Installation du serveur CAS

Adaptation d'un serveur CAS existant

Cette page décrit l'installation de A à Z d'un nouveau serveur CAS qui permet :

- l'authentification transparente des utilisateurs par la transmission de tickets Kerberos
- l'alimentation automatique d'un royaume Kerberos à partir des utilisateurs LDAP lors de leur connexion

Il est facile pour un administrateur CAs d'extraire de cette documentation les éléments nécessaires à l'adaptation d'un serveur CAS existant pour lui donner les fonctionnalités voulues.

Nom du serveur	**cas-kerb.univ-rennes1.fr**
Système	RedHat Entreprise 5
Ouverture de ports	**ssh** (22 tcp) **https** (443)

Installation système

Installer les packages Tomcat (**yum install tomcat5**), Apache (**yum install httpd**), puis Maven :

```
[root@cas-kerb ~]# cd /usr/local
[root@cas-kerb local]# wget ftp://ftp.inria.fr/pub/Apache/maven/binaries/apache-maven-2.2.1-bin.tar.gz
--2010-03-09 10:35:41--  ftp://ftp.inria.fr/pub/Apache/maven/binaries/apache-maven-2.2.1-bin.tar.gz
           => `apache-maven-2.2.1-bin.tar.gz'
Resolving ftp.inria.fr... 192.93.2.32
Connecting to ftp.inria.fr|192.93.2.32|:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.    ==> PWD ... done.
==> TYPE I ... done.  ==> CWD /pub/Apache/maven/binaries ... done.
==> SIZE apache-maven-2.2.1-bin.tar.gz ... 2840961
==> PASV ... done.    ==> RETR apache-maven-2.2.1-bin.tar.gz ... done.
Length: 2840961 (2.7M)
100%[================================>] 2,840,961   2.01M/s   in 1.3s
2010-03-09 10:35:43 (2.01 MB/s) - `apache-maven-2.2.1-bin.tar.gz' saved [2840961]
[root@cas-kerb local]# tar xf apache-maven-2.2.1-bin.tar.gz
[root@cas-kerb local]# ln -s apache-maven-2.2.1 maven2
[root@cas-kerb local]# cd /etc/profile.d
[root@cas-kerb profile.d]# cat > maven2.sh
export PATH=$PATH:/usr/local/maven2/bin
[root@cas-kerb profile.d]# chmod 644 maven2.sh
[root@cas-kerb profile.d]# . maven2.sh
[root@cas-kerb profile.d]#
```

Installation de CAS

Télécharger la dernière version de CAS depuis http://www.jasig.org/cas/download et décompresser :

```
[root@cas-kerb ~]# cd /usr/local
[root@cas-kerb local]# wget http://www.ja-sig.org/downloads/cas/cas-server-3.3.5-release.tar.gz
--2010-03-09 09:25:51--  http://www.ja-sig.org/downloads/cas/cas-server-3.3.5-release.tar.gz
Resolving www.ja-sig.org... 128.112.131.108
Connecting to www.ja-sig.org|128.112.131.108|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 14467126 (14M) [application/x-gzip]
Saving to: `cas-server-3.3.5-release.tar.gz'
100%[========================>] 14,467,126   427K/s   in 15s
2010-03-09 09:26:07 (919 KB/s) - `cas-server-3.3.5-release.tar.gz' saved [14467126/14467126]
[root@cas-kerb local]# tar xf cas-server-3.3.5-release.tar.gz
[root@cas-kerb local]# cd cas-server-3.3.5
[root@cas-kerb cas-server-3.3.5]#
```

Pour diminuer les temps de compilation, on ajoute la propriété **skipTests** au *plugin* **maven-sunfire** dans le fichier **pom.xml** à la racine du projet :

