ShibbolethWalkthroughStepTwo

To the IdentityProvider: UserAuthentication and the SingleSignOnService

The goal of this step is for the client to deliver the AuthnRequest from the SP to the SingleSignOnService at the !IdP so that it can generate a response message, generally using either the BrowserPOST or BrowserArtifact profile.

Today in Shibboleth, UserAuthentication is actually not part of the !IdP software itself. The SingleSignOnService runs as a Java application and the deployer's responsibility is to protect the SingleSignOnService with some form of web authentication, either using the Java container (e.g. Tomcat) or the web server itself.

If no UserAuthentication is done, access to the SingleSignOnService will be blocked because the user identity can't be determined. This results in the NoPrincipalName error.

Otherwise, the user's identity will be extracted from REMOTE_USER or some other specified header, and used as the "internal" principal name for the rest of the request. This is a value expected to be usable by the NameIDMapper and AttributeResolver subsystems to construct the SAML NameIdentifier and attribute set to give to the SP.

Processing the AuthnRequest

In Shibboleth today, the AuthnRequest messages defined by the Shibboleth profile are simple query strings containing three primary pieces of data:

- unique name of the SP (=providerId=)
- location of the AssertionConsumerService to use for the response message (=shire=)
- RelayState value to return to the SP (=target=)

RelyingParty Determination

The name of the SP is the primary input into the process for configuration purposes. Much as the SP configuration process is oriented around ShibbolethApplications, the !IdP configuration is expressed primarily in terms of RelyingParty groupings, which allows settings to be tuned at various levels of granularity.

The "name" of a RelyingParty is either a specific SP's name (its providerId) or the name of a grouping of SPs based on the MetaData loaded into the !IdP. The MetaData can contain nested grouping elements called an <`EntitiesDescriptor`> .

So, based on the `providerId` parameter in the AuthnRequest, the !IdP can locate the MetaData for the SP, and determine the most-specific matching RelyingParty to apply for configuration purposes. This determines many aspects of the resulting response, including how the principal's name is turned into a SAML NameIdentifier, whether AttributePush is used, and how the !IdP identifies itself to the SP, KeysAndCertificates used, if any, etc.

Currently, if no MetaData for the SP is found, the SP is treated with a special designation called "Unauthenticated", and the RelyingParty indicated by the d`efaultRelyingParty` attribute from the !IdP's configuration is applied.

Phishing Protection

The other significant part of AuthnRequest processing is a precaution to ensure that the information produced for the SP is actually going to that SP. Since there is no direct connection, the !IdP relies on MetaData to determine whether the AssertionConsumerService specified in the AuthnRequest is in fact "controlled" by the SP specified in the request. This prevents a blatant phishing attack against the user and allows the !IdP to release attributes in the case of AttributePush, even though the AuthnRequest could actually have been sent by anybody.

When the MetaData doesn't provide a match, the !IdP fails the request with an InvalidACS error.

Note that if multiple sources of MetaData are available, it is undefined what happens if the SP has a valid entry in more than one of them. Generally, one of the entries will be used to the exclusion of any others, but this isn't defined behavior.

Constructing the Response

In ShibOnedotThree, the two primary characteristics of the response to an AuthnRequest are which SAML 1.1 profile to use, and whether to utilize AttributePush.

%COMMENT%