Implementazione del *single sign–on* per l'accesso a sistemi, applicazioni, database e portali della intranet

Giuseppe Sacco

5 maggio 2006

Sommario

"Single sign–on" è una tecnologia che permette all'utente di avere una sola chiave di accesso ai vari applicativi e, in un secondo momento, di effettuare l'autenticazione una sola volta pur accedendo a tutta una serie di risorse e applicazioni. Se all'inizio c'era solo una applicazione, ad esempio per la contabilità, adesso gli si sono affiancati l'utenza del sistema operativo, quella della posta elettronica, quella sul nuovo gestionale, ed altre ancora. Tutte queste applicazioni richiedono di verificare l'identità dell'utilizzatore e in genere lo fanno in base a un "database" interno nel quale sono memorizzate le credenziali (*login* e *password*) e altri dati che riguardano l'identità dell'utente e i diritti di accesso all'applicazione. La gestione di tutti questi "database" è onerosa sia per l'utente che deve ricordarsi ogni sua credenziale, sia per l'amministratore che deve aggiungere o modificare i dati di una persona in molti posti. Ci sono varie tipologie di "single sign–on" alcune prevedono l'interposizione di un filtro tra l'utente e l'applicazione, altre invece richiedono che l'applicazione sia a conoscenza del metodo di autenticazione utilizzato.

1 Autenticazione esterna

Il primo passo verso la semplificazione della gestione delle identità è quello di affidarsi ad un gestore unico. Si tratta di un passo che richiede sia delle modifiche architetturali all'azienda, che deve dotarsi di tale servizio di autenticazione, sia delle modifiche alle applicazioni, che devono usare tale servizio.

In realtà, ormai gran parte delle applicazioni esistenti sul mercato permette di gestire l'autenticazione centralizzata. Lo fanno i sistemi operativi, capaci di collegarsi ad un server NIS, NIS+, LDAP, Kerberos o NTLM. Lo fanno i server di posta elettronica come `exim`, che sanno autenticare sul sistema operativo, oppure su utenti memorizzati in un DBMS. Lo fanno i server proxy come `squid` (che possono fungere da autenticatore per tutte le applicazioni web) capaci di autenticare su sistemi LDAP o NTLM. Lo fanno gli *application server* java, come `WebSphere` tramite JAAS e, per il secondo passo tramite GSS-API.

Probabilmente in ogni azienda ci sono delle macchine Windows che fanno parte di un dominio NTLM. Da questo punto è molto facile passare a dare l'accesso alle risorse di rete, anche se queste sono gestite dalla controparte open source chiamata `samba`. Il passo successivo è di norma quello di inserire nella gestione centralizzata gli account di posta elettronica e le rubriche personali e condivise degli utenti. Questo secondo passo di norma richiede il passaggio ad un *directory service* che in genere è LDAP, tramite la sua implementazione più conosciuta, `openldap`. Nel server LDAP possono essere memorizzate tutte quelle informazioni che prima erano nel server NTLM (che a volte è esso stesso un server LDAP nascosto dal nome ActiveDirectory.) Inoltre l'adozione di rubriche di indirizzi condivise aiuta spesso la gestione dei dipartimenti di una azienda e permette di avere sempre *online* le informazioni che prima risiedevano sui computer delle singole persone.

Chi già avesse sistemi operativi GNU/Linux o Unix non troverà difficoltà nel passaggio all'uso di LDAP e, probabilmente otterrà "gratis" anche il passaggio di alcune delle applicazioni che ci girano sopra e si basano sull'autenticazione del sistema operativo. Si possono autenticare su server LDAP, eventualmente tramite il sistema operativo,

gli applicativi scritti in java (usando `JAAS`), i database postgresql (tramite PAM), informix (tramite PAM), e oracle (tramite `RADIUS`).

