--
Transcript
--

Hello and welcome to a quick tutorial on how to configure the SAML2 authentication module for auto-federation using OpenAM 13.

We're going to walk through this from first principles, so you're going to see us configuring a service provider, an identity provider and finally then configuring the authentication module as part of a login chain. First of all we're going to create our SAML2 service provider.

Here you can see that we're configuring a new service provider, and giving it the circle of trust named 'cot'. We don't want to register the remote provider, because first of all we want to alter some of the settings in the just-created service provider. Head into the Federation tab and click on your service provider, and head to the services tab. Here we have to alter the assertion consumer service by adding the word 'Auth' in front of it. This simply tells us when returning to the assertion consumer service that we need to return to a specific location in OpenAM which knows how to handle the authentication process and hook you back into the rest of the remaining authentication chain.

Next, we make sure that auto federation is enabled - select the auto federation checkbox, and supply the 'uid' attribute which acts to tell OpenAM of the name of the attribute that the IdP will be sending the SP which we should use as the unique user identifier. We also need to tell the SP that it should map the attributes received from the IdP that will be used to generate the user's federated account on the SP. We want to generate our user in the SP with the same attribute names as those in the IdP, so we'll use the special '*=*' mapping which means "for each attribute in the assertion, generate an attribute with that same name and value and store it in the datastore if generating a local user account".

So, let's head over to another OpenAM 13 instance we have ready to act as our SAML2 identity provider. We're using the test signing key that comes with OpenAM, and we're creating another circle of trust here called 'cot' also.

On the IdP's side, we need to expose the attributes that we want to share with the SP. By filling in the attribute map with two

values - ‘uid=uid’ and ‘mail=mail’ we tell the IdP to include the user’s ‘mail’ attribute in the assertion that we send to the SP under a field named ‘mail’. As the SP was configured the the ‘*=*’ mapping, the ‘mail’ attribute in our assertion will be dynamically mapped to a ‘mail’ attribute in our SP’s datastore. The ‘uid’ field will be used as our unique user identifier.

Finally for the identity provider we need to register the remote service provider. So, we enter the URL that you can find through OpenAM’s documentation which will export the meta data necessary for the identity provider to be familiar with the service provider.

We’ll also check that we have a user with a ‘mail’ attribute configured in the identity provider, which we’ll use in the example run.

Back in the SP’s OpenAM 13 we can see that we are now about to register the remote identity provider, and this is a very similar process to that we just followed in the identity provider. We simply use the meta data URL for the identity provider and click okay.

Now it’s time to create our authentication module.

You can see that you can set the usual OpenAM authentication level at the top. Below that you can set the identity provider’s entity ID, and we’ll set this to the just-created identity provider over in the OpenAM 13 instance. We don’t need to set a linking chain in this example, but we need to ensure that dynamic account creation is configured for the realm’s authentication setting. This allows OpenAM to dynamically generate a user based on the user-federated data.

Having configured that, we’ll quickly check the list of users in the SP, to ensure that a new user is generated the first time we execute the chain for the idpUser account.

We’ll log out and try to log in using our newly-configured authentication chain. First of all, we’ll set the serviceName to ‘samlChain’. You’ll see us immediately redirected over to the identity provider to authenticate. Then we’re redirected back to the SP - and we’re logged in! We can check over the federated data - here we can see that the user’s mail address has been moved over while none of the other attribute’s we saw have been, just as we dictated to the IdP.

So, this user now has a persistent account on the SP, but authenticates using the IdP. We'll quickly log in as the admin to check that this user account was generated correctly on the SP.

And that completes our brief introduction to using the SAML2 authentication module in OpenAM 13 using auto federated user data. If you have further questions or wish to see more examples of how to configure this module, please refer to the knowledge base, documentation or contact support.

Thank you.