Disclaimer

Ringraziamenti

A Iliana per la sua pazienza.
A tutti coloro che contribuiscono alla diffusione e implementazione del software OpenSource.

Introduzione

Una delle caratterizzazioni della società odierna, è la velocità intesa come flussi di lavoro e scambio di dati, questo diviene oggi un valore portante, sia nella professione che nelle comunicazioni interpersonali private.
I posti di lavoro inoltre, perdono sempre più la connotazione di luoghi fisici dislocati in uno spazio preciso, per rivestirsi delle caratteristiche di mobilità, di nomadismo per le ragioni esposte in precedenza e sempre per queste, abbisognano di servizi dove lo scambio e il recupero di dati sia sempre più semplice, veloce, accessibile, sicuro.
Ecco perché la domanda dei servizi da parte degli utenti è caratterizzata sempre dalla richiesta di flessibilità, mobilità, tempi ristretti.
Mezzi di comunicazione funzionali e funzionanti con possibilità di accesso alle reti 24 ore al giorno, con tempi di uptime dei servizi prossimi al 99.9 %

La posta elettronica è diventata il mezzo principe per permettere la comunicazione, sia in tempo reale che differito, e lo scambio di dati in formato elettronico.
Potremmo pensare di integrarla poi con note e appuntamenti, e avremmo uno strumento completo per gestire l'agenda di tutti i giorni.
L'utente ritiene normale che sia possibile accedere ad Internet in qualsiasi luogo e in qualsiasi tempo, sia per lavoro e sempre più anche per l'offerta di intrattenimento.
Da un punto di vista strettamente tecnico la maggior richiesta ha fatto si che quella che definiamo Internet stia assorbendo tutte le altre reti di comunicazione nate con caratteristiche diverse (fonia, trasmissione dati, etc.)

Riassumendo l'utente finale pretende tempi di accesso ridotti, facilità di funzionamento, sicurezza dei dati, disponibilità del servizio e connettività ubiqua.
Spesso in queste analisi non si riesce a cogliere la complessità della struttura che garantisce il funzionamento di tutto, ovvero "l'altra faccia della medaglia".
Questi servizi vengono forniti tramite delle infrastrutture che colmano il gap tra le macchine e l'utenza.
Esistono delle figure professionali che si occupano proprio di creare quel ponte che permette l'accesso e l'uso delle tecnologie alle persone che richiedono queste tecnologie.

Questo scritto è rivolto a queste figure professionali, proponendo una soluzione veloce snella e sicura per la configurazione di una killer-application come quella della posta elettronica.
Di seguito si motiveranno le scelte fatte, la prima principale e sostanziale nel previlegiare l'uso di software Open Source che svincola l'operatore da licenze e costi aggiuntivi in caso di espansione del numero di utenti (ogni utente o gruppo di utenti necessitano di una licenza se il software non è Open Source).
La seconda altrettanto importante consiste nel fatto che il software OpenSource può scalare in modo agile da macchine con singolo processore e quantità ridotte di memoria a macchine con multiprocessore e grosse quantità di memoria, praticamente la soluzione proposta risulta agevole da usare sia nelle PMI che in ambito corporate.

Considerazioni e impostazione del sistema

I dati che noi dobbiamo trattare sono: UserID, password e indirizzo e-mail.
Lo UserID può essere di un utente con accesso shell al sistema oppure per un utente virtuale, cioè che non ha un account di sistema.
Il sistema dovrà essere facilmente integrabile con quanto già esistente.
L'accesso alla posta dovrà essere reso disponibile tramite i protocolli standard POP e IMAP
Deve esserci la possibilità di autenticazione per accettare il relay.
Sarà possibile l'utilizzo di domini multipli con un unico indirizzo IP.
Si dovrà lavorare con un repository di dati veloce, replicabile e riutilizzabile per l'autenticazione di altri servizi (SingleSignOn).
Il sistema operativo dovrà essere stabile ed avere caratteristiche di affidabilità e sicurezza.

Partendo dalla base il sistema operativo sarà GNU/Linux e nel caso specifico una RedHat 7.3 con tutte le patch di sicurezza.
L'utilizzo di GNU/Linux fa si che quanto esposto di seguito sia portabile su molte piattaforme di architettura diversa da quella x86.

