Livrable 4.1.2: Implantation des composants de sécurité Open PaaS

Janvier 2015

LORIA

Ahmed BOUCHAMI, Olivier PERRIN

Contents

1 Gestion des identités numériques (authentification) 3
1.1 Solution Proposée 3
1.2 Implantation 4
1.2.1 Dépendances 4

1 Gestion des identités numériques (authentification)

La gestion des identités numériques dans les plate-formes collaborative est une tâche fondamentale dans les environnements collaboratifs, et notamment dans les réseaux sociaux professionnels tel que *OpenPaaS RSP*. La plateforme OpenPaaS a pour but de permettre à différentes entreprises de collaborer dans un environnement riche et sécurisé. De plus, l'un des principaux objectifs visés par OpenPaaS RSP est de permettre à ces entreprises collaboratrices de préserver au maximum leur autonomie pour le contrôle d'accès à leurs ressources respectives ainsi qu'à la gestion des identités de leurs utilisateurs (vie privée). Dans un milieu professionnel où l'information a une grande valeur, cette idée d'autonomie s'avère très intéressante pour les entreprises, néanmoins, elle n'est pas évidente. Cette difficulté est due aux différents mécanismes de gestion d'authentification utilisés (i.e adoptés) par chaque entreprise. Par conséquent, un problème d'interopérabilité entre les mécanismes d'authentification implémentés au niveau de chaque entreprise est naturellement identifié. La gestion de cette interopérabilité est une tâche très compliquée vu que les mécanismes d'authentification les plus populaires (i.e utilisés) sont codés de différentes manières. Par conséquent, l'implémentation d'une plateforme capable de gérer ces différences semble être une œuvre fastidieuse et très couteuse. La solution que nous avons proposé est l'utilisation de l'outil *LemonLDAP::NG* qui prend en considération la plupart des outils d'authentification classiques, en particulier: login/password, annuaires LDAP, SAML, OAuth, OpenID, etc. *LLDAP::NG* peut être facilement installé au niveau de chaque partenaire[1] sur un serveur Unix.

1.1 Solution Proposée

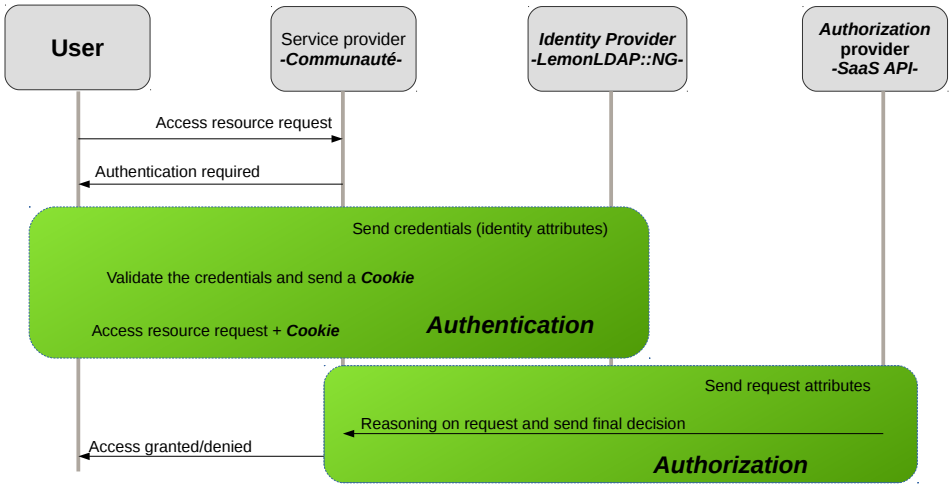

Figure 1: Architecture globale de plateforme de sécurité d'OpenPaaS

Afin d'authentifier un utilisateur, une fois LemonLDAP::NG installé sur un serveur distant, nous avons développé un service Web et nous interrogeons LLDAP::NG via ce service avec une requête d'authentification SOAP contenant des attributs concernant l'identité d'un utilisateur (e.g login et password) afin de récupérer un **cookie**. La valeur de ce *cookie* nous permettra de savoir si la requête d'authentification a été validée ou rejetée (i.e si l'utilisateur en question est authentifié ou pas). Pour cela, le service web établit une connexion avec LemonLDAP::NG et se charge d'envoyer une requête SOAP tout en émettant les attributs d'identité de l'utilisateur. Ces attributs dépendent du type de mécanisme d'authentification choisi au niveau du serveur d'identité LLDAP::NG. Une fois la requête d'authentification reçue, LLDAP::NG génère un cookie

[1] http://lemonldap-ng.org/documentation

en réponse, notre service web récupère le cookie et en se basant sur la valeur de ce cookie, le service web décide si l'utilisateur en question sera authentifié (ou pas).

1.2 Implantation

Pour tester le prototype, nous avons opté pour le mécanisme d'authentification Login/Password avec des comptes utilisateurs prédéfinis dans *LemonLDAP::NG*.

