforma - Linux&C 37-41
- Gentoo Forum- Replacing NIS with LDAP/Kerberos

http://forums.gentoo.org/viewtopic.php?t=55518

- KerberosAFSInstall

http://www.central.org/twiki/bin/view/AFSLore/KerberosAFSInstall

- How to set up AFS with Kerberos V

http://www.mathematik.uni-karlsruhe.de/~iwrmm/Persons/Schulz/Unix/afs/afs-krb5.html

APPENDICE

a. /etc/krb5.conf

```
[libdefaults]
        default_realm = BA.INFN.IT
        dns_lookup_realm = false
        dns_lookup_kdc = false
        default_tkt_enctypes = des3-hmac-sha1 des-cbc-crc rc4-hmac
        default_tgs_enctypes = des3-hmac-sha1 des-cbc-crc rc4-hmac
        permitted_enctypes = des3-hmac-sha1 des-cbc-crc rc4-hmac
        renew_lifetime = 7d

[realms]
        BA.INFN.IT = {
        kdc = master.ba.infn.it
        kdc = slave.ba.infn.it
        kdc = afsserver.ba.infn.it
        admin_server = master.ba.infn.it
        }

[domain_realm]
        .ba.infn.it = BA.INFN.IT
        ba.infn.it = BA.INFN.IT

[kdc]
        profile = /etc/krb5kdc/kdc.conf

[logging]
        kdc = FILE:/var/log/krb5kdc.log
        admin_server = FILE:/var/log/kadmin.log
        default = FILE:/var/log/krb5lib.log
```

b. /etc/krb5kdc/kdc.conf

```
[kdcdefaults]
        kdc_ports = 88,750

[realms]
        BA.INFN.IT = {
        database_name = /etc/krb5kdc/principal
        admin_keytab = /etc/krb5kdc/kadm5.keytab
        acl_file = /etc/krb5kdc/kadm5.acl
        dict_file = /usr/share/dict/words
```

```
    key_stash_file = /etc/krb5kdc/.k5.BA.INFN.IT
    kadmind_port = 749
    max_life = 24h 0m 0s
    max_renewable_life = 7d 0h 0m 0s
    master_key_type = des-cbc-crc
    kdc_supported_enctypes = des3-hmac-sha1:normal des-cbc-crc:normal des-cbc-crc:v4 des-cbc-crc:afs3 rc4-hmac:normal
    supported_enctypes = des3-hmac-sha1:normal des-cbc-crc:normal des-cbc-crc:v4 des-cbc-crc:afs3 rc4-hmac:normal
    default_principal_flags = +forwardable
    }
```

c. /etc/xinetd.conf

```
defaults
{
        instances      = 60
        log_type       = SYSLOG authpriv info
        log_on_success = HOST PID
        log_on_failure = HOST
        cps            = 25 30
}

service krb_prop
{
        flags      = REUSE NAMEINARGS
        socket_type = stream
        protocol    = tcp
        wait        = no
        user        = root
        server      = /usr/sbin/kpropd
        server_args = kpropd
}

service eklogin
{
        flags      = REUSE NAMEINARGS
        socket_type = stream
        protocol    = tcp
        wait        = no
        user        = root
        server      = /usr/sbin/klogind
        server_args = klogind -k -c -e
}

includedir /etc/xinetd.d
```

d. /etc/openldap/slapd.conf

```
include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/inetorgperson.schema
include         /etc/openldap/schema/nis.schema
include         /etc/openldap/schema/krb5-kdc.schema

include                         /etc/openldap/slapd.access

replogfile      /var/openldap-data/master-slapd.replog

#disallow bind_anon bind_simple

schemacheck     on

loglevel        256

threads         8

idletimeout     14400

# Define global ACLs to disable default read access.

# Do not enable referrals until AFTER you have a working directory
# service AND an understanding of referrals.
#referral       ldap://root.openldap.org

pidfile         /var/openldap-data/slapd.pid
argsfile        /var/openldap-data/slapd.args

security ssf=112

sasl-host       kerberos.ba.infn.it
sasl-realm      BA.INFN.IT
sasl-secprops   noplain,noactive,noanonymous

sasl-regexp     uid=([^/]*),cn=ba.infn.it,cn=gssapi,cn=auth
                ldaps:///dc=ba,dc=infn,dc=it??sub?(&(uid=$1)(objectClass=person))
```

```
TLSCipherSuite HIGH
TLSCertificateFile /etc/openldap/ssl/slapd.pem
TLSCertificateKeyFile /etc/openldap/ssl/key.pem

#######################################################################
# ldbm database definitions
#######################################################################

database        bdb
suffix          "dc=ba,dc=infn,dc=it"
rootdn          "cn=ldapadm,dc=ba,dc=infn,dc=it"

rootpw          segreta

directory       /var/lib/openldap-data

# Indices to maintain
index   sn,uid,krb5PrincipalName                pres,eq,approx
index   objectClass,uidNumber,gidNumber,memberUid       eq
index   cn,mail,givenname                       eq,subinitial

replica         uri=ldaps://slave.ba.infn.it:636
                authcId=host/master.ba.infn.it@BA.INFN.IT
                bindmethod=sasl  saslmech=GSSAPI
```