La struttura da realizzare è indicata dallo schema A allegato.

Scelta del repository dei dati

Il modo più semplice per garantire una struttura dati che sia riutilizzabile anche in ambiti SigleSignOn e che permetta l'autenticazione dei servizi è LDAP.
Gli utenti poi saranno tutti slegati dall'accesso al server e si definiranno virtuali.
In caso di Linux il software che ci permette questo è OpenLDAP.

Scelta del MTA di lavoro

I software più conosciuti che offre Linux sono tre: Sendmail, Qmail e Postfix.

Sendmail è lo storico MTA di Linux, ma non supporta il formato delle mail in Maildir e lavora con previlegi di root.
Qmail è molto più sicuro, ma viene rilasciato in con una licenza particolare e non GPL, supporta di default il formato Maildir e viene rilasciato solamente in formato tar.gz
Postfix supporta di default il formato mailbox,modificando la configurazione anche il formato Maildir, si interfaccia con diversi metodi di autenticazione e viene rilasciato in GPL allegato con varie distribuzioni.

La differenza sostanziale tra il formato mailbox e il formato maildir è che il primo ha tutte le mail in un unico file e in caso che questo si corrompa si rischia la perdita di tutti i messaggi, il secondo crea un file per ogni messaggio.
Questo permette una maggiore sicurezza in caso di errore, in caso di scrittura errata si perde al massimo un messaggio.

Il formato Maildir è intrinsecamente più sicuro e su questo è ricaduta la mia scelta, non posso usare

quindi sendmail.
Inoltre volendo mantenere l'installazione esclusivamente in pacchettizzazione per una migliore manutenzione della stessa, la scelta finale ricade su postfix.

Scelta del sistema pop/imap

Rendere disponibili le mail tramite protocolli conosciuti come POP3 e IMAP con i corrispondenti su SSL.
Il software scelto deve interfacciarsi con lo storage dei dati che è in formato Maildir e quindi la scelta ricade su un software come Courier-imap, che di default usa il formato richiesto ed ha dato prova di buona affidabilità e stabilità.

Scelta dell'interfaccia di amministrazione

L'idea di base è rendere la gestione rapida e senza bisogno di client particolari, per poter amministrare gli account e i domini in qualsiasi situazione.
Ci deve essere la possibilità di gestire domini multipli per via della scalabilità della struttura e si deve poter delegare la gestione di ogni singolo dominio al postmaster o ad un account di fiducia.
Deve esserci una gestione delle quote per evitare che gli utenti usino grosse parti di disco in modo indiscriminato e queste devono risiedere nello stesso posto dove scriviamo gli account.
Lo strumento che riunisce tutte queste caratteristiche si chiama Jamm.
E' l'acronimo di JAva Mail Manager ed è composto da classi Java che si interfacciano tramite il protocollo LDAP al servizio di OpenLDAP.
Permette la gestione degli account, dei domini, degli alias.
Si può definire con un semplice flag se il dominio, l'account o l'alias sono attivi o meno, questo è particolarmente utile in caso di vendita del servizio (ISP).
Permette una visione immediata del numero di account e alias presenti per dominio.
Per poterlo utilizzare si deve comunque installare un esecutore di classi Java.
Anche in questo caso la scelta cade su un software rilasciato con licenza GPL, affidabile e sviluppato dal gruppo di lavoro di apache, tomcat nella versione 4.

Scelta del servizio web

Per dare la possibilità di accesso tramite web dobbiamo installare apache compreso il mod_ssl per poi sfruttare il protocollo https quando trasferiamo utente, password e posta.
In generale serve inoltre una Java Virtual Machine per poter eseguire le classi richieste

Praticamente la struttura finale è quella indicata dallo schemaB.