Per quegli applicativi che non permettono l'autenticazione, nel senso che non hanno una gestione degli utenti, l'unica cosa da fare è interporre all'applicativo un altro programma che si occupi dell'autenticazione dell'utente. Nel caso di programmi utilizzabili da *browser*, quindi con una interfaccia web, è possibile utilizzare un *proxy* che richieda le credenziali prima di permettervi l'accesso. Nel caso di applicativi con interfaccia `ASCII` è normalmente utilizzato un programma, detto *screen scraper*, che finge di essere una persona e inserisce le credenziali prima di farvi accedere l'utente.

1.1 PAM

Il *Pluggable Authentication Module* è un meccanismo in uso sui vari sistemi GNU/Linux e Unix che permette di avere una astrazione dal vero sistema di autenticazione. L'applicazione deve solo preoccuparsi di utilizzare la libreria PAM che utilizza i file di configurazione opportunamente compilati per eseguire l'autenticazione sul sistema corretto. Esistono dei metodi per autenticare su file `/etc/passwd`, NIS, LDAP, database SQL e altro ancora.

La grande utilità di PAM è proprio la sua modularità che da un lato permette a chi sviluppa l'applicazione di non preoccuparsi dell'autenticazione dell'utente e dall'altro permette di avere un solo punto nel quale sono definite le identità degli utenti.

Il punto a sfavore di PAM rispetto all'uso di LDAP è che tramite PAM si possono solo controllare le credenziali come *username* e *password*, mentre con LDAP è possibile avere tutta una serie di informazioni associate a queste credenziali. Per questo, se è necessaria la gestione delle identità è opportuno accedere direttamente a LDAP.

1.2 RADIUS

RADIUS è un altro meccanismo analogo a quello del PAM. Non si tratta di una libreria, ma di una applicazione vera e propria, con un suo server, che accetta le richieste dalle altre applicazioni e verifica le credenziali. L'implementazione open source di *RADIUS*, `freeradius`, permette di eseguire poi l'autenticazione su diversi *backend* come LDAP, postgresql, MySQL e lo stesso meccanismo PAM.

Poiché si tratta di un servizio TCP/IP, *RADIUS* ha anche un protocollo sicuro di comunicazione tra le applicazioni e il server stesso.

1.3 JAAS

JAAS, come molte delle infrastrutture di Java, è il *porting* all'interno del mondo java di qualcosa che già esiste nel mondo dell'informatica. In questo caso particolare si tratta della copia di PAM per le applicazioni Java. Oltre all'autenticazione delle credenziali, *JAAS* fornisce una infrastruttura per le autorizzazioni, vale a dire per l'esecuzione di operazioni privilegiate.

2 Kerberos

Quando si parla di sistemi di autenticazione si finisce spesso con il parlare di *Kerberos*. Si tratta di un protocollo sviluppato al MIT negli anni '80 e divenuto uno degli standard per l'autenticazione sicura. In questo momento tutti i maggiori Unix e anche Windows 2000 hanno la possibilità di autenticare un utente o un servizio su un server *Kerberos*. Il protocollo è pubblico e il MIT ha sviluppato una sua implementazione open source che è molto diffusa.

Al contrario di molti meccanismi di autenticazione che possono essere trasparenti all'applicazione, *Kerberos* richiede una gestione specifica, non tanto per l'autenticazione dell'utente quanto per il passaggio delle credenziali da una applicazione all'altra. Ovviamente non tutto va scritto da zero e ci sono varie librerie che permettono di accedere abbastanza agevolmente a questo tipo di autenticazione. In particolare per quanto concerne Java, è stata sviluppata una libreria chiamata GSS-API per lo scambio di credenziali tra applicazioni (che in questo momento supporta solo *Kerberos*.)

Per l'implementazione completa del "single sign–on," che quindi prevede il passaggio da una applicazione all'altra mantenendo l'identità e non dovendo reinserire la *password*, si deve necessariamente utilizzare *Kerberos*, oppure sviluppare un meccanismo analogo.