ISTITUTO NAZIONALE DI FISICA NUCLEARE

INFN-14-01/CCR
15th January 2014

INTEGRAZIONE SINGLE SIGN ON SU JASPERSERVER PER INFN AAI

Francesco Serafini[1], Marco Canaparo[2]

1) *INFN-Amministrazione Centrale Via E. Fermi 40, I-00044 Frascati, Italy*
2) *INFN-CNAF, Viale Berti Pichat 6/2, I-40126 Bologna, Italy*

Abstract

Il presente documento descrive come implementare il Single Sign On tramite Shibboleth con JasperServer 4.5.
Lo scopo di questa attività è stato di integrare nel servizio di Business Intelligence dell'INFN, l'autenticazione gestita con AAI e il Single Sign On in modo tale da rendere questo servizio accessibile con le stesse modalità degli altri offerti da INFN.

Commissione Calcolo Reti

CCR-47/2014/P

*Published by **SIDS–Pubblicazioni***
Laboratori Nazionali di Frascati

Sommario

1 INTRODUZIONE 3

2 PREREQUISITI 3

3 PROCEDURA E CONFIGURAZIONI 3

3.1 Apache 2.2 (httpd) 4

3.2 Shibboleth 4

3.3 Proxy AJP 5

3.4 Tomcat 5

3.5 Configurazione del Single Sign On nel JasperServer 5

3.5.1 File /WEB-INF/applicationContext-security-web.xml 5

3.5.2 File /WEB-INF/applicationContext-security.xml 6

3.5.3 Creare il file /WEB-INF/applicationContext-ruoli.xml 8

4 BIBLIOGRAFIA 10

1 INTRODUZIONE

Il software JasperServer è stato adottato dall'INFN dal 2011 per la gestione del servizio nazionale di Business Intelligence. Allo stato attuale ospita vari report e viste multidimensionali che coinvolgono le seguenti aree:

- Contabilità (raccolta dei dati di bilancio)
- Presenze (percentuali assenze/presenze, report INPS)
- Bilancio sociale (distribuzione del personale dell'ente suddiviso per fasce d'età, qualifica, sede e altro).

JasperServer utilizza il modulo security di Spring Framework (spring-security v.2.0.7) per la gestione dell'autenticazione.

2 PREREQUISITI

Nella macchina in cui si vuole eseguire l'integrazione del Single Sign On con JasperServer, è necessario siano soddisfatti i seguenti prerequisiti:

- Installazione di JasperServer-Pro v.4.5.1
- Shibboleth 2.x
- Apache 2.2
- Tomcat 6.0+
- Proxy AJP (modulo di Apache)
- Libreria SSO dell'INFN (sviluppata da Francesco Serafini e Marco Canaparo)

3 PROCEDURA E CONFIGURAZIONI

Lo scopo di questa procedura è affidare la fase di autenticazione al modulo Shibboleth, e fare in modo che le informazioni di login siano elaborate dal software di JasperServer.

JasperServer permetterà quindi la gestione dell'autorizzazione dell'utente in base ai privilegi che si trovano sul server LDAP dell'INFN.

Per ottenere il risultato desiderato è necessario avere tutti i moduli elencati nella sessione precedente installati correttamente e così configurati:

- Shibboleth configurato correttamente secondo le istruzioni presenti in questo link (*http://wiki.infn.it/cn/ccr/aai/howto/saml-sp* sezione Shibboleth SP)
- Il server Apache in ascolto sulla porta 443 (https)
- Il `proxy ajp` che inoltra le richieste dalla porta 443 alla porta 8009
- Tomcat in ascolto sulla porta 8009, che tramite proxy ajp reindirizza alla porta 8443.

Analizziamo ora la configurazione di ogni singolo elemento.