Installazione dei pacchetti

Cominciamo con installare la distribuzione, nel nostro caso una RedHat 7.3 (http://www.redhat.com) e aggiornarla con le ultime patch di sicurezza.
In caso abbiate connettività consiglio l'utilizzo di uno strumento interessante derivato dal pacchetto apt di Debian per la RedHat (http://www.freshrpms.net)
Installiamo il software per ottenere i servizi minimi, nel nostro caso solo sshd per permetterci una gestione da remoto.
Installiamo i pacchetti openldap-servers, apache, mod_ssl, aspell-it, php, con le relative dipendenze.
Il postfix che dobbiamo installare deve avere il supporto LDAP oltre ad una patch che ci permette di utilizzare il campo quota presente sullo stesso.
Questa patch si chiama VDA (http://web.onda.com.br/nadal/) ed è stata integrata nelle ultime versioni di postfix (http://www.postfix.org)
Purtroppo la versione di default che viene montata sulla RedHat 7.3 è la 1.1.12-0.7 e non supporta le caratteristiche richieste.
Dobbiamo scaricare una versione aggiornata, per comodità di gestione conviene lavorare con un pacchetto RPM anche per il postfix e quindi possiamo scaricare il sorgente (http://postfix.wl0.org/ftp/SRPMS/) e ricompilarlo per ottenere un rpm con le caratteristiche da noi volute.
LDAP per questo pacchetto è supportato di default, ma dobbiamo ricordarci di includere la patch VDA e quant'altro ci interessi.
Installiamo il pacchetto procmail che ci permette di usare un filtro sulle mail locali.
Il problema si pone con il courier-imap che viene rilasciato in formato tar.gz (http://www.courier-mta.org/download.php#imap) e quindi non in linea con l'iniziale intenzione di lavorare solamente con la pacchettizzazione rpm.
Ricompiliamo quindi il tar.gz in modo da ottenere i pacchetti rpm richiesti.
La ricompilazione è molto rapida poichè il file .spec è contenuto all'interno del file tgz e ci permette di

ottenere i pacchetti con un unico comando.
I due pacchetti che ci interessano sono courier-imap e courier-imap-ldap, rispettivamente motore e autenticazione basata su LDAP.
Dagli stessi sorgenti è possibile ricavare anche i pacchetti per l'autenticazione su mysql e postgres.
Installiamo la JVM (http://www.java.com/en/download/linux_manual.jsp) e tomcat nella versione che preferite (http://jakarta.apache.org/tomcat/index.html).
Per quanto riguarda tomcat possiamo ricompilare il sorgente per ottenere il pacchetto rpm.
Scarichiamo Jamm (http://jamm.sourceforge.net/).
Scarichiamo e installiamo Squirrelmail (http://www.squirrelmail.org/download.php).
Eventualmente scarichiamo anche i plug-in che ci interessano.

Configurazione

Convenzioni:
Le scritte in corsivo sono comandi da dare: *ls -l*
Le righe che cominciano per # indicano comandi da fare come root
Le righe che cominciano per $ indicano comandi da dare come utente

Per la configurazione prenderemo in esame un dominio fittizio che sarà example.com
Tutto il traffico email verrà gestito con un utente reale del sistema che chiameremo vmail.
Creiamo l'utente vmail e definiamo la directory /home/vmail/domains dove andranno a finire i messaggi.

 # useradd -m -k -r vmail
 # su – vmail
$ *mkdir domains*

dobbiamo appuntare lo UserID dell'utente vmail in quanto verrà richiesto più tardi.

OpenLDAP

Dobbiamo creare la struttura della directory e configurare /etc/openldap/slapd.conf
Aggiungiamo tra gli schemi da usare jamm.schema che è presente nel pacchetto di Jamm.
Le righe di /etc/openldap/slapd.conf appariranno così:

```
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/jamm.schema
```

Aggiungiamo le regole che devono essere rispettate nell'accesso ai dati presenti su LDAP

```
access to dn=".*,jvd=([^,]+),o=hosting,dc=example,dc=com" attr=userPassword
        by self write
        by group/jammPostmaster/roleOccupant="cn=postmaster,jvd=$1,o=hosting,\
        dc=example,dc=com" write
        by anonymous auth
        by * none

access to dn=".*jvd=([^,]+),o=hosting,dc=example,dc=com"
        by self write
        by group/jammPostmaster/roleOccupant="cn=postmaster,jvd=$1,o=hosting,\
        dc=example,dc=com" write
        by * read

access to *
        by * read
```

Quanto indicato sopra permette di avere un account (postmaster) che può accedere in lettura e scrittura ai dati relativi al suo dominio di posta.
Permette ad ogni singolo account il cambio della propria password ed qualsiasi account che rientra nel gruppo di jammPostmaster può essere delegato ad amministrare gli account del proprio dominio.
Il rootdn anche se non indicato, di default ha sempre il permesso di scrittura.