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-surefire-plugin</artifactId>
  <configuration>
    <skipTests>true</skipTests>
    <includes>
      <include>**/*Tests.java</include>
    </includes>
    <excludes>
      <exclude>**/Abstract*.java</exclude>
    </excludes>
  </configuration>
</plugin>
```

Créer un répertoire **cas-server-rennes1** dans lequel seront stockées toutes les personnalisations et y ajouter le fichier **pom.xml** suivant :

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <parent>
    <groupId>org.jasig.cas</groupId>
    <artifactId>cas-server</artifactId>
    <version>3.3.5</version>
  </parent>
  <modelVersion>4.0.0</modelVersion>
    <groupId>fr.univrennes1.cas</groupId>
    <artifactId>cas-server-rennes1</artifactId>
    <version>3.3.5</version>
    <packaging>war</packaging>
    <name>University of Rennes 1 CAS webapp</name>
    <organization>
        <name>University of Rennes 1</name>
        <url>http://www.univ-rennes1.fr.fr</url>
    </organization>
    <description>The University of Rennes 1 customizations to the JA-SIG CAS server.</description>
    <dependencies>
        <dependency>
            <groupId>org.jasig.cas</groupId>
            <artifactId>cas-server-webapp</artifactId>
            <version>${project.version}</version>
            <type>war</type>
        </dependency>
        <dependency>
            <groupId>org.jasig.cas</groupId>
            <artifactId>cas-server-core</artifactId>
            <version>${project.version}</version>
        </dependency>
      </dependencies>
    <build>
        <finalName>cas</finalName>
        <plugins>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>RELEASE</version>
                <configuration>
                    <source>1.5</source>
                    <target>1.5</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
    <repositories>
        <repository>
            <id>jasig-repository</id>
            <name>JA-SIG Maven2 Repository</name>
            <url>http://developer.ja-sig.org/maven2</url>
        </repository>
    </repositories>
    <reporting>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-project-info-reports-plugin</artifactId>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-javadoc-plugin</artifactId>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-changelog-plugin</artifactId>
            </plugin>
        </plugins>
    </reporting>
</project>
```

Plutôt que modifier les fichiers de la distribution CAS, il est préférable de maintenir toutes modifications effectuées dans ce répertoire, ce qui facilite les mises à jour. Tous les fichiers trouvés dans ce répertoire écraseront les fichiers de la distribution, on adoptera donc exactement la même hiérarchie de fichiers que celle de la distribution.

Installation basique

Copier le fichier **src/main/webapp/WEB-INF/classes/log4j.properties** de **cas-server-webapp** dans **src/main/webapp/WEB-INF/classes** et indiquer le chemin des logs :

```
log4j.appender.logfile.File=/var/log/tomcat5/cas.log
```

Générer le WAR, le copier dans Tomcat et redémarrer :

```
[root@cas cas-server-rennes1]#  mvn package install
[root@cas cas-server-rennes1]#  cp target/cas.war /var/lib/tomcat5/webapps/ROOT.war
[root@cas cas-server-rennes1]#  /etc/init.d/tomcat5 restart
```

Debug

Ajouter dans le fichier **src/main/webapp/WEB-INF/classes/log4j.properties** la ligne suivante :

```
log4j.logger.org.jasig.cas=DEBUG
```

Les logs se trouvent dans le répertoire **/var/log/tomcat5**.

La mise au point la plus difficile est celle de Kerberos (les logs en debug de **Krb5LoginModule** se trouvent dans **catalina.out**).

Test

Tester http://cas-kerb.univ-rennes1.fr:8080 (user = test, password = test).

Script de déploiement de CAS

Pour faciliter le déploiement du serveur CAS et le redémarrage de Tomcat, on pourra ajouter le script **/usr/local/cas-server-3.3.5/deploy-restart.sh** suivant :

```
[root@cas-kerb ~]# cd /usr/local/cas-server-3.3.5
[root@cas-kerb cas-server-3.3.5]# cat > deploy-restart.sh
#!/bin/bash
service tomcat5 stop
pushd /var/lib/tomcat5/webapps
rm -rf ROOT ROOT.war
popd
rm -rf /usr/share/tomcat5/work/_
pushd /var/log/tomcat5
rm -rf cas.log catalina.out
touch cas.log catalina.out
chown tomcat.tomcat cas.log catalina.out
popd
pushd /usr/local/cas-server-3.3.5
pushd cas-server-rennes1
mvn package install
cp target/cas.war /var/lib/tomcat5/webapps/ROOT.war
popd
popd
service tomcat5 start
[root@cas-kerb cas-server-3.3.5]# chmod 744 deploy-restart.sh
[root@cas-kerb cas-server-3.3.5]#
```

Ajout d'un frontal Apache

On va dans cette partie configurer un frontal Apache sur le port 80, qui va accéder au Tomcat du serveur CAS en AJP sur le port 8009.

