2.10 SAML configuration and activation

1. Introduction

Using a Single Sign On service with PrivateServer let the customer use the same username and password across different systems. PrivateServer provides support for SAML 2.0 (Security Assertion Markup Language) is an XML-based open standard data format for exchanging authentication and authorization data between parties, in particular, between an identity provider and a service provider.

SAML protocol identify two different actors:

- **IdP** (Identity Provider) : is responsible for issuing identification information for all providers looking to interact / service with the system in any possible way. In our environment is the customer server that had all the users details and acts as an authenticator
- **SP** (Service Provider): requests and obtains an identity assertion from the identity provider. In our environment PrivateServer acts as a SP, delegating user identification to the customer IDP.

2. Setup

2.1. Pre-requisites

Before we configure PrivateServer , you must be sure you accomplished following tasks:

1. Created your SSO User/s.
2. Retrieved IDP file (which is an xml one)
3. Create on PrivateServer enough users to match the ones in SSO. Please remember such users on PrivateServer must have correct login, along with SSO Identity Provider. In our example login is equal to email field.

2.2. Keystore Generation

⚠ To perform following operations on PrivateServer you must be logged in by SSH console using privileged user root)

Create host directory:

```
mkdir -p /data/privateserver/security
cd /data/privateserver/security
```

Generate a new keystore and a signing key for the SAML request using the following command, considering first that:

- KEY_ALIAS is the signing key alias
- KEYSTORE_NAME is the keystore file name

```
keytool -genkey -alias KEY_ALIAS -keyalg DSA -keystore KEYSTORE_NAME.jks -keysize 1024
```

eg:

```
[root@dev security]# keytool -genkey -alias saml_sso -keyalg DSA -keystore test_saml.jks -keysize 1024
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  Road Runner
What is the name of your organizational unit?
  [Unknown]:  IT department
What is the name of your organization?
  [Unknown]:  Acme
What is the name of your City or Locality?
  [Unknown]:  New York City
What is the name of your State or Province?
  [Unknown]:  New York
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=Road Runner, OU=IT department, O=Acme, L=New York City, ST=New York, C=US correct?
  [no]:  yes
Enter key password for <saml_sso>
        (RETURN if same as keystore password):
Re-enter new password:
```

You're going to be asked about keystore password (KEYSTORE_PASSWORD) and the signing key encryption password (KEY_PASSWORD).

You can now export the certificate in PEM format using

```
keytool -export -keystore KEYSTORE_NAME.jks -alias KEY_ALIAS -rfc -file signing.cer
```

eg:

```
[root@dev security]# keytool -export -keystore test_saml.jks -alias saml_sso -rfc -file signing.cer
Enter keystore password:
Certificate stored in file <signing.cer>
```

Above command outcome will be used to configure sp.xml file, so keep it available for cut'n'paste