Definiamo ora la struttura del database di LDAP e il rootdn

```
database        ldbm
suffix          "dc=example,dc=com"
rootdn          "cn=manager,dc=example,dc=com"
```

Si consiglia di scrivere la password del rootdn in modalità criptata, ricavandola dal comando

 # slappasswd
New password: rhx
Re-enter new password: rhx
{SSHA}xdCLuqmdQ5aFXdzwMkDJX0klwcHqgl7R

La stringa risultante dovrà essere copiata esattamente così dentro al file /etc/openldap/slapd.conf

```
rootpw  {SSHA}xdCLuqmdQ5aFXdzwMkDJX0klwcHqgl7R
```

Definiamo gli oggetti che devono essere indicizzati dentro il db.

```
index           objectClass     eq
index           cn,mail         eq,subinitial
```

In questa sede non viene analizzata la replicazione dell'LDAP.
Possiamo salvare il file e riavviare il servizio ldap.

Ora deve essere popolato il database in modo di creare la struttura iniziale.
Il pacchetto di jamm propone uno schema di base che può essere utilizzato allo scopo, il file sample.ldif in realtà contiene molto di più che la semplice struttura di base ed è possibile evitare di usarlo creando un file **sample.ldif** contenente le righe indicate sotto.
Installiamo ldap-client che ci permette di lavorare in modo molto rapido con LDAP tramite alcuni script.
Il contenuto del file per la creazione della base deve essere il seguente:

```
-------------------------------------
dn: dc=example, dc=com
objectClass: top
objectClass: domain
dc: example

dn: o=hosting, dc=example, dc=com
objectClass: top
objectClass: organization
o: hosting
description: mail.example.com hosting root

dn: cn=manager,dc=example,dc=com
objectClass: top
objectClass: organizationalRole
cn: utente di amministrazione
-------------------------------------
```

Verifichiamo che il servizio di LDAP sia attivato con

 # service ldap status
slapd (pid 1290 1289 1243 1242 597 596 587) in esecuzione...

e inseriamo i dati del file sample.ldif in questa maniera:

$ *ldapadd -x -v -D "cn=manager,dc=example,dc=com" -W -f sample.ldif*

Chiede la password che avevamo dichiarato dentro /etc/openldap/slapd.conf e quindi non resta nella history della shell.
A questo punto LDAP è pronto per essere consultato.

Postfix

Il postfix deve essere ricompilato almeno con il supporto per ldap, pcre, tls e vda.
Il principale file di configurazione è /etc/postfix/main.cf.
In questa sede affronteremo le problematiche di integrazione in questa struttura e non quello che deve

essere fatto per il controllo antispam, relay e quant'altro.
Editiamo il main.cf.

Per prima cosa abilitiamo l'uso di procmail

```
mailbox_command = /usr/bin/procmail
```

Definiamo dove postfix deve recuperare i dati
possiamo trovare queste informazioni all'interno dei file di esempio che vengono installati con il pacchetto.
Il file in questione è: /etc/postfix/samples/sample-ldap.cf
e risulta anche molto ben commentato, i parametri sono comunque i seguenti.

```
ldap_timeout = 10
ldap_server_host = 127.0.0.1
ldap_server_port = 389
ldap_bind_dn = cn=manager,dc=example,dc=com
ldap_domain = o=hosting,dc=example,dc=com
```

Dobbiamo definire che "transport" deve essere usato per le mail di ogni singolo dominio, si consiglia la lettura del documento http://www.postfix.org/big-picture.html per comprendere la struttura dei processi di postfix.
Praticamente diciamo al qmgr (queue manager) che strada deve fare per indirizzare la posta.

```
transport_server_host = localhost
transport_search_base = o=hosting,dc=example,dc=com
transport_query_filter = (&(&(jvd=%s)(objectClass=JammVirtualDomain))(accountActive=TRUE)\
        (delete=FALSE))
transport_result_attribute = postfixTransport
transport_cache = no
transport_bind = no
transport_scope = one
transport_maps = ldap:transport
```

L' objectClass è JammVirtualDomain che nel dn jvd contiene il nome a dominio per il quale offriamo il servizio.
Come potete notare verifichiamo che il dominio sia attivo e non sia in fase di cancellazione (questi flag verranno analizzati meglio nella sezione riservata a Jamm)
Ricaviamo da questa interrogazione il valore del transport da usare.