Il n'est pas obligatoire de mettre un frontal Apache devant Tomcat, mais cela délègue le chiffrement à Apache au lieu de Tomcat et simplifie l'administration système (cette architecture est employée de manière générale sur les plateformes d'exploitation).

Configuration de Apache

Insérer dans **/etc/httpd/conf.d/proxy_ajp.conf** les lignes suivantes :

```
ProxyPass / ajp://cas-kerb.univ-rennes1.fr:8009/ min=0 max=100 smax=50 ttl=10
```

Configuration de Tomcat

S'assurer que le connecteur AJP sur le port 8009 n'est pas commenté et a bien le paramètres **tomcatAuthentication** positionné à **false** :

```
<Connector port="8009"
    debug="0"
    enableLookups="false"
    redirectPort="8443"
    protocol="AJP/1.3"
    tomcatAuthentication="false" />
```

Test

Après redémarrage de Apache et Tomcat, le serveur CAS doit désormais répondre sur l'URL https://cas-kerb.univ-rennes1.fr (sur le port 80 par défaut en HTTP).

Passage en HTTPS

Installer si nécessaire le package **mod_ssl** (**yum install mod_ssl**).

Installerle certificat x509 et la cléprivée du serveur dans les répertoires appropriés (/etc/pki/tls/

Configuration de Apache

Installer le certificat du serveur en éditant **/etc/httpd/conf.d/ssl.conf** et modifier les lignes suivantes dans le *virtual host* **_default_:443** :

```
#SSLCertificateFile /etc/pki/tls/certs/localhost.crt
SSLCertificateFile /etc/pki/tls/certs/cas-kerb.univ-rennes1.fr.pem
#SSLCertificateKeyFile /etc/pki/tls/private/localhost.key
SSLCertificateKeyFile /etc/pki/tls/private/cas-kerb.univ-rennes1.fr.key
```

Test

Après redémarrage de Apache, le serveur CAS doit désormais répondre sur l'URL https://cas.ifsic.univ-rennes1.fr (sur le port 443 par défaut en HTTPS).

Suppression de l'accès en HTTP

Ne pas oublier de supprimer la ligne suivante de **/etc/httpd/conf/httpd.conf** :

```
Listen 80
```

Ajout de l'authentification LDAP

Configuration de CAS pour LDAP

Ajouter la dépendance vers le module **cas-server-support-ldap** dans le fichier **pom.xml** :

```
<dependency>
  <groupId>org.jasig.cas</groupId>
  <artifactId>cas-server-support-ldap</artifactId>
  <version>${project.version}</version>
</dependency>
```

Copier le fichier **src/main/webapp/WEB-INF/deployerConfigContext.xml** de **cas-server-webapp** dans **src/main/webapp/WEB-INF**, ajouter le bean suivant pour déclarer le contexte LDAP :

```
<bean id="contextSource" class="org.springframework.ldap.core.support.LdapContextSource">
  <property name="pooled" value="true"/>
  <property name="urls">
    <list>
      <value>ldap://ldapglobal.univ-rennes1.fr/</value>
    </list>
  </property>
  <property name="userDn" value=""/>
  <property name="password" value=""/>
  <property name="baseEnvironmentProperties">
    <map>
      <entry>
       <key>
         <value>java.naming.security.authentication</value>
       </key>
       <value>simple</value>
      </entry>
     </map>
  </property>
</bean>
```

puis changer le handler **SimpleTestUsernamePasswordAuthenticationHandler** par celui-ci :

```
<bean
   class="org.jasig.cas.adaptors.ldap.FastBindLdapAuthenticationHandler" >
     <property name="filter" value="uid=%u,ou=people,dc=univ-rennes1,dc=fr" />
     <property name="contextSource" ref="contextSource" />
</bean>
```

Test

Redéployer le serveur CAS, et tester l'authentification d'un utilisateur LDAP.

Ajout de l'authentification Kerberos

La documentation de référence est http://www.ja-sig.org/wiki/display/CASUM/SPNEGO .