3.1 Apache 2.2 (httpd)

Installare il modulo Apache.

```
yum install httpd mod_ssl
```

Per redirigere tutte le connessioni da http a https, alla fine del file `/etc/httpd/conf/httpd.conf` abilitare il `RewriteEngine`:

```
RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}
```

Per disabilitare la pagina di Login ufficiale di JasperServer e ridirigere da `/` a `/jasperserver-pro`, alla fine del file `/etc/httpd/conf.d/ssl.conf`, prima della chiusura del tag `</VirtualHost>`, aggiungere:

```
RewriteEngine on
RewriteRule   ^/$  /jasperserver-pro [R]
RewriteRule   ^/jasperserver-pro/login.html.*  /jasperserver-pro [R]
```

3.2 Shibboleth

Nel file di configurazione di Shibboleth è necessario specificare i percorsi che saranno protetti da autenticazione. Quindi tutti i path da `/jasperserver-pro` devono essere protetti, tranne `/jasperserver-pro/services`, che invece mantiene l'autenticazione gestita da software di Jasper.

Il file è `/etc/httpd/conf.d/shib.conf`

```
LoadModule mod_shib /usr/lib64/shibboleth/mod_shib_22.so
<Location /jasperserver-pro>
     AuthType shibboleth
     ShibRequestSetting requireSession true
     Require valid-user
     ShibUseHeaders On
</Location>
```

```
<Location /jasperserver-pro/services>
     AuthType shibboleth
     ShibRequestSetting requireSession false
     Require shibboleth
</Location>
```

3.3 Proxy AJP

Dobbiamo ora configurare il modulo AJP di Apache per ridirigere tutte le richieste ricevute da Apache (su porta 443) al webserver Tomcat.

Creiamo il file `/etc/httpd/conf.d/proxy_ajp.conf` con il seguente contenuto:

```
proxyIOBufferSize 65536
ProxyPass /jasperserver-pro ajp://localhost:8009/jasperserver-pro timeout=600000
```

3.4 Tomcat

Il file `/[TOMCAT_HOME]/conf/server.xml` deve essere modificato abilitando il connector alla porta 8009

```
     <Connector port="8009" protocol="AJP/1.3" redirectPort="8443"
         tomcatAuthentication="false" packetSize="65536" />
```

3.5 Configurazione del Single Sign On nel JasperServer

Copiare la libreria `INFNSSO.jar` nella cartella `WEB-INF/lib` e modificare i due file xml che fanno parte della configurazione del modulo security di Spring Framework.

Creare un file xml che serve per la gestione, attraverso la libreria, dei ruoli e degli attributi (che vengono letti da LDAP una volta eseguita l'autenticazione)

3.5.1 File /WEB-INF/applicationContext-security-web.xml

In questo file sono specificati i nomi delle regole di esecuzione dei filtri in base al path. Occorre aggiungere due regole: il `preautheticationFilter` (si trova nel file `applicationContext-securtity.xml` e verrà mostrato in seguito) e il `logoutFilter`. Il layout del file dovrebbe essere qualcosa tipo quello riportato sotto:

```
/**=httpSessionContextIntegrationFilter,preauthenticationFilter,multipartRequestWra
pperFilter,webAppSecurityFilter,jsCsrfGuardFilter,${bean.loggingFilter},${bean.user
PreferencesFilter},JIAuthenticationSynchronizer,anonymousProcessingFilter,exception
TranslationFilter,filterInvocationInterceptor,switchUserProcessingFilter,iPadSuppor
tFilter,logoutFilter
```

Il logout filter è un bean così composto:

```
<bean id="logoutFilter"
class="org.springframework.security.ui.logout.LogoutFilter">
      <constructor-arg value="https://bi-
```

```
sisinfo.infn.it/Shibboleth.sso/Logout?return=http://bi-
sisinfo.cnaf.infn.it/jasperserver-pro/"/>
      <constructor-arg>
        <list>
          <bean
class="org.springframework.security.ui.logout.SecurityContextLogoutHandler"/>
        </list>
      </constructor-arg>
      <property name="filterProcessesUrl" value="/logout.html"/>
</bean>
```

