The Kiuru JAAS module for Shibboleth 2.0 Whitepaper v0.2

Introduction

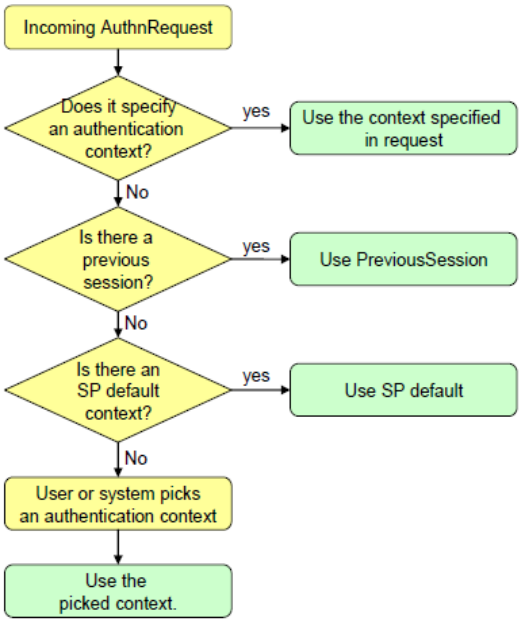
This paper presents a set of tools for integrating ETSI M-COMM[1] standard based mobile signature service provider (MSSP) with a Shibboleth 2 IDP. The tools give you an ability to use MobileTwoFactorContract authentication context without having to write any code of your own.

Out-of-the-box, a Shibboleth 2 IdP supports three SAML2 authentication contexts. The first is transport-protected username and password authentication for end-user interaction. The second is the cookie-based previous session context. The third is "unspecified", which allows the IdP to determine which context to use. "Unspecified" is generally a useful pick for clients in an SSO environment, since it allows the IdP to trigger a login procedure or use an existing session.

This paper introduces a fourth authentication context for MobileTwoFactorContract authentication contect with an MSSP. In specific terms, the authentication contexts are as follows:

```
urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
urn:oasis:names:tc:SAML:2.0:ac:classes:PreviousSession
urn:oasis:names:tc:SAML:2.0:ac:classes:Unspecified
urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorContract
```

The new MobileTwoFactorContract authentication context is an alternative for username and password authentication. When the IdP received a new authentication request from an SP, the IdP must determine which of the two concrete authentication processes it will use. The flowing diagram below shows how the IdP selects the authentication context (and the method) for an incoming authentication request.



The tricky thing is determining whether to use username and password or mobile authentication, when both are possible. This happens when the request does not specify a context and when the SP has no default context. In practice, the Shibboleth 2 IdP engine ends up picking up one effectively at random. This is certainly not desirable. Determining what to do in this situation is outside the scope of this toolkit.

Level of Assurance (LOA) feels like a workable framework for tackling this problem. Danish OIOSAML has more information about LOA in Shibboleth 2 environment. We have used OIOSAML SP module in our implementations.

Operational architecture

The Kiuru MSSP JAAS toolkit includes a JAAS login module and Shibboleth 2 -compatible integration components (login handler, sample MsspLogin.jsp and an authentication servlet).

Components

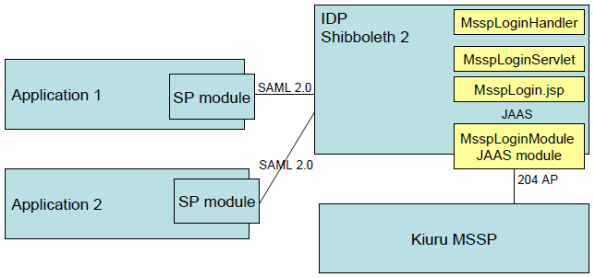


Figure. IDP environment.

The Kiuru MSSP JAAS toolkit includes a JAAS login module and Shibboleth 2 -compatible integration components (login handler, sample MsspLogin.jsp and an authentication servlet).

Component	Purpose	Configuration
MsspLoginHandler	Tell Shibboleth about mobile two factor authentication method and configure it as a Shibboleth LoginHandler.	$IDPHOME/conf/handler.xml
MsspLoginServlet	Parse input from end user, invoke the JAAS login mechanism, forward to Shibboleth 2 authn engine.	Init-params in web.xml
MsspLogin.jsp	Login view presented on the end user's web browser	No configuration as such. Edit the file itself.
MsspLoginModule	Send signature request to MSSP, receive response, populate Subject with Principal and Credential data.	Configure MSSP connection in $IDPHOME/conf/mssplogin.config

[1] ETSI TS 102 204, ETSI TS 102 207, ETSI TR 102 206, ETSI TR 102 203

Methics Ltd, Vuorikatu 16, FIN-00100 Helsinki
Business ID: 1778566-8
www.methics.fi

Installation

Start with a Shibboleth 2 IDP installation. This paper refers to the base directory of this installation as $IDPHOME. There are four primary files to install.

- mssp-shibplugin-3.4.0.jar
- mssp-jaas.jar
- mssplogin.config
- MsspLogin.jsp

Install all other libs in $IDPHOME/webapps/idp/WEB-INF/lib/.

All of the three deployed components require configuration. We'll look at each of them separately.