Configuration de Kerberos

Editer le fichier **/etc/krb5.conf** pour intégrer le serveur au domaine Kerberos :

```
[logging]
 default = FILE:/var/log/krb5libs.log
 kdc = FILE:/var/log/krb5kdc.log
 admin_server = FILE:/var/log/kadmind.log

[libdefaults]
 default_realm = UNIV-RENNES1.FR
 dns_lookup_realm = false
 dns_lookup_kdc = false
 ticket_lifetime = 24h
 forwardable = yes

[realms]
 UNIV-RENNES1.FR = {
  kdc = kerb1.univ-rennes1.fr:88
  admin_server = kerb1.univ-rennes1.fr:749
  default_domain = univ-rennes1.fr
 }

[domain_realm]
 .univ-rennes1.fr = UNIV-RENNES1.FR
 univ-rennes1.fr = UNIV-RENNES1.FR

[appdefaults]
 pam = {
   debug = false
   ticket_lifetime = 36000
   renew_lifetime = 36000
   forwardable = true
   krb4_convert = false
 }
```

Ajouter le *principal* **HTTP/cas-kerb.univ-rennes1.fr** (casse importante) au royaume et l'exporter dans le fichier **/etc/http.keytab** :

```
[root@cas-kerb ~]# kadmin
Authenticating as principal rootifsic/admin@UNIV-RENNES1.FR with password.
kadmin: Client not found in Kerberos database while initializing kadmin interface
[rootifsic@cas-kerb ~]# kadmin -p root/admin
Authenticating as principal root/admin with password.
Password for root/admin@UNIV-RENNES1.FR:
kadmin:  addprinc -randkey HTTP/cas-kerb.univ-rennes1.fr
WARNING: no policy specified for HTTP/cas-kerb.univ-rennes1.fr@UNIV-RENNES1.FR; defaulting to no policy
Principal "HTTP/cas-kerb.univ-rennes1.fr@UNIV-RENNES1.FR" created.
kadmin:  ktadd -k /etc/http.keytab HTTP/cas-kerb.univ-rennes1.fr
Entry for principal HTTP/cas-kerb.univ-rennes1.fr with kvno 3, encryption type Triple DES cbc mode with HMAC
/sha1 added to keytab WRFILE:/etc/http.keytab.
Entry for principal HTTP/cas-kerb.univ-rennes1.fr with kvno 3, encryption type ArcFour with HMAC/md5 added to
keytab WRFILE:/etc/http.keytab.
Entry for principal HTTP/cas-kerb.univ-rennes1.fr with kvno 3, encryption type DES with HMAC/sha1 added to
keytab WRFILE:/etc/http.keytab.
Entry for principal HTTP/cas-kerb.univ-rennes1.fr with kvno 3, encryption type DES cbc mode with RSA-MD5 added
to keytab WRFILE:/etc/http.keytab.
kadmin:  exit
[root@cas-kerb ~]#
```

Le fichier **/etc/http.keytab** sera utilisé par la librairie JCIFS, il doit être lisible par l'utilisateur **tomcat** :

```
[root@cas-kerb ~]# cd /etc
[root@cas-kerb etc]# chown root:tomcat http.keytab
[root@cas-kerb etc]# chmod 640 http.keytab
[root@cas-kerb etc]#
```

Debug

Ajouter dans le fichier **/etc/tomcat5/tomcat5.conf** les lignes suivantes :

```
JAVA_OPTS="$JAVA_OPTS -Dcom.ibm.security.jgss.debug=all"
JAVA_OPTS="$JAVA_OPTS -Dsun.security.jgss.debug=true"
JAVA_OPTS="$JAVA_OPTS -Dsun.security.krb5.debug=true"
JAVA_OPTS="$JAVA_OPTS -Djava.security.debug=logincontext,policy,scl,gsslogginconfig
```

Les logs se trouvent dans **/var/log/tomcat5/catalina.out**.

Configuration de CAS

Ajouter le support du *handler* spnego

Editer le fichier **pom.xml** et ajouter la dépendance suivante (par exemple juste après la dépendance vers le module **cas-server-support-ldap**) :

```
<dependency>
     <groupId>org.jasig.cas</groupId>
     <artifactId>cas-server-support-spnego</artifactId>
     <version>${project.version}</version>
</dependency>
```

Modifier le login webflow

Editer le fichier **src/main/webapp/WEB-INF/login-webflow.xml** et ajouter l'état suivant juste avant l'état *viewLoginForm* :

```
<action-state id="startAuthenticate">
  <action bean="negociateSpnego" />
  <transition on="success" to="spnego" />
</action-state>

<action-state id="spnego">
  <action bean="spnego" />
  <transition on="success" to="sendTicketGrantingTicket" />
  <transition on="error" to="viewLoginForm" />
</action-state>
```