- Nous avons commencé par installer tous les logiciels nécessaires au bon fonctionnement de LLDAP::NG:

  apt-get install apache2 libapache2-mod-perl2 libapache-session-perl libnet-ldap-perl libcache-cache-perl libdbi-perl perl-modules libwww-perl libcache-cache-perl libxml-simple-perl libsoap-lite-perl libhtml-template-perl libregexp-assemble-perl libjs-jquery libxml-libxml-perl libcrypt-rijndael-perl libio-string-perl libxml-libxslt-perl libconfig-inifiles-perl libjson-perl libstring-random-perl libemail-date-format-perl libmime-lite-perl libcrypt-openssl-rsa-perl libdigest-hmac-perl libclone-perl libauthen-sasl-perl libnet-cidr-lite-perl libcrypt-openssl-x509-perl libauthcas-perl libtest-pod-perl libtest-mockobject-perl libauthen-captcha-perl libnet-openid-consumer-perl libnet-openid-server-perl libunicode-string-perl libconvert-pem-perl libmouse-perl,

- ensuite, nous avons installé LemonLDAP::NG

  `http://lemonldap-ng.org/documentation/quickstart`

- enfin, nous avons configuré l'accès SOAP à LemonLDAP::NG

  `http://lemonldap-ng.org/documentation/latest/soapsessionbackend`.

Le nom de la méthode qui nous permet de récupérer le cookie est *getCookies*. Elle se trouve dans la classe *AuthenticationPortTypeProxy* dans le package *lemonLdap*. Ce package est obtenu grâce au fichier de description de service *portal.wsdl* fourni par *lemonLDAP::NG*. La méthode *getCookie* prend en paramètres un nom d'utilisateur et un mot de passe, et si ces paramètres sont valides, elle retourne un Cookie non null, sinon la valeur du Cookie sera null.
Étant donné le choix d'architecture sélectionné pour OpenPaaS (architecture RESTful), nous avons ensuite *wrappé* le service Web comme une ressource accessible via des appels respectant le style d'architecture REST. Le client *REST* qui se charge d'interroger LLDAP::NG et de récupérer le Cookie est illustré dans la figure 3. Ce client établit un appel REST via une requête HTTP de type GET sur le service d'authentification *AuthenticationService*. Au niveau du service d'authentification *AuthenticationService* (basé sur le package LemonLDAP::NG), nous avons défini la méthode *authentication* illustrée dans la figure2. Cette méthode permet d'établir une connexion avec LemonLDAP::NG et récupérer la valeur du Cookie[2]. Cette méthode, nous l'avons exposée pour qu'elle soit accessible par un appel en GET depuis le service d'application ou celui du contrôle d'accès.

1.2.1 Dépendances

La liste des dépendances nécessaires pour l'utilisation de LemonLDAP::NG sous *maven* est la suivante:

- apache-jakarta-commons-discovery
- axis-client
- commons-logging-1.2

[2]La vérification d'authentification de l'utilisateur se fait au niveau du service *AuthenticationService* grâce à la variable booléenne *isAuthenticatedUser*. Par conséquent, au niveau du client, on pourrait récupérer directement la valeur de *isAuthenticatedUser* pour vérifier si l'utilisateur est bien authentifié (ou pas) dans le cas où on voudrait pas faire un test supplémentaire

```java
package boundary;

import lemonldap.AuthenticationPortTypeProxy;
import java.rmi.RemoteException;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.Response;

/**
 *
 * @author ahmed
 */
@Path("authentication")

public class Authentication {

    @GET
    public String authentication(@QueryParam("subject") String subject, @QueryParam("password") String password) throws RemoteException {
        boolean isAuthenticatedUser = false;
        String authenticationCookie = " ";
        AuthenticationPortTypeProxy a = new AuthenticationPortTypeProxy();
        authenticationCookie = a.getCookies(subject, password).getCookies().getLemonldap();
        if (authenticationCookie != null) {
            isAuthenticatedUser = true;
        }
        //return isAuthenticatedUser;
        return authenticationCookie;
    }
}
```

Figure 2: Méthode de l'authentification

```java
public String authentication(String subj, String psw) throws RemoteException {

    Client client = ClientBuilder.newClient();
    WebTarget target = client.target("http://localhost:8080/AuthenticationService/resources/authentication?subject="+subj+"&password="+psw);
    this.RESTCall = target.request().get(String.class);

    return RESTCall;
}
```

Figure 3: Client authentification

- mail-1.4.7
- wsdl4j-1.5.2
- javax.mail-1.5.2
- activation-1.1
- javaee-api-7.0

Afin de faciliter l'installation et le test de LemonLDAP::NG, des comptes de démonstration ont été mis en place. Pour la gestion des comptes utilisateur, une documentation plus détaillée expliquant la procédure à suivre sur : `http://lemonldap-ng.org/documentation/latest/authdemo`.