e. base.ldif

```
dn: dc=ba,dc=infn,dc=it
objectclass: organization
objectclass: dcObject
objectClass: top
o: INFN
dc: ba
description: INFN sezione di Bari

dn: ou=gruppo1,dc=ba,dc=infn,dc=it
ou: gruppo1
objectClass: top
objectClass: organizationalUnit
objectClass: domainRelatedObject
associatedDomain: ba.infn.it
```

dn: ou=hosts,ou=gruppo1,dc=ba,dc=infn,dc=it

ou: hosts

objectClass: top

objectClass: organizationalUnit

objectClass: domainRelatedObject

description: Host gruppo 1

associatedDomain: ba.infn.it

dn: ou=people,ou=gruppo1,dc=ba,dc=infn,dc=it

ou: people

objectClass: top

objectClass: organizationalUnit

objectClass: domainRelatedObject

description: Persone gruppo 1

associatedDomain: ba.infn.it

dn: ou=admins,ou=gruppo1,dc=ba,dc=infn,dc=it

ou: admins

objectClass: top

objectClass: organizationalUnit

objectClass: domainRelatedObject

description: Amministratori gruppo 1

associatedDomain: ba.infn.it

f. users.ldif

dn: uid=utente1,ou=people,ou=gruppo1,dc=ba,dc=infn,dc=it

uid: utente1

cn: utente1

sn: utente1

mail: domenico.diacono@ba.infn.it

objectClass: person

objectClass: organizationalPerson

objectClass: inetOrgPerson

objectClass: top

objectClass: posixAccount

objectClass: krb5Principal

krb5PrincipalName: utente1@BA.INFN.IT

loginShell: /bin/bash

uidNumber: 91001

gidNumber: 9100

homeDirectory: /home/utente1

dn: uid=utente2,ou=people,ou=gruppo2,dc=ba,dc=infn,dc=it

uid: utente2

cn: utente2

sn: utente2

mail: domenico.diacono@ba.infn.it

objectClass: person

objectClass: organizationalPerson

objectClass: inetOrgPerson

objectClass: top

objectClass: posixAccount

objectClass: krb5Principal

krb5PrincipalName: utente2@BA.INFN.IT

loginShell: /bin/bash

uidNumber: 91002

gidNumber: 9101

homeDirectory: /home/utente2

dn: uid=admin1,ou=admins,ou=gruppo1,dc=ba,dc=infn,dc=it

uid: admin1

cn: admin1

sn: admin1

mail: domenico.diacono@ba.infn.it

objectClass: person

objectClass: organizationalPerson

objectClass: inetOrgPerson

objectClass: top

objectClass: posixAccount

objectClass: krb5Principal

krb5PrincipalName: admin1@BA.INFN.IT

loginShell: /bin/bash

uidNumber: 91003

gidNumber: 9100

homeDirectory: /home/admin1

dn: uid=domenico,ou=people,ou=admins,dc=ba,dc=infn,dc=it

uid: domenico

cn: domenico

sn: domenico

```
mail: domenico.diacono@ba.infn.it
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: top
objectClass: krb5Principal
objectClass: posixAccount
krb5PrincipalName: domenico@BA.INFN.IT
loginShell: /bin/bash
uidNumber: 1000
gidNumber: 100
homeDirectory: /home/domenico
```

g. /etc/openldap/slapd.access (master)

```
access to dn.subtree="ou=gruppo1,dc=ba,dc=infn,dc=it"
        by dn.subtree="ou=admins,ou=gruppo1,dc=ba,dc=infn,dc=it" write
        by * read break
access to dn.subtree="ou=gruppo2,dc=ba,dc=infn,dc=it"
        by dn.subtree="ou=admins,ou=gruppo2,dc=ba,dc=infn,dc=it" write
        by * read break
access to dn.subtree="ou=gruppo3,dc=ba,dc=infn,dc=it"
        by dn.subtree="ou=admins,ou=gruppo3,dc=ba,dc=infn,dc=it" write
        by * read break
access to dn.subtree="ou=gruppo4,dc=ba,dc=infn,dc=it"
        by dn.subtree="ou=admins,ou=gruppo4,dc=ba,dc=infn,dc=it" write
        by * read break
access to dn.subtree="ou=gruppo5,dc=ba,dc=infn,dc=it"
        by dn.subtree="ou=admins,ou=gruppo5,dc=ba,dc=infn,dc=it" write
        by * read break
access to dn.subtree="ou=amministrazione,dc=ba,dc=infn,dc=it"
        by dn.subtree="ou=admins,ou=amministrazione,dc=ba,dc=infn,dc=it" write
        by * read break
access to dn.subtree="ou=servizi,dc=ba,dc=infn,dc=it"
        by dn.subtree="ou=admins,ou=servizi,dc=ba,dc=infn,dc=it" write
        by * read break
access to attr=loginShell,mail
      by self write
      by * none
access to attr=krb5PrincipalName
      by self read
      by * none
```

```
access to *
      by dn.subtree="ou=people,ou=admins,dc=ba,dc=infn,dc=it" write
      by * read
```