Dobbiamo dichiarare a postfix che deve usare anche il valore ricavato dalla query a ldap, aggiungendo in coda a mydestination la transport_maps

```
mydestination = $myhostname, localhost.$mydomain, $mydomain, $transport_maps
```

Definiamo gli alias di posta.

```
aliases_server_host = localhost
aliases_search_base = o=hosting,dc=example,dc=com
aliases_query_filter = (&(&(objectClass=JammMailAlias)(mail=%s))(accountActive=TRUE))
aliases_result_attribute = maildrop
aliases_bind = no
aliases_cache = no
```

L'objectClass interrogato è JammMailAlias.
Il record è presente solamente su LDAP, non possiede dati scritti sul filesystem.
In caso di cancellazione viene eliminato immediatamente e quindi l'unica cosa che dobbiamo controllare è se è attivo o meno.
Il campo maildrop conterrà le mail reali verso le quali dirottare la mail indirizzata all'alias.

Definiamo gli account reali, cioè coloro che alla fine hanno accesso alla scrittura su filesystem e che quindi archiviano il contenuto delle mail, fino alla richiesta da parte dell'utente autorizzato.

```
accounts_server_host = localhost
accounts_search_base = o=hosting,dc=example,dc=com
accounts_query_filter = (&(&(objectClass=JammMailAccount)(mail=%s))(accountActive=TRUE)\
```

```
        (delete=FALSE))
accounts_result_attribute = mailbox
accounts_cache = no
accounts_bind = no


accountsmap_server_host = localhost
accountsmap_search_base = o=hosting,dc=example,dc=com
accountsmap_query_filter = (&(&(objectClass=JammMailAccount)(mail=%s))(accountActive=TRUE)\
        (delete=FALSE))
accountsmap_result_attribute = mail
accountsmap_cache = no
accountsmap_bind = no
```

L'objectClass è JammMailAccount e facciamo due richieste distinte.
Se la mailbox è valida e prima di inviarla all'account di catchall (eventualmente attivato) verifica la presenza dell'account reale.
Anche in questo caso verifichiamo che l'account sia attivo e non in cancellazione.

Definiamo ora l'utente di sistema che deve archiviare le mail.
Noi ne avevamo definito uno (vmail) e ora consideriamo che l'utente abbia uno UserID=501 e un GID=501.
Definiamo inoltre la struttura dove devono essere scritti i messaggi, richiedendo gli account da LDAP.

```
virtual_mailbox_base = /home/vmail/domains
virtual_mailbox_maps = ldap:accounts
virtual_minimum_uid = 501
virtual_uid_maps = static:501
virtual_gid_maps = static:501
```

Definiamo quindi come il processo di postfix deve verificare gli account virtuali e in che ordine.
Qui di seguito indichiamo la riga da inserire, riservandosi di utilizzare anche il file di testo /etc/postfix/virtual con precedenza sugli account residenti su LDAP.
Indichiamo anche come mappare gli account che devono essere locali.

```
virtual_maps = hash:/etc/postfix/virtual, ldap:aliases, ldap:accountsmap
local_recipient_maps =
```

Definiamo la quota che deve essere verificata su LDAP prima di accettare la mail e il messaggio che riceverà il mittente in caso di superamento della stessa.
Questi parametri funzionano solamente se sul codice sorgente di postfix risulta applicata la patch VDA.

```
quota_server_host = localhost
quota_search_base = o=hosting,dc=example,dc=com
quota_query_filter = (&(&(objectClass=JammMailAccount)(mail=%s))(accountActive=TRUE)\
        (delete=FALSE))
quota_result_attribute = quota
quota_cache = no
quota_bind = no
```

Il messaggio che riceverà il mittente.

```
virtual_maildir_limit_message = "The user you're trying to reach is over quota."
virtual_mailbox_limit_maps = ldap:quota
virtual_mailbox_limit_override = yes
virtual_mailbox_limit_inbox = yes
```

Ora il postfix è pronto per essere avviato.