Dans ce même fichier, remplacer les références à *viewLoginForm* par *startAuthenticate* pour les deux *decision-state gatewayRequestCheck* et *renewRequ estCheck* :

```
<decision-state id="gatewayRequestCheck">
  <if
    test="${externalContext.requestParameterMap['gateway'] != '' &amp;&amp; externalContext.requestParameterMap
['gateway'] != null &amp;&amp; flowScope.service != null}"
    then="redirect"
    else="startAuthenticate" />
</decision-state>
```

```
<decision-state id="renewRequestCheck">
  <if
    test="${externalContext.requestParameterMap['renew'] != '' &amp;&amp; externalContext.requestParameterMap
['renew'] != null}"
    then="startAuthenticate"
    else="generateServiceTicket" />
</decision-state>
```

Déclarer le bean implémentant le nouvel état du webflow en ajoutant les lignes suivantes dans le fichier **src/main/webapp/WEB-INF/cas-servlet.xml** (par exemple juste avant le bean **authenticationViaFormAction**) :

```
<bean
    id="negociateSpnego"
    class="org.jasig.cas.support.spnego.web.flow.SpnegoNegociateCredentialsAction" />

<bean
    id="spnego"
    class="org.jasig.cas.support.spnego.web.flow.SpnegoCredentialsAction">
        <property name="centralAuthenticationService" ref="centralAuthenticationService"/>
</bean>
```

Modifier le schéma d'authentification

Pour modifier le schéma d'authentification, éditer le fichier **src/main/webapp/WEB-INF/deployerConfigContext.xml** et modifier le bean **authenticationManager** en ajoutant :

- **PrincipalBearingCredentialsToPrincipalResolver** après les *resolvers* existants de **credentialsToPrincipalResolvers**
- **PrincipalBearingCredentialsAuthenticationHandler** avant les *handlers* existants de **authenticationHandlers**

```
<bean id="authenticationManager" class="org.jasig.cas.authentication.AuthenticationManagerImpl">
  <property name="credentialsToPrincipalResolvers">
    <list>
      <!--  ... the others credentialsToPrincipalResolvers ... -->
      <bean class="org.jasig.cas.support.spnego.authentication.principal.SpnegoCredentialsToPrincipalResolver"
/>
    </list>
  </property>
  <property name="authenticationHandlers">
    <list>
      <bean class="org.jasig.cas.support.spnego.authentication.handler.support.JCIFSSpnegoAuthenticationHandler"
>
        <property name="authentication">
          <bean class="jcifs.spnego.Authentication" />
        </property>
        <property name="principalWithDomainName" value="false" />
        <property name="NTLMallowed" value="false"/>
      </bean>
      <!--  ... the others authenticationHandlers... -->
    </list>
  </property>
</bean>
```

Le bean **authenticationManager** doit ainsi ressembler à :

```
<bean   id="authenticationManager"
        class="org.jasig.cas.authentication.AuthenticationManagerImpl">
    <property name="credentialsToPrincipalResolvers">
      <list>
        <bean class="org.jasig.cas.authentication.principal.UsernamePasswordCredentialsToPrincipalResolver" />
        <bean class="org.jasig.cas.authentication.principal.HttpBasedServiceCredentialsToPrincipalResolver" />
        <bean class="org.jasig.cas.support.spnego.authentication.principal.
SpnegoCredentialsToPrincipalResolver" />
      </list>
    </property>
    <property name="authenticationHandlers">
      <list>
        <bean class="org.jasig.cas.support.spnego.authentication.handler.support.
JCIFSSpnegoAuthenticationHandler">
          <property name="authentication">
            <bean class="jcifs.spnego.Authentication" />
          </property>
          <property name="principalWithDomainName" value="true" />
          <property name="NTLMallowed" value="false"/>
      </bean>
      <bean class="org.jasig.cas.authentication.handler.support.
HttpBasedServiceCredentialsAuthenticationHandler"
              p:httpClient-ref="httpClient" />
        <bean class="org.jasig.cas.adaptors.ldap.FastBindLdapAuthenticationHandler" >
          <property name="filter" value="uid=%u,ou=people,dc=univ-rennes1,dc=fr" />
          <property name="contextSource" ref="contextSource" />
        </bean>
      </list>
    </property>
</bean>
```