h. /etc/openldap/slapd.access (slave)

```
access to attr=loginShell,mail, krb5PrincipalName
      by self read
      by * none
access to *
      by dn="uid=host/bakdcmaster.ba.infn.it,cn=ba.infn.it,cn=gssapi,cn=auth" write
      by * read
```

i. /etc/init.d/mit-krb524d

```
#!/sbin/runscript
#-------------------------------------------------------------------------
# This script starts/stops the MIT Kerberos 5 krb524d
#-------------------------------------------------------------------------
daemon="MIT Kerberos 5 - AFS"
exec="/usr/sbin/krb524d"
opts="start stop restart"
depend() {
      need net mit-krb5kdc
}
start() {
      ebegin "Starting $daemon"
      start-stop-daemon --start --quiet --exec ${exec} -- -m >/dev/null 1>&2
      eend $? "Error starting $daemon"
}
stop() {
      ebegin "Stopping $daemon"
      start-stop-daemon --stop --quiet --oknodo --exec ${exec} 1>&2
      eend $? "Error stopping $daemon"
}
restart() {
  svc_stop
  svc_start
}
```

j. /usr/portage.local/app-crypt/afs-krb5/afs-krb5-2.0.ebuild

```
# Copyright 1999-2003 Gentoo Technologies, Inc.
# Distributed under the terms of the GNU General Public License v2
# $Header: /home/cvsroot/gentoo-x86/app-crypt/pam_krb5/pam_krb5-1.0.ebuild,v 1.15 2003/10/01 09:39:25 aliz Exp $

DESCRIPTION="NRL AFS Krb5 Migration Kit"
SRC_URI="ftp://ftp.cmf.nrl.navy.mil/pub/kerberos5/${P}.tar.gz"
HOMEPAGE="http://www.mathematik.uni-karlsruhe.de/~iwrmm/Persons/Schulz/Unix/afs/afs-krb5.html"

inherit eutils

SLOT="0"
LICENSE="as-is"
KEYWORDS="~x86"
IUSE="afs kerberos krb4"

DEPEND="app-crypt/mit-krb5
        net-fs/openafs"

S=${WORKDIR}/${PN}/src

src_compile() {
#       econf CPPFLAGS="-I/usr/afsws/include" CFLAGS="$CFLAGS" || die
        econf CFLAGS="$CFLAGS" || die

        make aklog || die
        make asetkey || die
}

src_install() {
        exeinto /usr/bin
        doexe aklog

        exeinto /usr/sbin
        doexe asetkey

        cd ..
        dodoc AFS_K5_NAME_CHANGE COPYRIGHT ISSUES README THEORY
}
```

k. /usr/vice/etc/CellServDB

```
>caspur.it #CASPUR Inter-University Computing Consortium, Rome
193.204.5.45            #pomodoro.caspur.it
193.204.5.50            #maslo.caspur.it
193.204.5.46            #banana.caspur.it
>infn.it #Istituto Nazionale di Fisica Nucleare (INFN), Italia
141.108.3.252           #afsrm1.roma1.infn.it
192.84.134.75           #afsna.na.infn.it
131.154.1.7             #afscnaf.infn.it
>cern.ch #European Laboratory for Particle Physics, Geneva
137.138.246.51          #afsdb2.cern.ch
137.138.118.228         #afsdb5.cern.ch
137.138.118.227         #afsdb4.cern.ch
137.138.246.50          #afsdb3.cern.ch
137.138.128.148         #afsdb1.cern.ch
```

l. /etc/pam.d/system-auth

#%PAM-1.0

auth required /lib/security/pam_env.so

auth sufficient /lib/security/pam_unix.so likeauth nullok nodelay

auth sufficient /lib/security/pam_krb5.so debug use_first_pass forwardable

auth required /lib/security/pam_deny.so

account required /lib/security/pam_unix.so

password required /lib/security/pam_cracklib.so retry=3

password sufficient /lib/security/pam_unix.so nullok md5 shadow use_authtok

password sufficient /lib/security/pam_krb5.so use_authtok debug

password required /lib/security/pam_deny.so

session required /lib/security/pam_limits.so

session required /lib/security/pam_unix.so

session optional /lib/security/pam_krb5.so debug

session optional /lib/security/pam_openafs-krb5.so debug