Michigan State University

Implementing SAML in Java Web Application

Rob Tucker
08/23/2020

SAML in Java Web Application

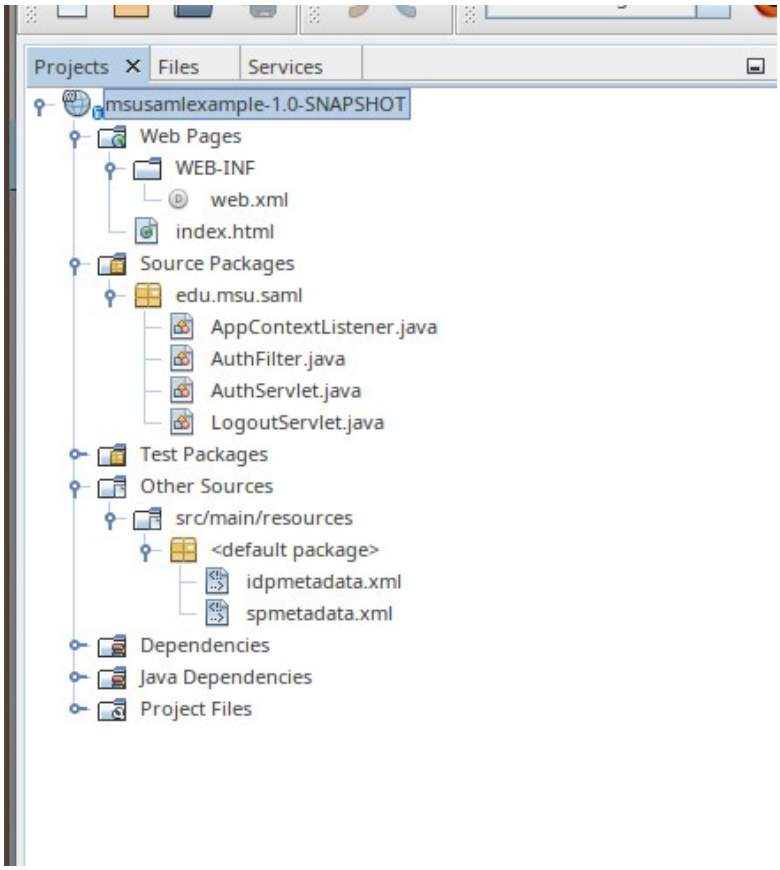
Contents

OVERVIEW

This document describes a simple SAML authentication implementation in a Java Web Application. This example uses a Java maven build and uses the com.coveo.saml.SamlClient library to perform the SAML communications. The com.coveo.saml.SamlClient is available on github at https://github.com/coveooss/saml-client.git and can be pulled in via maven by adding the following depedency:

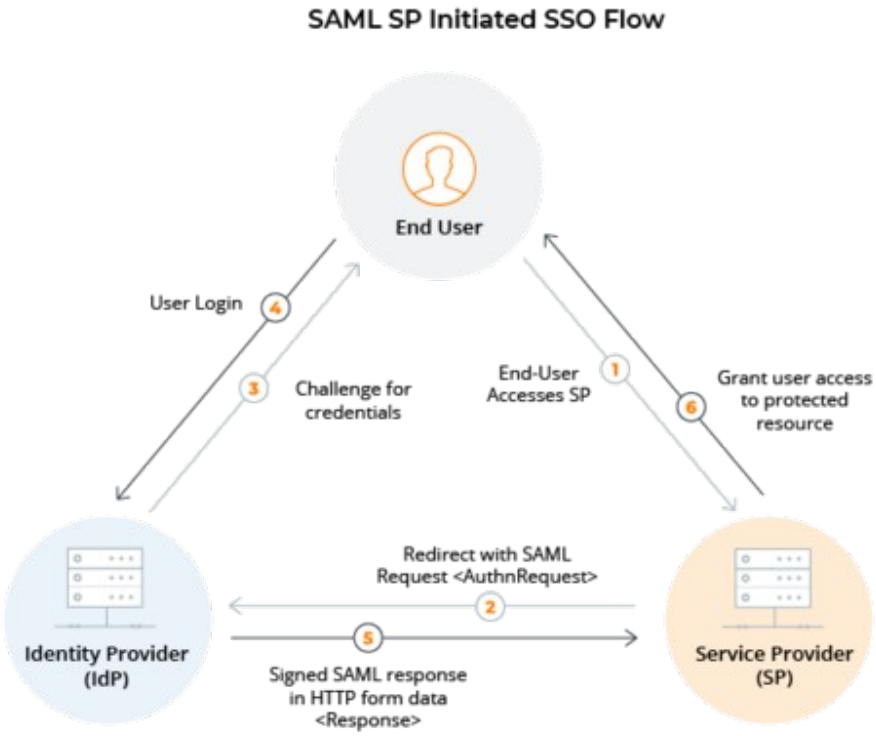
```
<dependency>
    <groupId>com.coveo</groupId>
    <artifactId>saml-client</artifactId>
    <version>3.0.2</version>
 </dependency>
```