Ajouter enfin le bean **jcifsConfig**, qui done les options de configuration de JCIFS :

```
<bean name="jcifsConfig" class="org.jasig.cas.support.spnego.authentication.handler.support.JCIFSConfig">
  <property
      name="jcifsServicePrincipal"
      value="HTTP/cas-kerb.univ-rennes1.fr" />
  <property
      name="kerberosDebug"
      value="true" />
  <property
      name="kerberosRealm"
      value="UNIV-RENNES1.FR" />
  <property
      name="kerberosKdc"
      value="kerb1.univ-rennes1.fr" />
  <property
      name="loginConf"
      value="/etc/jcifs/login.conf" />
</bean>
```

Configuration de JCIFS

La configuration de JCIFS se fait également dans le fichier login.conf pointé par le bean **jcifsConfig**. Créer ce fichier avec le contenu suivant :

```
com.sun.security.jgss.krb5.accept {
  com.sun.security.auth.module.Krb5LoginModule
  required
  debug=true
  storeKey=true
  useKeyTab=true
  keyTab="/etc/http.keytab"
  principal="HTTP/cas-kerb.univ-rennes1.fr";
};
```

Configuration de Tomcat

Il faut passer à la JVM qui exécute Tomcat l'option **-Djavax.security.auth.useSubjectCredsOnly=false**, par exemple en éditant le fichier **/etc/tomcat5/tomcat5.conf** et en ajoutant la ligne suivante :

```
JAVA_OPTS="$JAVA_OPTS -Djavax.security.auth.useSubjectCredsOnly=false"
```

Test

Un navigateur bien configuré et possédant des credentials Kerberos valides doit maintenant se connecter au serveur CAS sans aucune interaction....

Ajout de l'alimentation Kerberos

L'alimentation Kerberos désigne le processus qui permet d'alimenter un royaume Kerberos avec comptes utilisateurs lors de la connexion au serveur CAS. Elle peut être utilisée pour amorcer le remplissage du royaume à partir d'un annuaire LDAP, ce que nous montrons dans cet exemple.

Configuration de Kerberos

En premier lieu, déclarer le *principal* qui servira à la mise à jour du royaume Kerberos (ici **cas/admin**) et l'exporter dans le fichier /etc/cas-admin.keytab :

```
[root@cas-kerb ~]# kadmin -p root/admin
Authenticating as principal root/admin with password.
Password for root/admin@UNIV-RENNES1.FR:
kadmin:  addprinc -randkey cas/admin
WARNING: no policy specified for cas/admin@UNIV-RENNES1.FR; defaulting to no policy
Principal "cas/admin@UNIV-RENNES1.FR" created.
kadmin:  ktadd -k /etc/cas-admin.keytab cas/admin
Entry for principal cas/admin with kvno 3, encryption type Triple DES cbc mode with HMAC/sha1 added to keytab
WRFILE:/etc/cas-admin.keytab.
Entry for principal cas/admin with kvno 3, encryption type ArcFour with HMAC/md5 added to keytab WRFILE:/etc
/cas-admin.keytab.
Entry for principal cas/admin with kvno 3, encryption type DES with HMAC/sha1 added to keytab WRFILE:/etc/cas-
admin.keytab.
Entry for principal cas/admin with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc
/cas-admin.keytab.
kadmin:  exit
[root@cas-kerb ~]#
```

Le fichier **cas-admin.keytab** doit être lisible par l'utilisateur **tomcat** :

```
[root@cas-kerb ~]# cd /etc
[root@cas-kerb etc]# chown root:tomcat cas-admin.keytab
[root@cas-kerb etc]# chmod 640 cas-admin.keytab
[root@cas-kerb etc]#
```

Il est également nécessaire de modifier les permissions du fichier de *log* de **kadmin** sans quoi l'appel de **kadmin** par l'utilisateur **tomcat** provoquerait une erreur.

```
[root@cas-kerb ~]# cd /var/log
[root@cas-kerb log]# touch kadmind.log
[root@cas-kerb log]# chown root:tomcat kadmind.log
[root@cas-kerb log]# chmod 664 kadmind.log
[root@cas-kerb log]#
```

Intégration du module cas-server-integration-kerberosfeed

L'alimentation du royaume Kerberos se fait en utilisant le module **cas-server-integration-kerberosfeed** (téléchargeable ici), qui doit être installé au même niveau que les autres modules du projet puis compilé (lancer la commande **mvn package install** depuis le répertoire du module).

