Université de Bouira
Département d'Informatique
Module : Les Architectures Orientées Services
Niveau M1-ISIL

Série TD / TP 6

Sécurité des Services Web SOAP - Authetification

I. Buts

- ✓ Présentation de la norme WS-Security pour les services SOAP et les différentes possibilités pour sécuriser des services web.
- ✓ Implémentation de la technique d'authentification (login/password).

[1]Les architectures SOA se sont généralisées petit à petit au sein des entreprises et des organisations pour construire des systèmes d'information capables d'offrir des fonctionnalités partagées via des services.

Ces services, souvent implémentés en services web, peuvent être internes et ne concerner qu'une organisation ou être ouverts sur l'extérieur dans le cadre d'échanges B2B.

Dès lors que l'on propose de la valeur ajoutée ou transporte des données dites sensibles, ces services sont alors confrontés à des buts contradictoires :

- Exposer de l'information et la rendre facilement accessible à un tiers (personne ou système)
- Sécuriser l'information pour la rendre uniquement consommable par des personnes ou des systèmes habilités à la voir et à l'utiliser.

La sécurité est au cœur des préoccupations des entreprises pour garantir la cohérence et la pérennité des systèmes.

Au-delà du tout sécuritaire, une approche pragmatique de la sécurité pilotée par les risques est préférable. Rappelons les fonctions de sécurité disponibles qui pourront s'appliquer à des services:

- **Authentification** : le mécanisme qui permet de vérifier l'identité d'une personne/d'un système ;
- **Habilitation** : le droit d'accéder ou non à une fonctionnalité, à une donnée ;
- **Intégrité** (ou signature) : la non modification d'une donnée échangée ;
- **Imputabilité** : traçabilité des actions d'un individu sur un système ;
- **Confidentialité** (ou chiffrement) : la non lisibilité de la donnée par un tiers ne partageant pas un secret.

II. WS Security

WS-Security est un standard pour sécuriser des services web SOAP. Ce dernier aborde les aspects classiques comme l'authentification, l'habilitation mais également des besoins plus rares comme le chiffrement et la signature permettant de garantir l'origine du message en échangeant un jeton chiffré avec une clé partagée (via les certificats X509). Cette norme WS

[1] http://blog.octo.com/securite-des-services-web-1ere-partie/

Security s'applique uniquement au niveau des messages échangés et n'a aucun impact sur le protocole et le transport choisis. *Elle repose sur l'ajout d'éléments dans les headers/en-têtes SOAP*. Ce standard appelé aussi WSS (Web Services Security) tente de normaliser les échanges sécurisés de messages et de garantir l'interopérabilité entre systèmes.

Des Profiles sont disponibles pour couvrir l'ensemble des problématiques liées à la sécurité (authentification, chiffrement et signature). Le consortium OASIS vise la standardisation des spécifications WS-* pour permettre une meilleure interopérabilité entre les technologies.

III. Authentification

L'authentification s'appuie sur la notion de token c'est à dire des *headers SOAP* capables de véhiculer les credentials (typiquement un login et un mot de passe) jusqu'au service cible. Le niveau de confidentialité de ces données échangées peut également être choisi (hash de password, chiffrement des données, etc.).

Les principaux tokens disponibles sont :

- **User Name Token** : échange de couples login/mot de passe ;
- **SAML Token** : échange de données d'authentification au format XML SAML ;
- **X509 Token** : échange de données d'authentification par certificat ;
- **Tickets Kerberos** : échange de clés secrètes**.**

Les credentials échangés entre les services et les clients représentent donc des métadonnées ajoutées dans la partie *header* d'une enveloppe SOAP. Ces métadonnées sont appelées **Message Context** dans JAX-WS.

a. Authentification dans JAX-WS

L'un des moyens les plus utilisés pour gérer l'authentification dans JAX-WS est de demander au client consommateur de fournir et de transmettre un *nom utilisateur* et un *mot de passe*, joint dans l'entête de la requête SOAP. Une fois la requête reçue par le service, ce dernier l'analyse (parser) afin d'extraire les credentials fournis avant les comparer pour les valider (depuis une base données, etc.).

b. Implémentation coté Client

Coté client, le nom d'utilisateur et le mot de passe doivent être ajoutés à l'entête de la requête SOAP. Pour ce faire, le *contexte du message-requête* doit d'abords être récupéré en utilisant l'interface **BindingProvider** et sa méthode **getContextRequest**. Le contexte de requête est une collection java de type java.util.**Map**<String, Objetc>.

```
BindingProvider bp = (BindingProvider) stub;

Map<String, Object> reqContext = bp.getRequestContext();
```