This maven project layout is shown below:



SAML Communication Flow

In this example the msusamlexample Service Provider (SP) communicates to the SAML Identity Provider (IDP) for SSO authentication. If the SP has not previously authenticated the IDP will display a login. If login credentials are verified the IDP will callback to the configured AssertionConsumerService on the SP with with any configured attributes.



SAML IDP/SP Metadata

As SAML technology has matured, the importance of SAML metadata has steadily increased. Today an implementation that supports SAML web browser requires a schema-valid SAML metadata file for each SAML partner. In this example we have idpmetadata.sml and spmetadata.xml. The IDP metadata can be pulled from the IDP server you are using. The SP metadata is used to define communication parameters for the IDP to SP callbacks. Examples are shown below:

IDP Metadata

```
 ?xml version="1.0" encoding="UTF-8"?>
<EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
                  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
                  xmlns:shibmd="urn:mace:shibboleth:metadata:1.0"
                  xmlns:xml="http://www.w3.org/XML/1998/namespace"
```

```
                entityID="https://rbtdev/idp/shibboleth">
    <IDPSSODescriptor errorURL="https://rbtdev/identity/feedback.htm"
         protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
        <Extensions>
            <shibmd:Scope regexp="false">rbtdev</shibmd:Scope>
        </Extensions>
        <KeyDescriptor use="signing">
            <ds:KeyInfo>
                <ds:X509Data>
                    <ds:X509Certificate>{cert info}
                    </ds:X509Certificate>
                </ds:X509Data>
            </ds:KeyInfo>
        </KeyDescriptor>
        <KeyDescriptor use="encryption">
            <ds:KeyInfo>
                <ds:X509Data>
                    <ds:X509Certificate>{cert info}
                    </ds:X509Certificate>
                </ds:X509Data>
            </ds:KeyInfo>
        </KeyDescriptor>
        <ArtifactResolutionService
          Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
          Location="https://rbtdev/idp/profile/SAML2/SOAP/ArtifactResolution"
          index="1"/>
        <SingleLogoutService
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
            Location="https://rbtdev/idp/profile/SAML2/Redirect/SLO"/>
        <SingleLogoutService
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
            Location="https://rbtdev/idp/profile/SAML2/POST/SLO"/>
        <SingleLogoutService
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST-
            SimpleSign"
            Location="https://rbtdev/idp/profile/SAML2/POST-SimpleSign/SLO"/>
        <SingleLogoutService
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
            Location="https://rbtdev/idp/profile/SAML2/SOAP/SLO"/>
        <NameIDFormat>urn:mace:shibboleth:1.0:nameIdentifier</NameIDFormat>
        <NameIDFormat>
            urn:oasis:names:tc:SAML:2.0:nameid-format:transient
        </NameIDFormat>
        <NameIDFormat>
            urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
        </NameIDFormat>
        <SingleSignOnService
            Binding="urn:mace:shibboleth:2.0:profiles:AuthnRequest"
            Location="https://rbtdev/idp/profile/SAML2/Unsolicited/SSO"/>
```

```
        <SingleSignOnService
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
            Location="https://rbtdev/idp/profile/SAML2/POST/SSO"/>
        <SingleSignOnService
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST-
            SimpleSign"
            Location="https://rbtdev/idp/profile/SAML2/POST-SimpleSign/SSO"/>
        <SingleSignOnService
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
            Location="https://rbtdev/idp/profile/SAML2/Redirect/SSO"/>
    </IDPSSODescriptor>
    <AttributeAuthorityDescriptor
        protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
        <Extensions>
            <shibmd:Scope regexp="false">rbtdev</shibmd:Scope>
        </Extensions>
        <KeyDescriptor use="signing">
            <ds:KeyInfo>
                <ds:X509Data>
                    <ds:X509Certificate>{cert info}
                    </ds:X509Certificate>
                </ds:X509Data>
            </ds:KeyInfo>
        </KeyDescriptor>
        <KeyDescriptor use="encryption">
            <ds:KeyInfo>
                <ds:X509Data>
                    <ds:X509Certificate>{cert info}
                    </ds:X509Certificate>
                </ds:X509Data>
            </ds:KeyInfo>
        </KeyDescriptor>
        <AttributeService Binding="urn:oasis:names:tc:SAML:1.0:bindings:SOAP-
            binding"
            Location="https://rbtdev/idp/profile/SAML1/SOAP/AttributeQuery"/>
        <AttributeService Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
            Location="https://rbtdev/idp/profile/SAML2/SOAP/AttributeQuery"/>
        <NameIDFormat>urn:mace:shibboleth:1.0:nameIdentifier</NameIDFormat>
        <NameIDFormat>
            urn:oasis:names:tc:SAML:2.0:nameid-format:transient
        </NameIDFormat>
        <NameIDFormat>
            urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
        </NameIDFormat>
    </AttributeAuthorityDescriptor>
    <Organization>
        <OrganizationName xml:lang="en">MSU</OrganizationName>
        <OrganizationDisplayName xml:lang="en">MSU</OrganizationDisplayName>
        <OrganizationURL xml:lang="en">https://rbtdev</OrganizationURL>
```

```
    </Organization>
</EntityDescriptor>
```

