Migration de l'authentification de LDAP à Kerberos (archive)

Sous réserve de complétude des tests en cours, les pages de cet espace montrent qu'il est effectivement possible de complètement déléguer l'authentification des utilisateurs à un serveur Kerberos.

Cette page discute de la migration d'une architecture où l'authentification des utilisateurs est confiée à un annuaire LDAP à une architecture où l'annuaire LDAP n'aurait en charge que son métier propre (le service d'annuaire) et où l'authentification des utilisateurs serait assurée par Kerberos.

Pourquoi ne pas basculer directement de LDAP à Kerberos ?

Parce que c'est impossible.

Cette phase de migration, dans laquelle l'authentification serait assurée à la fois par l'annuaire LDAP et le serveur Kerberos est obligatoirement à envisager pour les raisons suivantes :

- L'ajout d'un utilisateur (de son principal) dans le KDC nécessite la connaissance de son mot de passe ; puisque le mot des utilisateurs dans l'annuaire LDAP est chiffré de manière non réversible, il est impossible de créer automatiquement à un instant donné les comptes des utilisateurs dans le royaume Kerberos à partir des informations contenues dans l'annuaire LDAP.
- Certaines applications (*Legacy*) peuvent nécessiter une authentification LDAP et doivent continuer à fonctionner.

> ⚠ **Application Legacy qui nécessitent une authentification LDAP**
>
> Au terme de la phase de migration, c'est-à-dire quand tous les utilisateurs existants avant la mise en place de Kerberos seront présents dans le royaume, il sera possible (sous réserve de tests) de modifier l'authentification pour qu'un bind user/password soit validé en s'appuyant sur le serveur Kerberos et non les mots de passe contenus dans l'annuaire LDAP.
>
> Il est peut-être même possible de faire en sorte de configurer l'annuaire LDAP de telle manière que les binds user/password soient validés en s'appuyant sur les mots de passe de l'annuaire LDAp présents, et sur le serveur Kerberos sinon, grâce à un démon **sals_authd** utilisant la GSSAPI (à confirmer).

Quand et comment migrer les utilisateurs ?

Il faut donc disposer d'un moyen de créer les utilisateurs dans le royaume Kerberos a la volée, à un moment où on dispose de leur mot de passe.
Les moments candidats sont les suivants :

- Lors de l'authentification sur un poste client. Cette solution, envisageable grâce au module PAM **pam_krb5_migrate**, est rejetée car elle nécessiterait un accès privilégié de création des comptes sur les postes clients, estimé comme un trop gros risque en matière de sécurité.
- Lors de l'authentification à un service, consulté par les utilisateurs de manière suffisamment régulière pour que la migration soit la plus rapide possible. Le seul service candidat suivant cette contrainte est le serveur de mail. Cette solution se heurte néanmoins au fait que dans de nombreux établissements, la consultation des mails se fait (au moins pour les étudiants) à travers un webmail CASifié, qui ne reçoit pas les mots de passe des utilisateurs (c'est dans ce cas le module **pam_cas** qui valide l'authentification des utilisateurs grâce à des ST ou PT émis par le serveur CAS).
- Lors de l'authentification sur le serveur CAS. C'est cette solution qui est retenue.

Comment faire ?

Pour rajouter au royaume Kerberos les utilisateurs qui n'y sont pas déjà

Il faut rajouter au serveur CAS le code nécessaire pour, à chaque fois qu'un utilisateur se connecte avec une authentification différente de Kerberos :

- vérifier si l'utilisateur existe dans le royaume Kerberos
- s'il n'existe pas, le créer avec le mot de passe avec lequel il s'est authentifié auprès d'une autre source (en l'occurrence l'annuaire LDAP).

Voir plus loin pour le code ajouté au serveur CAS.

Pour créer les nouveaux utilisateurs à la fois dans le royaume Kerberos et l'annuaire LDAP

Il faut modifier la procédure de création des comptes des utilisateurs et y rajouter le code nécessaire pour créer les utilisateurs dans le royaume Kerberos.

Selon les procédures en vigueur dans l'établissement, il est également possible de créer le compte dans le royaume Kerberoslors de la première validation du compte à travers une interface web dédiée (comme cela est fait pour l'interface Sésame à l'université de Rennes 1).

Pour maintenir la cohérence des mots de passe LDAP et Kerberos

Il faut pendant la phase de migration s'assurer que les changements de mot de passe soient faits à la fois dans l'annuaire LDAP et le royaume Kerberos.

Pour cela, le changement de mot de passe ne doit pas être possible depuis les postes clients (car il ne serait répercuté dans l'annuaire LDAP) et doit se faire via une interface web centraliséequi répercute les changements à la fois dans le royaume Kerberos et l'annuaire LDAP.