Anche la direttiva basicProcessingFilter deve essere modificata per fargli mantenere il db di Jasper come sorgente di autenticazione.

```
     <bean id="basicProcessingFilter"
class="org.springframework.security.ui.basicauth.BasicProcessingFilter">
        <property name="authenticationManager">
            <ref bean="authenticationManagerBasic"/>
        </property>
     ...
```

3.5.2 File /WEB-INF/applicationContext-security.xml

Contiene la parte più corposa delle modifiche per il SSO. Raggruppa tutti i bean per la configurazione dei parametri del Database, di LDAP e degli header ritornati da Shibboleth. Definisce inoltre gli `authenticationManager` sia per il SSO e per il `BasicAuthentication`.

```
     <!-- ================= PRE-AUTHENTICATION =================== -->

     <bean id="preAuthenticatedProcessingFilterEntryPoint"
class="org.springframework.security.ui.preauth.PreAuthenticatedProcessingFilterEntr
yPoint" />

     <bean id="preauthenticationFilter"
class="it.infn.preauthplugin.INFNPreauthenticationFilter">
      <property name="principalRequestHeader" value="AJP_eppn" />
      <property name="authenticationManager" ref="authenticationManager" />
     </bean>

     <bean id="preauthAuthProvider"
class="org.springframework.security.providers.preauth.PreAuthenticatedAuthenticatio
nProvider">
      <property name="preAuthenticatedUserDetailsService">
        <bean id="userDetailsServiceWrapper"
class="org.springframework.security.userdetails.UserDetailsByNameServiceWrapper">
          <property name="userDetailsService" ref="ldapUserDetailsService" />
        </bean>
      </property>
     </bean>

     <bean id="ldapUserDetailsService"
class="org.springframework.security.userdetails.ldap.LdapUserDetailsService">
      <constructor-arg index="0" ref="userSearch" />
      <constructor-arg index="1" ref="infnAuthoritiesPopulator" />
     </bean>
```

```
     <bean id="userSearch"
class="org.springframework.security.ldap.search.FilterBasedLdapUserSearch">
       <constructor-arg index="0" value="ou=People" />
       <constructor-arg index="1" value="(uid={0})" />
       <constructor-arg index="2" ref="ldapContextSource" />
       <property name="searchSubtree" value="true" />
     </bean>

     <bean id="infnAuthoritiesPopulator"
class="it.infn.preauthplugin.INFNLdapAuthoritiesPopulator">
       <constructor-arg index="0" ref="ldapContextSource" />
       <constructor-arg index="1" value="ou=People" />
       <property name="groupRoleAttribute" value="isMemberOf" />
       <property name="groupSearchFilter" value="uid={1}" />
       <property name="searchSubtree" value="false" />
       <property name="attributeManager"><bean
class="it.infn.preauthplugin.INFNProfileAttributeManager"><property name="db"
ref="infnDBConnection" /></bean></property>
       <property name="folderManager"><bean
class="it.infn.preauthplugin.INFNFolderManager"><property name="db"
ref="infnDBConnection" /></bean></property>
       <property name="userCreation"><bean
class="it.infn.preauthplugin.INFNUserCreation"><property name="db"
ref="infnDBConnection" /></bean></property>
       <property name="rolesContainer" ref="rolesContainer" />
     </bean>

     <bean id="infnDBConnection" class="it.infn.preauthplugin.INFNDBConnection">
       <constructor-arg index="0" value="jdbc:mysql://localhost:3306/jasperserver"
/>
       <constructor-arg index="1" value="jasperdb" />
       <constructor-arg index="2" value="password" />
       <property name="driver" value="com.mysql.jdbc.Driver" />
     </bean>

     <bean id="ldapContextSource"
class="org.springframework.security.ldap.DefaultSpringSecurityContextSource">

       <constructor-arg value="ldaps://example.com/dc=example,dc=com" />
       <property name="userDn" value="cn=jasper,ou=Services,dc=example,dc=com" />
       <property name="password" value="XXXXXXXXX" />

     </bean>



     <!-- ==================== AUTHENTICATION ==================== -->
     <bean id="authenticationManager"
class="org.springframework.security.providers.ProviderManager">
         <property name="providers">
             <list>
               <ref bean="preauthAuthProvider"/>
                 <!-- not on by default <ref local="ldapAuthenticationProvider"/>
                 <ref bean="${bean.daoAuthenticationProvider}"/>
                 <ref bean="anonymousAuthenticationProvider"/> -->
                 <!--ref local="jaasAuthenticationProvider"/-->
             </list>
         </property>
```

```
    </bean>

    <bean id="authenticationManagerBasic"
class="org.springframework.security.providers.ProviderManager">
        <property name="providers">
            <list>
              <ref bean="${bean.daoAuthenticationProvider}"/>
              <ref bean="anonymousAuthenticationProvider"/>
            </list>
        </property>
    </bean>
```