Courier-imap

Il courier-imap ci permette di mettere a disposizione degli utenti la posta tramite i protocolli POP e IMAP, compresi i loro corrispondenti over SSL.

Configuriamo il file principale di courier /usr/lib/courier-imap/etc/authdaemonrc per permette l'autenticazione su LDAP e modifichiamo la seguente riga:

authmodulelist="authldap authpam"

Definiamo a che server LDAP deve essere fatta la richiesta in /usr/lib/courier-imap/etc/authldaprc e modifichiamo le seguenti righe:

```
LDAP_SERVER         localhost
LDAP_PORT           389
LDAP_BASEDN         o=hosting,dc=example,dc=com
LDAP_TIMEOUT        5
LDAP_AUTHBIND       1
LDAP_MAIL           mail
LDAP_GLOB_UID       vmail
LDAP_GLOB_GID       vmail
LDAP_HOMEDIR        homeDirectory
LDAP_MAILDIR        mailbox
LDAP_MAILDIRQUOTA   quota
LDAP_FULLNAME       cn
LDAP_CLEARPW        clearPassword
LDAP_CRYPTPW        userPassword
LDAP_DEREF          never
LDAP_TLS            0
```

Vengono definiti tutti i parametri di interfacciamento con il servizio di LDAP locale e l'utente che deve essere usato per le operazioni sul filesystem.

Per avviare i servizi anche con il supporto di SSL si consiglia la lettura delle pagine di manuale dei pacchetti e della documentazione in linea presente su http://www.inter7.com/courierimap.html

Per la generazione dei certificati e per quello che riguarda in genere il protocollo SSL consiglio la consultazione di http://www.openssl.org/docs/ e http://www.openssl.org/related/.

Ora courier-imap è pronto per essere avviato.

Tomcat4

Questo è l'esecutore delle classi Java contenute in JAMM e in questo caso è stato installato con un pacchetto rpm.
Le operazioni che andremo a compiere sono esclusivamente per la messa in sicurezza e per lo spostamento dell'interfaccia di amministrazione su di un protocollo over SSL.

La home directory dell'utente tomcat è /var/tomcat4.
Editiamo il file /var/tomcat4/conf/tomcat-users.xml e rimuoviamo le seguenti righe:

```
<user username="tomcat" password="tomcat" roles="tomcat"/>
<user username="role1" password="tomcat" roles="role1"/>
<user username="both" password="tomcat" roles="tomcat,role1"/>
```

Generiamo un certificato self-signed per tomcat.

 # /usr/java/j2sdk[version]/bin/keytool -genkey -keyalg RSA -alias tomcat -keystore keystore_filename

Il *keystore_filename* indica il file con il PATH completo dove deve essere scritto il certificato.
Dobbiamo tenere traccia della passphrase richiesta in quanto deve essere inserita all'interno del file di configurazione di tomcat.

Ora modifichiamo la configurazione di /var/tomcat4/conf/server.xml, e disabilitiamo l'ascolto del servizio sulle porte standard 8009 e 8080, successivamente attiviamo quello over SSL sulla 8443.

Commentiamo le seguenti righe per disabilitare 8009 e 8080:

```
<Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
        port="8080" minProcessors="5" maxProcessors="75"
        enableLookups="false" redirectPort="8443"
        acceptCount="100" debug="0" connectionTimeout="20000"
        useURIValidationHack="false" disableUploadTimeout="true" />
```

```
<Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
        port="8009" minProcessors="5" maxProcessors="75"
        enableLookups="true" redirectPort="8443"
        acceptCount="10" debug="0" connectionTimeout="0"
        useURIValidationHack="false"
        protocolHandlerClassName="org.apache.jk.server.JkCoyoteHandler"/>
```

Decommentiamo le seguenti righe per attivare la 8443:

```
<Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
        port="8443" minProcessors="5" maxProcessors="75"
        enableLookups="false"
        acceptCount="100" debug="0" scheme="https" secure="true"
        useURIValidationHack="false" disableUploadTimeout="true">
        <Factory className="org.apache.coyote.tomcat4.CoyoteServerSocketFactory"
        clientAuth="false" keystoreFile="keystore_filename" keystorePass="passphrase" \
        protocol="TLS" />
  </Connector>
```

Ora anche tomcat risulta essere pronto per essere avviato, ma prima bisogna configurare JAMM.