In the XML above rbtdev is the IDP server. The {cert info} sections would contain the certificate. This XML will passed to the SamlClient to initiate the SP to IDP communications.

SP METADATA

SP metadata is used when registering a Service Provider with an Identity Provider and define the callback parameters used during the authentication process. An example xml is shown below:

```
<?xml version="1.0"?>
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
    entityID="https://rbtdev:8443/msusaml">
    <md:SPSSODescriptor AuthnRequestsSigned="false"
        WantAssertionsSigned="false"
        protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
        <md:SingleLogoutService
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
            Location="https://rbtdev:8443/msusaml/redirect" />
        <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-
            format:unspecified</md:NameIDFormat>
        <md:AssertionConsumerService
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
            Location="https://rbtdev:8443/msusaml/auth"  index="1" />
    </md:SPSSODescriptor>
</md:EntityDescriptor>
```

This is a very simple SP metadata definition that does not define any certificates for signing and encryption. These can be added if desired. The areas highlighted in yellow are the SP application defined callback URLs.

JAVA IMPLEMENTATION

The java implementation consists of 4 java files and the web.xml configuration for the web application.

- ApplicationContextListener – creates the SamlClient at context startup and stores the client as a context attribute for use by other classes
- AuthFilter – check to see if user is authenticated, if not, redirects to the IDP for authentication

- AuthServlet – this is the callback defined for the IDP authentication, this gets called if user is authenticated. Any defined attributes can be loaded here and stored in the current session for later use.
- LogoutServet – handles logout redirect from IDP.

WEB.XML

The java classes described above are configured here. Context parameters (highlighted in yellow) contain application-specific initialization parameters:

```
<web-app version="4.0" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd">
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>

    <!-- IDP interface information -->
    <context-param>
        <param-name>relaying.party.identifier</param-name>
        <param-value>https://rbtdev:8443/msusaml</param-value>
    </context-param>
    <context-param>
        <param-name>saml.assertion.handler.url</param-name>
        <param-value>https://rbtdev:8443/msusaml/auth</param-value>
    </context-param>

    <listener>
        <listener-class>edu.msu.saml.AppContextListener</listener-class>
    </listener>
    <filter>
        <filter-name>AuthFilter</filter-name>
        <filter-class>edu.msu.saml.AuthFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>AuthFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <servlet>
        <servlet-name>AuthServlet</servlet-name>
        <servlet-class>edu.msu.saml.AuthServlet</servlet-class>
    </servlet>
```

```
    <servlet-mapping>
        <servlet-name>AuthServlet</servlet-name>
        <url-pattern>/auth</url-pattern>
    </servlet-mapping>
    <servlet>
        <servlet-name>LogoutServlet</servlet-name>
        <servlet-class>edu.msu.saml.LogoutServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>LogoutServlet</servlet-name>
        <url-pattern>/logout</url-pattern>
    </servlet-mapping>
</web-app>
```