2.3. SP XML metadata file

In /data/privateserver/security directory we must also write one file named sp.xml, that is used to configure your SP. Download sample sp.xml

```
<?xml version="1.0" encoding="UTF-8"?>
  <md:EntityDescriptor entityID="server.hostname" xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata">
       <md:SPSSODescriptor AuthnRequestsSigned="true" WantAssertionsSigned="false" protocolSupportEnumeration="
urn:oasis:names:tc:SAML:2.0:protocol">
    <md:Extensions>
      <idpdisco:DiscoveryResponse xmlns:idpdisco="urn:oasis:names:tc:SAML:profiles:SSO:idp-discovery-protocol"
Binding="urn:oasis:names:tc:SAML:profiles:SSO:idp-discovery-protocol" Location="https://SERVER-FQDN/spring-
security-saml/login/auth/alias/localhost?disco=true"/>
    </md:Extensions>
    <md:KeyDescriptor use="signing">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:X509Data>
          <ds:X509Certificate>
MIIC9jCCArSgAwIBAgIETo67pDALBgcqhkjOOAQDBQAwXjELMAkGA1UEBhMCVUsxEDAOBgNVBAgT
B1Vua25vd24xDzANBgNVBAcTBmxvbmRvbjENMAsGA1UEChMEYnVyYjENMAsGA1UECxMEYnVyYjEO
MAwGA1UEAxMFZmVyb3owHhcNMTExMDA3MDg0MzE2WhcNMTIwMTA1MDg0MzE2WjBeMQswCQYDVQQG
EwJVSzEQMA4GA1UECBMHVW5rbm93bjEPMA0GA1UEBxMGbG9uZG9uMQ0wCwYDVQQKEwRidXJiMQ0w
CwYDVQQLEwRidXJiMQ4wDAYDVQQDEwVmZXJvejCCAbgwggEsBgcqhkjOOAQBMIIBHwKBgQD9f1OB
HXUSKVLfSpwu7OTn9hG3UjzvRADDHj+AtlEmaUVdQCJR+1k9jVj6v8X1ujD2y5tVbNeBO4AdNG/y
ZmC3a5lQpaSfn+gEexAiwk+7qdf+t8Yb+DtX58aophUPBPuD9tPFHsMCNVQTWhaRMvZ1864rYdcq
7/IiAxmd0UgBxwIVAJdgUI8VIwvMspK5gqLrhAvwWBz1AoGBAPfhoIXWmz3ey7yrXDa4V7l5lK+7
+jrqgvlXTAs9B4JnUVlXjrrUWU/mcQcQgYC0SRZxI+hMKBYTt88JMozIpuE8FnqLVHyNKOCjrh4r
s6Z1kW6jfwv6ITVi8ftiegEkO8yk8b6oUZCJqIPf4VrlnwaSi2ZegHtVJWQBTDv+z0kqA4GFAAKB
gQDKBDz1DFPPmmWp9n1FskJOev7CnnVFsKji1NLUDdifvS+uW+cnvnDfD3yPdxzUeknCrPTBRp+B
IvYUvLQ57LMIuLgKQ12RujGl0Oz9JbFMAHuBV2I/7ZykzGQPysSEqKCqG+kDc8VZ4AfIf/S8YnQk
xqdWQ5jLTIzXvcWd0WEYbDALBgcqhkjOOAQDBQADLwAwLAIUGP/oZpi79ZM1793XzZvnmrnmz5gC
FBm4bDN8h/0hAa83jaD8joLr098I
                      </ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
    </md:KeyDescriptor>
    <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location="https://SERVER-
FQDN/spring-security-saml/saml/SingleLogout/alias/localhost"/>
    <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" Location="
https://SERVER-FQDN/spring-security-saml/saml/SingleLogout/alias/localhost"/>
    <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP" Location="https://SERVER-FQDN
/spring-security-saml/saml/SingleLogout/alias/localhost"/>
    <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</md:NameIDFormat>
    <md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:transient</md:NameIDFormat>
    <md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:persistent</md:NameIDFormat>
    <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified</md:NameIDFormat>
    <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName</md:NameIDFormat>
    <md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location="
https://SERVER-FQDN/spring-security-saml/saml/SSO/alias/localhost" index="0" isDefault="true"/>
    <md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact" Location="
https://SERVER-FQDN/spring-security-saml/saml/SSO/alias/localhost" index="1" isDefault="false"/>
    <md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:PAOS" Location="https://SERVER-
FQDN/spring-security-saml/saml/SSO/alias/localhost" index="2" isDefault="false"/>
  </md:SPSSODescriptor>
</md:EntityDescriptor>
```

Three changes are required:

1. Change entityID="server.hostname" with a unique UI that identify PrivateServer as SAML service provider. This can easily be PrivateServer FQDN.
2. Cut&Paste the signing certificate (without the -----BEGIN CERTIFICATE----- and -----END CERTIFICATE----- placeholders) in **<ds: X509Certificate>** XML TAG . This is what you created at the end of 4.2 paragraph. To retrieve it:

   ```
   cat signing.cer
   ```

3. Change all "**SERVER-FQDN**" occurrences with your PrivateServer FQDN

> Every change to this file requires PrivateServer Tomcat to be restarted

Once you're done you must be sure a proper service entry is present on your IDP. Eg: Some IDP requires your sp.xml to be uploaded.

2.4. IDP XML metadata file

Retrieve the IDP XML metadata file from the company SAML server and save it in file: /data/privateserver/security/idp.xml .

2.5. Config file

In order to enable SAML edit file /data/privateserver/Config.groovy and append the following lines (those are to be considered one template and so they require proper configuration):

```
grails.plugins.springsecurity.saml.active = true
grails.plugins.springsecurity.saml.userAttributeMappings = ["username":"EmailAddress"]
grails.plugins.springsecurity.saml.metadata.url = "/saml/metadata" // Metadata xml URL
grails.plugins.springsecurity.saml.metadata.defaultIdp = 'ping'
grails.plugins.springsecurity.saml.metadata.providers = [ping: 'file:///data/privateserver/security/idp.xml']
grails.plugins.springsecurity.saml.metadata.sp.file = 'file:///data/privateserver/security/sp.xml'
grails.plugins.springsecurity.saml.metadata.sp.defaults = [
        local: true,
        alias: '<ac:macro ac:name="brand"><ac:parameter ac:name="brand">server</ac:parameter></ac:macro>',
        securityProfile: 'metaiop',
        signingKey: 'KEY_ALIAS',
        encryptionKey: 'KEY_ALIAS',
        tlsKey: 'KEY_ALIAS',
        requireArtifactResolveSigned: false,
        requireLogoutRequestSigned: false,
        requireLogoutResponseSigned: false ]
grails.plugins.springsecurity.saml.keyManager.storeFile = 'file:///data/privateserver/security/KEYSTORE_FILE'
grails.plugins.springsecurity.saml.keyManager.storePass = 'KEYSTORE_PASSWORD'
grails.plugins.springsecurity.saml.keyManager.passwords = [ KEY_ALIAS: 'KEY_PASSWORD' ]
grails.plugins.springsecurity.saml.keyManager.defaultKey = 'KEY_ALIAS'
```