JAMM

Procediamo all'installazione di Jamm, poniamo che il pacchetto di jamm risieda in /root.
Eseguiamo nell'ordine le seguenti operazioni:

```
# tar -xzvf jamm[versione]-bin.tar.gz
# cd jamm[versione]
# cp jamm.schema /etc/openldap/schema
# mkdir /var/tomcat4/webapps/jamm
# cd /var/tomcat4/webapps/jamm
# /usr/java/j2sdk[versione]/bin/jar -xvf ~/jamm[versione]/jamm[versione].war
# cd WEB-INF
# cp jamm.properties.dist jamm.properties
```

Viene quindi creata la directory dove vengono contenute le classi che dialogano con il servizio di LDAP locale.

Per configurare il jamm editiamo jamm.properties in questa maniera:

```
jamm.ldap.host = localhost
jamm.ldap.port = 389
jamm.ldap.search_base = o=hosting,dc=example,dc=com
jamm.ldap.root_dn = cn=manager,dc=example,dc=com
jamm.ldap.root_login = manager@example.com
```

Quindi il nostro account di amministrazione di Jamm sarà "manager@example.com" e la sua password sarà quella del rootdn.

Come dicevamo dobbiamo analizzare ora tre situazioni, poiche jamm ci permette di creare, modificare o cancellare degli account, ma queste variazioni saranno presenti solo sul db di ldap.

Caso 1: Creazione nuovo dominio con account
Vengono effettuate le variazioni sul db, ma la struttura dell'archivio non è ancora presente, quindi se un utente tenta di scaricarsi la posta ottiene solamente un messaggio di errore anche se utilizza la password corretta.
Questo perchè courier cerca di accedere ad una struttura dati non ancora esistente.
Bisogna quindi creare una struttura mandando per esempio un messaggio di benvenuto, questo primo invio forza l'utente vmail utilizzato da postfix a scrivere tutte le directory dati.

Caso 2: Cancellazione di un dominio o di un account
Vengono effettuate le variazioni sul db di LDAP, ma jamm non ha il permesso di scrivere su filesystem per ragioni di sicurezza.
Dobbiamo prevedere un lavoro in cron che ad intervalli regolari analizzi i campi in LDAP e cancelli quelli

richiesti.
Per fare questo esiste nel pacchetto Jamm un file di nome jammUtils[versione].tar, quindi le operazioni sono le seguenti:

 # cd
 # cp jamm[versione]/jammUtils[versione].tar ~vmail
*# chown vmail:vmail ~vmail/jammUtils**
 # su – vmail
$ *tar -xvf jammUtils[versione].tar*

Ora non resta che creare uno script per la pulizia del filesystem dai domini o dagli account cancellati usando il binario jammCleaner.

Caso 3: Cancellazione di un alias
Il dato è solamente residente su LDAP, praticamente è un forward di una mail e quindi si elimina subito.

Purtoppo il campo quota non è ancora presente sull'interfaccia di Jamm e non viene nemmeno inserito in modo automatico, al momento potete utilizzare uno script che inserisca il valore della quota tramite il comando ldapadd ed un file con un contenuto simile a questo:

---

dn: mail=utente1@dominio.tld,jvd=dominio.tld,o=hosting,dc=example,dc=com
changetype: modify
replace: quota
quota: 10240000

Per la quota di 10Mb della casella di posta.

Squirrelmail

Squirrelmail non richiede particolari accorgimenti, se non quello di verificare che sia disponibile solamente via protocollo HTTPS per evitare che i dati circolino in chiaro su Internet.

Conclusioni

Basta ora il riavvio del servizio di tomcat4 e il nostro server funziona con l'interfaccia di amministrazione sulla porta 8443.
Vi manca solamente la configurazione del server web apache per far si che tutti i domini siano mappati correttamente, ma non è argomento di questo scritto.

Tutti i pacchetti ricompilati qui descritti e anche questo documento sono disponibili per il download all'indirizzo http://open.rhx.it