LoginHandler configuration

LoginHandler configuration is done in $IDPHOME/conf/handler.xml. First, declare the Kiuru Shib-plugin namespace and tell Shibboleth to look for the schema in its classpath.

```
<ProfileHandlerGroup
    xmlns="urn:mace:shibboleth:2.0:idp:profile-handler"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:kiuru="http://kiuru.methics.fi/shibboleth/authn"
    xsi:schemaLocation="urn:mace:shibboleth:2.0:idp:profile-handler
    classpath:/schema/shibboleth-2.0-idp-profile-handler.xsd
    http://kiuru.methics.fi/shibboleth/authn
    classpath:/schema/mssplogin-profile-handler.xsd">
```

Add the red items (Kiuru namespace declaration and schemaLocation) to the ProfileHandlerGroup element. Second, declare the LoginHandler. Add the following to the list of LoginHandlers at the end of the file.

```
<!-- MSSP login handler -->
<LoginHandler xsi:type="kiuru:MsspLogin"
    jaasConfigurationLocation="$IDP_HOME/shibboleth2.0.0/conf/\
    mssplogin.config"
   authenticationServletURL="/Authn/MobileSignature"
>
<AuthenticationMethod>
   urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorContract
</AuthenticationMethod>
</LoginHandler>
```

Write your $IDP_HOME on the jaasConfiguration\ Location URL.

Note the authentication servlet URL. Do not change it unless you have a good reason. The URL must match the URL you assign for the MsspLoginServlet.

Servlet configuration

Servlet configuration is done in $IDPHOME/webapps/idp/WEB-INF/web.xml . Insert the following servlet declaration and mapping.

```
<servlet>
    <servlet-name>MsisdnAuthHandler</servlet-name>
    <servlet-class>  fi.methics.jaasmodule.MsisdnLoginServlet
    </servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>MsisdnAuthHandler</servlet-name>
    <url-pattern>/Authn/MobileSignature</url-pattern>
</servlet-mapping>
```

Note that "/Authn/MobileSignature" is the preset URL that the MSSPLoginHandler component uses.

There are two optional init-parameters. They are the same parameters used by the vanilla Shibboleth UsernamePasswordLoginServlet.

```
<servlet>
<servlet-name>MsisdnAuthHandler</servlet-name>
<servlet-class>fi.methics.shib.MsspLoginServlet</servlet-class>
<!-- Login page url. Optional, default is "MsspLogin.jsp". -->
<!-- JAAS configuration name. Default is "ShibMsspAuth". -->
<!--
   <init-param>
   <param-name>jaasConfigName</param-name>
   <param-value>ShibMsspAuth</param-value>
</init-param>
-->
<!--
   <init-param>
   <param-name>loginPage</param-name>
   <param-value>MsspLogin.jsp</param-value>
</init-param>
-->
</servlet>
```

JSP configuration

The MSSP login JSP page is in MsspLogin.jsp . The important items are the param names.

Param	Purpose
msisdn	Subject's MSISDN (mobile phone number).
ap_transId	ETSI TS 102 204 AP_TransID. The application (IdP) must set this. This is optional. The MsspLoginServlet assigns a value for this if it's not set by the JSP.
sessionId	FiCom session id. According to FiCom rules (FiCom 1.1, section 6.2), the web page must display a "session id" to the end user and the MSSP authentication request must also display this same session id. The MsspLoginServlet sets up a four-digit value for the session id and stores it in the Java Session.
nospam	FiCom nospam code. See FiCom 1.1, section 6.1. This is optional and is off by default.

JAAS module configuration

JAAS module configuration is done in mssplogin.config. MSSP configuration options are defined by your MSS provider.

```
ShibMsspAuth {
 fi.methics.jaasmodule.MsspLoginModule required
 serviceSignatureUrl="http://orion:8080/soap/services/MSS_Signat
 urePort"
 serviceStatusUrl="http://orion:8080/soap/services/MSS_StatusQue
 ryPort"
 serviceReceiptUrl="http://orion:8080/soap/services/MSS_Receipt
 _Port"
 apId="http://ap1.methics.fi"
 apPassword="ap1pass"
 msspIdUri="http://my-mssp.id.invalid"
 signatureProfile="http://mss.ficom.fi/TS102206/v1.0.0/FiCom-
 FINEID-citizencertificate-authentication-profile.xml"
 nospamVerify=”no”;
};
```

To use HTTPS in the MSSP connection, add the following options.

```
https.truststore=”keystore”
https.truststore.password=”tspassword”
https.keystore=”keystore”
https.keystore.password=”kspassword”
https.keystore.type=”JKS”
```

You can use the JAAS flags (required, requisite, optional…) to support both username and mssp authentication in a single file. However, this toolkit includes a separate JAAS config file for MSSP authentication.

Software Distribution

The open source package and documentation distribution will be located here:

http://www.methics.fi/resources.html

or contact our sales (the address below).

You can find also testing MSSP configuration (he1-metadata.xml) and Java demo phone details from the same address. Kiuru JAAS module license is LPGL and Kiuru MSSAPI has a proprietary license.

For more information

See also the Kiuru Shibboteth 2 Installation Instructions for detailed instruction and Kiuru MSSP product factsheet to find out how to build infrastructure for mobile signature services. For more information, call us at

tel. +358 (0)9 5840 0188
e-mail methics.sales@methics.fi

About Methics

Methics Ltd is a privately held ISV, which has developed PKI and wireless PKI products for application providers and operators and built unique security and performance testing tools for operators. The company focuses on security applications and digitizing customer relationship, processes and customer interaction.

Figure. JAAS authentication flow.