Please check all values are fine according to your needs. For explanation of each config key please read following table:

Config key	Description	Values
`grails.plugins.springsecurity.saml.active`	Enable - Disable SAML integration	true - false
`grails.plugins.springsecurity.saml.userAttributeMappings`	Mapping between IdP identification key and Privateserver account identification key	["username":"XXXXXXX"] where XXXXXXX is the IdP Key. **Just change XXXXXXX with your key.** Check following paragraph 2.6 if in doubt.
`grails.plugins.springsecurity.saml.metadata.url`	SP metadata URL	**don't change unless you know what you're doing**
`grails.plugins.springsecurity.saml.metadata.defaultIdp`	IdP alias	**don't change unless you know what you're doing**
`grails.plugins.springsecurity.saml.metadata.providers`	IdP xml file	[ping :"PATH"] **idp.xml file path**: it must starts with "file://"
`grails.plugins.springsecurity.saml.metadata.sp.file`	SP xml file	**sp.xml file path**: it must starts with "file://"
`grails.plugins.springsecurity.saml.metadata.sp.defaults`	SP configuration parameters	Inside this block of settings you have to: • change the "**alias**" one with "entityID" specified in sp.xml (we suggested to use PrivateServer FQDN) • change all **KEY_ALIAS** instances

`grails.plugins.springsecurity.saml.keyManager.storeFile`	Keystore file	SAML keystore path - it must starts with "file://" (in our example you have to change KEYSTORE_FILE with your actual keystore name)
`grails.plugins.springsecurity.saml.keyManagerstorePass`	Keystore password	SAML keystore password
`grails.plugins.springsecurity.saml.keyManager.passwords`	Key password	[KEY_ALIAS: 'XXXXXXX'] where KEY_ALIAS is the signing key alias and XXXXX is the key password
`grails.plugins.springsecurity.saml.keyManager.defaultKey`	Default SAML key	The KEY_ALIAS created in "Keystore generation" step

Once you went throughout your first-time configuration, please consider that every changes in this file requires PrivateServer Tomcat to be restarted.

Download sample Config.groovy

2.6. User creation

A PrivateServer account must be created for every SSO account, with the right ROLES. Tipically SAML IdP returns an "email" as username, so the mapping between PrivateServer account identifier and SAML account identifier could be (see the example above)

```
grails.plugins.springsecurity.saml.userAttributeMappings = ["username":"EmailAddress"]
```

If SAML IdP use a different key in order to identify an user (for example "LoginName") the configuration must be changed. Once the the configuration is done, PrivateServer administrator can create every new account using the IdP identification key as username, using a fake password and setting the right ROLES to the user.

In your config.groovy you can specify a "grails.plugins.springsecurity.saml.autoCreate.active = true" option that create a PrivateServer local account at every SAML new login. The new account created has no right,it only simplifies the new user creation with the right username given by SAML IDP

That's pretty much it. Once you've finished you must restart tomcat by performing:

```
/data/bin/restart-http.sh
```

then each access to PrivateServer will require SSO authentication and thus will result in redirecting on your IdP login page.

2.7. Automatic account creation

Those lines must be add to Config.groovy to enable auto create account:

```
grails.plugins.springsecurity.saml.autoCreate.active = true
grails.plugins.springsecurity.saml.autoCreate.key = 'username'
```

That's pretty much it. Once you've finished you must restart tomcat by performing:

```
/data/bin/restart-http.sh
```

After completing the configuration process try to login to PrivateServer using SSO account that is not created on server previously, you are able to log on to PrivateServer with no rights or permission and a user is created server side.

Login to Web console with an admin user go to users tab and find the user created and give the right permission and role.

2.8. Troubleshooting

Few words of advice:

- Your best friend is /var/log/tomcat6/Privateserver.log : in that file you can find detailed infos about issues arose
- Always check no empty line heads up at the beginning of any xml file.
- Check that copy'n'paste is respectful of character settings and thus it doesn't introduce weird or alien chars in your configuration files
- Always check double time configuration file syntax.