3.5.3 Creare il file /WEB-INF/applicationContext-ruoli.xml

Questo file contiene tutte le regole di creazione di ruoli e attributi all'interno del DB di Jasper a partire dagli attributi che l'utente loggato ha su LDAP.

```
<bean id="rolesContainer" class="it.infn.preauthplugin.INFNRolesContainer">
 <constructor-arg>
   <list>
     <ref bean="admin" />
     <ref bean="responsabileLocale" />
     ...
   </list>
 </constructor-arg>
</bean>

<bean id="admin" class="it.infn.preauthplugin.INFNRoleAttribute">
 <property name="pattern" value="app:jasper::admin:admin" />
 <property name="attributo" value="" />
 <property name="ruolo" value="ADMINISTRATOR|SUPERUSER" />
 <property name="tenantId" value="1" />
</bean>
<bean id="responsabileLocale" class="it.infn.preauthplugin.INFNRoleAttribute">
 <property name="pattern" value="i:infn:([^:]+):csn.:([^:]+)::resp:locale" />
 <property name="attributo" value="{1}-{2}" />
 <property name="ruolo" value="RESP_LOCALE_ESP" />
</bean>
...
```

E' possibile eseguire delle regexp sul contenuto del parametro LDAP e creare ruoli e attributi dinamicamente.

Nel caso del `RESP_LOCALE_ESP` qui sopra vediamo che l'attributo avrà come valore il nome della struttura INFN e il nome dell'esperimento.

es: lnf-atlas

Ad esempio, un utente potrebbe avere come ruoli

Determinato dalla seguente regexp

```
    <bean id="admin" class="it.infn.preauthplugin.INFNRoleAttribute">
        <property name="pattern" value="app:jasper::admin:admin" />
        <property name="attributo" value="" />
        <property name="ruolo" value="ADMINISTRATOR|SUPERUSER" />
        <property name="tenantId" value="1" />
    </bean>
    <bean id="responsabileLocale" class="it.infn.preauthplugin.INFNRoleAttribute">
        <property name="pattern" value="i:infn:([^:]+):csn.:([^:]+)::resp:locale"
/>
        <property name="attributo" value="{1}-{2}" />
        <property name="ruolo" value="RESP_LOCALE_ESP" />
    </bean>
      <bean id="coordinatoreLocale"
class="it.infn.preauthplugin.INFNRoleAttribute">
        <property name="pattern" value="i:infn:([^:]+):(csn[1-7])::n:coordinatore"
/>
        <property name="attributo" value="{1}" />
        <property name="ruolo" value="COORDINATORE_LINEA_SCIENTIFICA_{2}" />
      </bean>
```

Gli attributi assegnati all'utente sono:

- [RESP_LOCALE_ESP:roma1-windows]
- [RESP_NAZIONALE_ESP:calcolo]
- [COORDINATORE_LINEA_SCIENTIFICA_csn7:RM1]

4 BIBLIOGRAFIA

[1] http://docs.spring.io/spring-security/site/docs/2.0.7.RELEASE/reference/preauth.html
[2] https://www.icts.uiowa.edu/confluence/display/ICTSit/Shibboleth+Spring-Security
[3] http://pulkitsinghal.blogspot.it/2010/06/sso-pre-authentication-spring-security.html