Le nom utilisateur et le mot de passe sont créés et encapsulés dans une collection java type java.util.HashMap<String, List<String>> avant d'être ajoutés comme entête au RequestContext.

```
HashMap<String, List<String>> headers = new HashMap<String,
                                                    List<String>();
headers.put("Username", Collections.singletonList("username"));
headers.put("Password", Collections.singletonList("password"));

reqContext.put(MessageContext.HTTP_REQUEST_HEADERS, headers);
```

c. *Implémentation coté Service Web*

Coté serveur, le contexte de message doit d'abords être récupéré dans un objet javax.xml.ws.handler.**ContextMessage** en utilisant l'interface **WebServiceContext** et sa méthode **getContextMessage**.

Ensuite, l'entête de la requête SOAP reçue est récupéré dans un objet Map, qui sera utilisée pour récupérer le nom utilisateur ainsi que le mot de passe.

NB : Il est préférable d'isoler ces étapes dans une méthode (*isAuthenticated* par exemple) dans la classe implémentation du service web.

```
@Resource
WebServiceMessage wsContext;

Private Boolean isAuthenticated ( ) {
     MessageContexte messageContext = wsContext.getMessageContext( );

     Map httpHeaders = (Map)
     messageContext.get(MessageContext.HTTP_REQUEST_HEADERS) ;

     List usernameList = (List) httpHeaders.get("Username");
     List passwordList = (List) httpHeaders.get("Password");

     //TODO Test && Validation – return true/false
     ….
}
```

IV. Exercice

Ecrire un service web HelloWorld fournissant deux opérations :

- ✓ simpleHello sans paramètre en entrée retournant la chaîne de caractères *Hello World*.
- ✓ nameHello qui prend en paramètre une chaîne de caractères *name* et retourne la chaîne de caractères *Hello World, name*.

Ecrire un client HelloWorldClient consommant les deux opérations du service HelloWorld.

Sécuriser les deux opérations du service web HelloWorld à l'aide des mécanismes d'authentification. Le client ne doit recevoir une réponse SOAP à sa requête que si le nom utilisateur et le mot de passe fournis sont correctes.

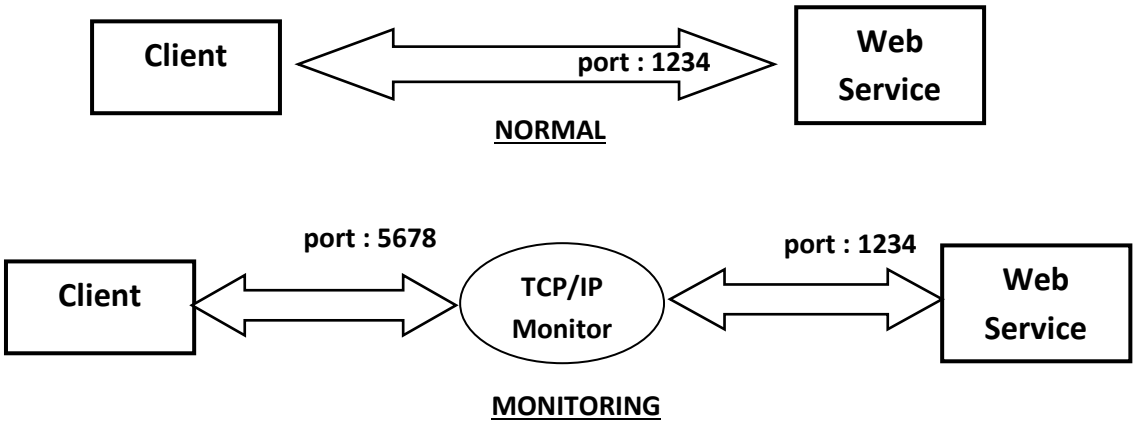
V. Tracer les messages SOAP à l'aide du TCP/IP Monitor d'Eclipse

Dans le but d'analyser, de tracer, et de vérifier le bon fonctionnement des opérations exposées par le service web en cours de développement, il peut être nécessaire d'accéder aux messages SOAP (au format xml) transmis dans les deux sens entre le client et le service. Ci-dessous est expliqué l'utilisation du moniteur TCP/IP d'Eclipse afin de monitorer les communications client-serveur et d'analyser les messages SOAP échangés.

