Implementing MFA Using Native IDPv3.3 Duo Plugin (for IDPs who upgraded from 2.x)

In this article we will discuss how to get the native Duo plugin working with shibboleth v3.3. There are a few other articles that discuss this, as well as some shibboleth forum posts, but we had trouble getting those to work with our configuration. In our configuration we used the IDP upgrade script to go from 2.4 to 3.2.1 and then went to 3.3 from there before attempting to integrate with Duo. Because we upgraded from 2.4 to 3.2 some of our config files were missing pieces that come standard in a fresh 3.3 install.

Step-by-step guide

First, follow the instructions on the Duo shibboleth documentation to generate the ikey, skey and akey. Enable a shibboleth application in the Duo admin panel, populate it with the appropriate keys and copy the apiHost. Put all of these into our IDP configs by editing **[idp]/conf/authn/duo.properties**. Populate the apiHost, applicationKey, integrationKey and secretKey. Discuss with your security folks about whether to fail open (`idp.duo.failmode = safe`) or closed (`idp.duo.failmode = secure`).

Next make sure that that properties file is being referenced by the top level **[idp]/conf/idp.properties**. The variable `idp.additionalProperties` should contain `/conf/authn/duo.properties`. Also, there have been some security rumblings about Duo MFA in shibboleth IDP being defeated, and the current thoughts are that we should set `idp.authn.identitySwitchIsError = true` in **idp.properties**. We haven't seen any uptick in errors since turning it on, but do your own research. Finally, in **idp.properties** file add MFA to the list of available authn flows. For us, we wanted to allow SPs to choose whether to use a password flow, an ECP flow, or a Duo flow: `idp.authn.flows = Password|MFA|RemoteUserInternal`. More on how SPs communicate that choice later.

The MFA authn flow is defined by **[idp]/conf/authn/mfa-authn-config.xml**. More info on this file can be found in the Shibboleth IDP wiki. We configured ours so that if an SP chooses to request the MFA flow we always invoke password authn and then Duo authn. We left in the scripted contextFunction in case we need to extend this functionality later. All together it looks like this:

```
<util:map id="shibboleth.authn.MFA.TransitionMap">
  <!-- First rule runs the Password login flow. -->
    <entry key="">
      <bean parent="shibboleth.authn.MFA.Transition" p:nextFlow="authn/Password" />
    </entry>
  <!-- Second rule runs a function if Password succeeds, to determine whether an additional factor is required. --
>
   <entry key="authn/Password">
     <bean parent="shibboleth.authn.MFA.Transition" p:nextFlowStrategy-ref="checkSecondFactor" />
   </entry>
   <!-- An implicit final rule will return whatever the final flow returns. -->
</util:map>
<!-- Example script to see if second factor is required. -->
<bean id="checkSecondFactor" parent="shibboleth.ContextFunctions.Scripted" factory-method="inlineScript">
  <constructor-arg>
    <value>
      <![CDATA[
        nextFlow = "authn/Duo";
        nextFlow;   // pass control to second factor or end with the first
      ]]>
    </value>
  </constructor-arg>
</bean>
```

Duo has it's own velocity template that needs to be edited so that it looks like the rest of your templates. It lives at **[idp]/views/duo.vm**. Besides skinning this to look like the rest of your IDP, we needed to make the iframe a bit bigger by setting a style attribute on the iframe tag: `style="width: 100%; height: 400px;"`.

Finally, edit **[idp]/conf/authn/general-authn.xml**. This part was different (for us) from all the resources I could find online, and it seems to be because **general-authn.xml** looks different if you've upgraded from 2.4 versus installing 3.2 from scratch. Two flows must be defined / edited in this file. The MFA flow is referenced by **idp.properties** authnflows setting. The Duo flow is referenced by the MFA flow. As the comments in **general-authn.xml** suggest, the MFA flow should be a concatenation or union of all the other subflows referenced in the **mfa-authn-config.xml**. Since **mfa-authn-config.xml**references Duo and Password authn, the MFA supportedPrincipals should be all the supportedPrincipals that apply to either password authn or Duo authn. The supportedPrincipals for Duo authn haven't been formally agreed on yet, so while supportedPrincipals that reference most auth methods take the form of `urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport`, MFA supportedPrincipals currently resemble something like `http://example.org/ac/classes/mfa`. This is ok because we can have multiple supportedPrinciples tied to an authn flow. So as inCommon releases further guidance on how to properly federate Duo MFA, we can add those supportedPrincipals here as well. SPs that request the example.org supportedPrincipal will still work as long as they're left in **general-authn.xml**. Putting all that together gives us these two beans in the general-autn.xml file:

```
    <bean id="authn/Duo" parent="shibboleth.AuthenticationFlow" p:forcedAuthenticationSupported="true" p:
nonBrowserSupported="false">
     <!--
      The list below should be changed to reflect whatever locally- or
      community-defined values are appropriate to represent MFA. It is
      strongly advised that the value not be specific to Duo or any
      particular technology.
    -->
    <property name="supportedPrincipals">
      <list>
        <bean parent="shibboleth.SAML2AuthnContextClassRef" c:classRef="http://example.org/ac/classes/mfa" />
        <bean parent="shibboleth.SAML1AuthenticationMethod" c:method="http://example.org/ac/classes/mfa" />
     </list>
   </property>
</bean>

<bean id="authn/MFA" parent="shibboleth.AuthenticationFlow" p:passiveAuthenticationSupported="false" p:
forcedAuthenticationSupported="true" p:nonBrowserSupported="false">
  <!--
    The list below almost certainly requires changes, and should generally be the
    union of any of the separate factors you combine in your particular MFA flow
    rules. The example corresponds to the example in mfa-authn-config.xml that
    combines IPAddress with Password.
  -->
  <property name="supportedPrincipals">
    <list>
      <bean parent="shibboleth.SAML2AuthnContextClassRef" c:classRef="urn:oasis:names:tc:SAML:2.0:ac:classes:
PasswordProtectedTransport" />
      <bean parent="shibboleth.SAML2AuthnContextClassRef" c:classRef="urn:oasis:names:tc:SAML:2.0:ac:classes:
Password" />
      <bean parent="shibboleth.SAML1AuthenticationMethod" c:method="urn:oasis:names:tc:SAML:1.0:am:password" />
      <bean parent="shibboleth.SAML2AuthnContextClassRef" c:classRef="http://example.org/ac/classes/mfa" />
      <bean parent="shibboleth.SAML1AuthenticationMethod" c:method="http://example.org/ac/classes/mfa" />
    </list>
  </property>
</bean>
```

So that’s the IDP configurations. SPs (at least 2.4 SPs) can now use this setting in a Location block or htaccess file when enabling shib protection to request MFA: `ShibRequestSetting authnContextClassRef "http://example.org/ac/classes/mfa"`.

Related articles

Openclemson Shibboleth Blog

- Implementing MFA Using Native IDPv3.3 Duo Plugin (for IDPs who upgraded from 2.x)
- Replicating Multi-Context Broker Functionality (Duo + Username/Password with user-opt-in forcing Duo)