Il faut ensuite ajouter la dépendance de **cas-server-rennes1** vers **cas-server-integration-kerberosfeed** en ajoutant dans le fichier **pom.xml** du module **cas-server-rennes1** les lignes suivantes :

```
<dependency>
  <groupId>org.jasig.cas</groupId>
  <artifactId>cas-server-integration-kerberosfeed</artifactId>
  <version>${project.version}</version>
</dependency>
```

Configuration de CAS

On utilise la classe **KerberosFeedAuthenticationHandlerWrapper** pour enrober l'appel des *authenticationHandlers* pour lesquels on souhaite mettre en place une alimentation du royaume Kerberos. Les lignes suivantes de **src/main/webapp/WEB-INF/deployerConfigContext.xml**

```
<bean class="org.jasig.cas.adaptors.ldap.FastBindLdapAuthenticationHandler" >
  <property name="filter" value="uid=%u,ou=people,dc=univ-rennes1,dc=fr" />
  <property name="contextSource" ref="contextSource" />
</bean>
```

Seront ainsi remplacées par :

```
<bean class="org.esupportail.cas.adaptors.kerberosfeed.KerberosFeedAuthenticationHandlerWrapper" >
  <property name="authenticationHandler">
    <bean class="org.jasig.cas.adaptors.ldap.FastBindLdapAuthenticationHandler" >
      <property name="filter" value="uid=%u,ou=people,dc=univ-rennes1,dc=fr" />
      <property name="contextSource" ref="contextSource" />
    </bean>
  </property>
  <property name="config" ref="kerberosFeedConfig" />
  <property name="registry" ref="kerberosFeedRegistry" />
</bean>
```

Lorsqu'un *authenticationHandler* est enrobé de la sorte et que l'authentification du bean enrobé (ici **FastBindLdapAuthenticationHandler**) s'effectue avec succès, alors elle est suivie de l'ajout dans le royaume de l'utilisateur à l'aide d'une commande **kadmin**, en utilisant la configuration donnée par le bean **kerberosFeedConfig** :

```
<bean
    id="kerberosFeedConfig"
    class="org.esupportail.cas.adaptors.kerberosfeed.KerberosFeedConfig">
  <property name="kadminPath" value="/usr/kerberos/bin/kadmin" />
  <property name="realm" value="UNIV-RENNES1.FR" />
  <property name="principal" value="cas/admin" />
  <property name="useKeytab" value="true" />
  <property name="keytab" value="/etc/cas-admin.keytab" />
  <!-- property name="password" value="secret" /-->
  <property name="passwordAllowedChars"
    value="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789&amp;~#{([-|`\\_^@)]=+}$%*!:/;.,?&gt;
&lt;" />
</bean>
```

La propriété **password** n'est utilisée que lorsque **useKeytab** est positionnée à **false**, dans le cas ci-dessous on utilise la *keytab* générée précédemment.

L'ajout d'un utilisateur déjà présent dans le royaume Kerberos provoque une erreur de **kadmin**, qui n'est pas prise en compte (rien n'est fait dans ce cas, l'erreur n'est pas dommageable). Néanmoins, afin de ne pas rejouer l'ajout d'un utilisateur déjà présent dans le royaume, le bean **KerberosFeedAuthenticationHandlerWrapper** s'appuie sur un registre (propriété **registry**), dans lequel il mémorise les utilisateurs déjà ajoutés dans le royaume. Par défaut (lorsque la propriété **registry** n'est pas positionnée), la mémorisation est faite en mémoire (implémentation **InMemoryRegistryImpl**) et les informations sont perdues à chaque redémarrage du serveur CAS. On peut aussi utiliser un registre basé sur BerkeleyDb, dont les informations seront permanentes :

```
<bean
    id="kerberosFeedRegistry"
    class="org.esupportail.cas.adaptors.kerberosfeed.registry.BerkeleyDbRegistryImpl">
  <property name="dbPath" value="/tmp" />
</bean>
```

Les informations seront ici stockées dans le répertoire **/tmp**.

Test

Se connecter sur le serveur CAS avec un utilisateur de l'annuaire LDAP (par exemple **dupont**) et vérifier qu'il est bien ajouté dans la base Kerberos (**getprinc dupont** sous **kadmin** sur le KDC).