Modifications du serveur CAS

Comme vu précédemment il faut, à chaque fois qu'un utilisateur se connecte avec une authentification différente de Kerberos :

- vérifier si l'utilisateur existe dans le royaume Kerberos
- s'il n'existe pas, le créer avec le mot de passe avec lequel il s'est authentifié auprès d'une autre source (en l'occurrence l'annuaire LDAP).

L'alimentation du royaume Kerberosest faite par un wrapper de AuthenticationHandler ; de cette manière, elle peut être activée pour pour certains handlers seulement.

Modifications des sources

On crée tout d'abord un module supplémentaire nommé **cas-server-integration-kerberosfeed** en installant les sources du zip attaché .

On ajoute le nouveau module dans la liste des modules de **/pom.xml** :

```
<modules>
  <module>cas-server-core</module>
  [...]
  <module>cas-server-webapp</module>
  <module>cas-server-integration-kerberosfeed</module>
</modules>
```

On joute également la propriété **skipTests** au plugin **maven-sunfire** pour éviter de rejouer tous les tests à la compilation :

```
<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>
```

Pour compiler le nouveau module (nécessaire après tout changement), exécuter :

```
mvn \-pl cas-server-integration-kerberosfeed install
```

Ajouter une dépendance du module **cas-server-webapp** vers le nouveau module dans **/cas-server-webapp/pom.xml** :

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

A chaque fois que l'on modifie la configuration du module **cas-server-webapp**, exécuter :

```
mvn -pl cas-server-webapp package
```

Cela créee le war **/cas-server-webapp/target/cas.war** qui peut être déployé.

Configuration

Les beans ci-dessous sont dans le fichier **/cas-server-webapp/src/main/webapp/WEB-INF/deployerConfigContext.xml**.

On remplace tout d'abord le bean FastBindLdapAuthenticationHandler 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>
  <!--
   | The configuration used to feed the Kerberos Realm, mandatory.
  -->
  <property name="config" ref="kerberosFeedConfig" />
  <!--
   | The registry used to store the usernames of the users that have already been added
   | to the realm. Defaults to an 'in memory' implementation where addings will be
   | lost on server startup.
  -->
  <property name="registry" ref="kerberosFeedRegistry" />
</bean>
```

Le bean **kerberosFeedConfig** mutualise la configuration de l'accès au serveur Kerberos :

```
<!--
   | The configuration used to feed the Kerberos Realm.
   | The values below are used to perform a bash kadmin command.
  --><bean id="kerberosFeedConfig"  class="org.esupportail.cas.adaptors.kerberosfeed.KerberosFeedConfig">
  <!--
   | The name of the Kerberos realm, mandatory.
  -->
  <property name="realm" value="UNIV-RENNES1.FR" />
  <!--
   | The name of the principal used to authenticate in kadmin, defaults to cas/admin.
  -->
  <property name="principal" value="cas/admin" />
  <!--
   | Set this property to true to use a keytab to authenticate in kadmin (preferred), or false to use
   | a password. Defaults to true.
  -->
  <property name="useKeytab" value="true" />
  <!--
   | The name of the keytab used when useKeytab is set to true (unused otherwise).
   | Defaults to /etc/admin.keytab.
  -->
  <property name="keytab" value="/etc/admin.keytab" />
  <!--
   | The password used authenticate in kadmin, defaults to secret (you shall probably
   | change it since kadmin authentication will fail with the default value).
  -->
  <!-- property name="password" value="secret" /-->
  <!--
   | A string that contains the chars allowed for the users' passwords (set to the default here).
  -->
  <property
    name="passwordAllowedChars"
    value="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789&amp;~#{([-|`\\_^@)]=+}$%*!:/;.,?&gt;
&lt;" />
</bean>
```

Enfin le bean suivant est le registre utilisé pour mémoriser les noms des utilisateurs qui ont déjà été alimentés dans le royaume Kerberso (pour ne pas rejour deux fois) :

```
<!--
   | The registry used to store the ids of the users that have already been added
   | to the realm. Unlike the default implementation where usernames are backed to
   | memory, the implementation below uses a Berkeley DB and thus is persistent.
  -->
  <bean id="kerberosFeedRegistry"  class="org.esupportail.cas.adaptors.kerberosfeed.registry.
BerkeleyDbRegistryImpl">
    <!--
     | The path used to store the data (must be writable by the tomcat user).
     | Defaults to /tmp.
    -->
  <property name="dbPath" value="/tmp" />
</bean>
```

x