NB : Utiliser cet outil Monitor pour tracer le nom d'utilisateur et le mot de passe dans la requête formulée par le Client (dans l'exercice précédent).

Explication du fonctionnement du TCP/IP Monitor

Le moniteur TCP/IP a pour but d'héberger un serveur entre le client et le service web afin d'intercepter les requêtes envoyées par le client depuis un port donné, les afficher, puis les transmettre au serveur de destination via un autre port. Par conséquent, le moniteur sera capable d'intercepter et de tracer tous les messages qui transitent au milieu. Cette technique est appelée : Port Forwarding. La figure ci-dessous illustre ce fonctionnement [2]:

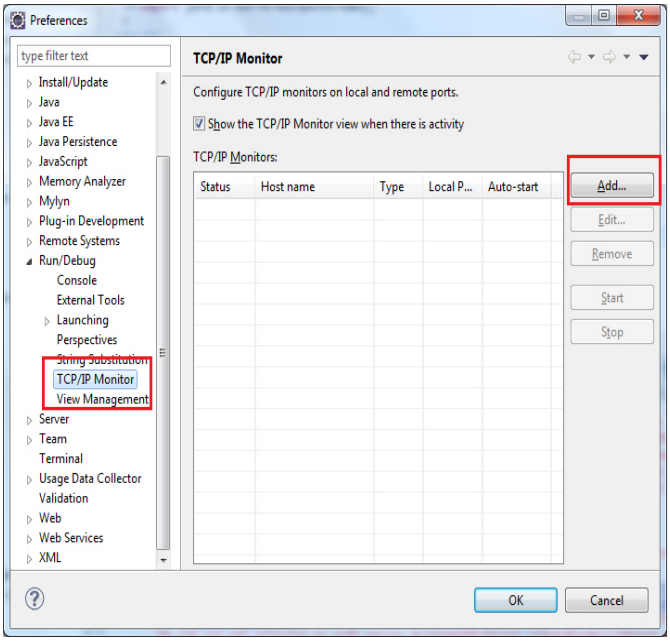


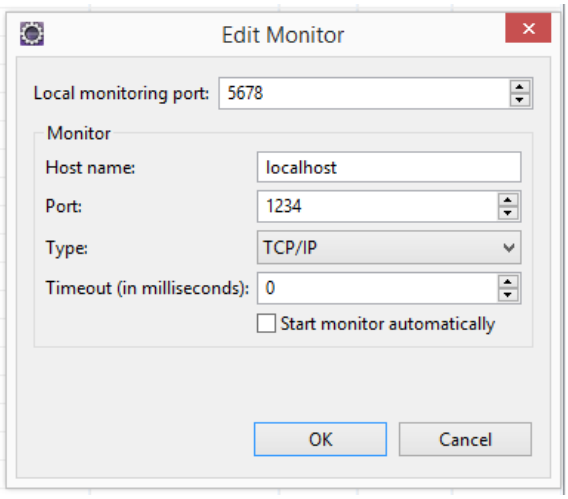
Configuration d'un serveur moniteur

Cas d'utilisation : exercice précédent – Supposons que le service web et le client ont été implémentés avec succès et que le client et le serveur du service web écoutent le numéro de port 1234 (i.e. l'URL de publication du service web est : http://**localhost:1234**/xxyyzz).

- Depuis le projet du **client**, ouvrir la classe d'implémentation <ServiceName> *HelloWorldAuthImplService* générée via wsimport.
- Rechercher et remplacer tous les numéros de port 1234 par 5678.
- Depuis le menu principal d'Eclipse, cliquer sur *Window* → *Preferences* → *Run/Debug* → *TCP/IP Monitor*. (voir figure ci-dessous).
- Cliquer sur le bouton *Add* puis rentrer les informations suivantes dans la fenêtre qui s'ouvre :

[2] http://www.codejava.net/java-ee/web-services/monitoring-soap-messages-using-tcpip-monitor-in-eclipse





- Cliquer sur OK, un nouveau moniteur est ajouté à la liste. Sélectionner le moniteur et cliquer sur ***Start*** puis sur OK pour commencer le traçage.

- Pour afficher la vue (view) du Moniteur TCP/IP, cliquer sur *Window* → *Show View* → *Other* → rechercher et choisir le moniteur depuis la liste. Cliquer sur OK.

- Relancer le client (Run As…) afin d'afficher et d'analyser les messages de communication entre le serveur et le client, capturés par le moniteur configuré.

- Remarquer notamment le nom d'utilisateur et le mot de passe attachés au message de requête SOAP. Exemple :