Single Sign On

Single Sign On

Single Sign On (SSO) is used to mean several different things. I'll describe (my understanding of) the dimensions and some possible solutions and technologies.

What does SSO Mean?

For now, I'll just talk about web app SSO

Several apps on one server

BASIC/DIGEST authentication

RFC 2617 says

> *The realm directive (case-insensitive) is required for all authentication schemes that issue a challenge. The realm value (case-sensitive), in combination with the canonical root URL (the absoluteURI for the server whose abs_path is empty; see section 5.1.2 of 2) of the server being accessed, defines the protection space.*

This means that once the client gets your credentials for a server and realm name it will keep supplying them to every app on that server with that realm name. So you can get SSO for all servers that appear to be the same one to the client (such as accessed through a load balancer) and with the same realm. This won't work for several apps on servers that appear to be different to the client.

FORM authentication

The servlet spec says that session info is restricted to a single web app identified by its context root. Normally form auth is implemented by storing the credentials in the http session, so you cant share these credentials across web apps. Web containers often provide a way to violate the spec and share sessions across context roots sharing some initial URI. Jetty 7 may provide a cross context psuedo session that can be configured for arbitrary apps that can be used to store the creds for all the apps.

From the client side perspective form authentication is usually implemented with the aid of cookies that mention the domain and path. At least in jetty these are configurable although non-default path values are not servlet spec compliant as they would result in sharing a session across several web apps.

More than one server, with different apps

It doesn't appear possible to have BASIC or DIGEST sso in this case if the client can recognize that there are different servers (i.e. not concealed behind a load balancer).

For FORM-like authentication, more is possible using cookies. The servers must be on related domains, e.g. a.foo.com and b.foo.com so the cookie domain can be foo.com. Similarly the paths must be compatible, such as "/". In these cases the user agent will send the same cookie to all the servers in the related domains when accessing a compatible path. The cookie can contain either the id for a cross-context psuedo "session" or the security info itself. If the security info (identity) is encrypted I don't see how sending the security info would be less secure than sending the pseudo-session id. However it could easily exceed the cookie size limit.

A cross context psuedo-session would need to be replicated on all the servers either on demand when a request arrives with a unknown session id or more preemptively with a clustering solution such as tribes, wadi, or terracotta.

Other methods

Kerberos

In a kerberos environment where the clients are already authenticated using kerberos SPNEGO can theoretically be used to automatically authenticate users to all servers that are part of the kerberos environment. AFAIK this requires basically JASPI support with a SPNEGO auth module and a SPNEGO compatible user agent. JASPI support should be available in jetty 7 (currently available in an experimental branch).

OpenID

OpenID provides some SSO like features in that only one server (the openid provider) actually logs you in, however you have to indicate your openID identifier to each openid enabled server you wish to access and you have to tell the openid provider you want to allow authentication to each such server.

SSO Implementations

JSecurity

JSecurity or more usefully http://www.jsecurity.org/ is a well established security framework that is now moving to apache and is currently under incubation. It's ASL 2.0. It is not yet integrated in Geronimo and does not appear to directly relate to javaee spec security yet although it may be useful to integrate as a JACC provider. Some details on SSO support are at http://www.jsecurity.org/node/996 which appear to indicate that it relies on a cross-context psuedo session and would need a custom communication with a particular server to share session contents.

CAS (Central Authentication Server)

CAS is a BSD licensed framework that is kerberos like. A user signs onto the central server, which maintains a cookie and session remembering the login. Applications redirect to the central server for authentication and the central server redirects back after authentication, including a ticket. The client app then validates the ticket against the central server. Apparently the only information returned is "validated" and the user id; there is no role information returned. Thus, as with openid, each application will need to maintain identity to role mapping. There is a spring security integration but I can find no direct jetty or tomcat integrations that allow use of the javaee security constraints.

JOSSO

JOSSO is GPL licensed which makes its use with ASL licensed code difficult to understand for me. It appears to have a geronimo-tomcat integration. Use with jetty or geronimo-jetty would require modifying the jetty server to include an interceptor or handler before the security handler so the sso information could be injected before the normal security handling. Presumably this code would be GPL thus making the use of Jetty or Geronimo GPL. For this reason Geronimo can't include integration code for JOSSO.
JOSSO works by having a central authentication server ("gateway") and "sessions" that are shared between the auth server and secured "partner" apps. Presumably the session id is shared using a cookie. There is some combination of an authentication message flow that redirects to the central auth server and a JAAS login module for the partner app that retrieves the auth info from the central server using SOAP. It's possible that the authentication message flow, currently implemented apparently via a tomcat valve, could be implemented using JASPI, thus making it more cross-platform. There's some description here:
http://www.josso.org/confluence/display/JOSSO1/Architecture+Overview

ESOE

ESOE is ASL 2.0 but apparently written primarily in C++. It supports standards such as SAML and XACML. It's unclear how it relates to the above discussion. There's a tomcat integration but no visible jetty or geronimo integration. It relies on a central authentication server.