NativeSPProtectContent

There are two ways that you can use the SP to protect content:

- Actively, by intercepting requests for particular resources and ensuring that a valid, authenticated session exists between the user agent and the SP software before passing along the request.
- Passively, by publishing information about valid, authenticated sessions through CGI, but passing unauthenticated requests through unmolested.

In both cases, the information about the session supplied by the SP is provided uniformly so that applications can be programmed to respond dynamically based on the information. The NativeSPAttributeAccess topic describes this mechanism in detail.

Active Protection

In the "active" mode, static web pages can be protected, and dynamic applications can be written to assume that authentication is done before they execute. It's also possible to apply static access control rules that restrict access based on the information associated with a session.

To impose active session requirements, you attach the requireSession content setting to the resource. Mechanisms for doing this include native configuration approaches (Apache, Sun/iPlanet) and a generic configuration mechanism used with IIS or FastCGI.

Passive Protection

In the "passive", or lazy session mode, the same set of resources can be exposed to both authenticated and unauthenticated access at the same time. This is obviously incompatible with both static resources (which would be left exposed, making authentication moot) and static access control (which would deny unauthenticated access). Thus, it's useful for dynamic applications that typically want to offer a "guest" mode by default, or possibly support multiple forms of authentication, and initiate a login via SAML only when desired or chosen by the user.

When using passive protection, you do NOT apply the requireSession content setting to the resource, but merely ensure that the SP software is active for the request (or often simply for the entire virtual host). For details, refer to the appropriate web server configuration topic (Apache, IIS, Sun/iPlanet, Fast CGI).

To determine whether a session exists with your application, you can check for the presence of one of the "fixed" CGI variables or headers, such as "Shib-Identity-Provider"/"HTTP_SHIB_IDENTITY_PROVIDER".

If you want the user to login and begin a session, your application must issue an HTTP redirect to the location of a SessionInitiator handler using parameters described here. In a "modern" SP configuration, the location to redirect to is usually "/Shibboleth.sso/Login".