APPLICATIONCONTEXTLISTENER

```
public class AppContextListener implements ServletContextListener {
    public static final String AUTHENTICATED_SESSION_DATA_KEY =
"authenticated.session.data";

    @Override
    public void contextInitialized(ServletContextEvent e) {
        // create SamlClient and store for later use
        System.out.println("in AppContextListener.contextInitialized()");
        SamlClient samlClient;
        try {
            // create the SamlClient
            samlClient = SamlClient.fromMetadata(
                e.getServletContext()
                  .getInitParameter("relaying.party.identifier"),
                e.getServletContext()
                 .getInitParameter("saml.assertion.handler.url"),
                getIDPMetaData(e.getServletContext()));
            // set the SamlClient a a context attribute for later use
            e.getServletContext().setAttribute("samlclient", samlClient);
            System.out.println("SamlClient created");
        } catch (SamlException ex) {
            throw new RuntimeException(ex.toString(), ex);
        }

    }


    @Override
    public void contextDestroyed(ServletContextEvent e) {
    }

    private Reader getIDPMetaData(ServletContext servletContext) {
        // load the IDP metadata file from idpmetadata.xml on the classpath
```

```
        return new InputStreamReader(servletContext
            .getResourceAsStream("/WEB-INF/classes/idpmetadata.xml"));
    }
}
```

AuthFilter

```
public class AuthFilter implements Filter {

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
    }

    @Override
    public void doFilter(ServletRequest request, ServletResponse response,
        FilterChain chain) throws IOException, ServletException {
        HttpServletRequest httpServletRequest = (HttpServletRequest) request;
        HttpServletResponse httpServletResponse
            = (HttpServletResponse) response;
        System.out.println("in AuthFilter.doFilter() - "
            + httpServletRequest.getRequestURL().toString());

        try {
            // if we have a SamlResponse in the request or authenticated
            // session data process normally, if not redirect to IDP for
            // authentication
            if ((httpServletRequest.getParameter("SAMLResponse") != null)
                || (httpServletRequest.getSession()
                    .getAttribute(
                    AppContextListener.AUTHENTICATED_SESSION_DATA_KEY)
                      != null)) {
               chain.doFilter(request, response);
            } else {
                System.out.println("redirect to IDP");
                // get the SamlClient from the context
                SamlClient samlClient =
                  (SamlClient)httpServletRequest.getServletContext()
                 .getAttribute("samlclient");
                samlClient.redirectToIdentityProvider(
                    httpServletResponse, null);
            }
        } catch (SamlException ex) {
            throw new ServletException(ex);
        }
    }

    @Override
    public void destroy() {
    }
```

```
}
```

AUTHSERVLET

```
@WebServlet(name = "AuthServlet", urlPatterns = {"/auth"})
public class AuthServlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        System.out.println("in AuthServlet.doPost()");
        // get SamlClient from context
        SamlClient samlClient = (SamlClient)
            request.getServletContext().getAttribute("samlclient");
         try {
            // pull SamlResponse object from request
            SamlResponse samlResponse =
              samlClient.decodeAndValidateSamlResponse(
                 request.getParameter("SAMLResponse"));
            System.out.println("samlResponse - " + samlResponse);

            // pull configured attributes from SamlResponse and add to session
            // for later use
            request.getSession().setAttribute(
                AppContextListener.AUTHENTICATED_SESSION_DATA_KEY,
                getAttributes(samlResponse.getAssertion()
                   .getAttributeStatements()));

            // forward on to desired page
            request.getRequestDispatcher("index.html").forward(request,
                response);
        } catch (SamlException ex) {
            throw new ServletException(ex.toString(), ex);
        }
    }


    // this method pulls attributes from SamlResponse and
    // puts them in a HashMap
    private Map<String, String> getAttributes(
        List <AttributeStatement> attributeStatements) {
        Map<String, String> retval = new HashMap<>();

        if (!attributeStatements.isEmpty()) {
            for (Attribute att : attributeStatements.get(0)
           .getAttributes()) {
               retval.put(att.getFriendlyName(),
                  att.getAttributeValues().get(0).getDOM().getTextContent());
            }
```

```
        }

        System.out.println("SAML attributes - " + retval.toString());

        return retval;
    }
}
```

LogoutServlet

```
@WebServlet(name = "LogoutServlet", urlPatterns = {"/logout"})
public class LogoutServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request,
       HttpServletResponse response) throws ServletException, IOException {
       System.out.println("in LogoutServlet.doGet()");

       // remove authentication session data and invalidate session
       request.removeAttribute(
           AppContextListener.AUTHENTICATED_SESSION_DATA_KEY);
       request.getSession().invalidate();
       request.getRequestDispatcher("index.html").forward(request, response);
    }
}
```