Authentification des services REST

Auteurs : Laurent HOULETTE - Patrick GIORDANO, Julien LEFEBVRE

Authentification des services REST

Table des matières

Préambule

Les mots-clefs **doit**, **ne doit pas**, **requis**, **devrait**, **ne devrait pas**, **recommandé**, **peut** et **optionnel**, lorsqu'ils apparaissent dans ce document, doivent être interprétés comme décrits dans la [RFC2119].

Règles applicables et exigences

Chaque appel de web services entre applications DOIT être authentifié et PEUT être sécurisé.

Cas d’usage

Tous les appels des web services entre applications WT.

Le mode de sécurisation décrit dans cette fiche s’applique aux échanges entres applications internes de La Poste uniquement :

- Intra-bulles
- Inter-bulles
- Ecrin de/vers bulle

La sécurité des échanges bulles de/vers Internet fait l’objet d’une autre fiche.

Architecture

L’authentification pour les appels de web services internes se fait via la mise en œuvre d’un partage de clé privée entre les applications.

Les clefs secrètes sont fabriquées et gérées par l’exploitant via un processus spécifique (avant sa mise en œuvre, le projet génère ses propres clés) et doivent être connues de l’appelé et de l’appelant.

L’authentification de la requête est basée sur l’utilisation d’une signature qui est envoyée avec la requête et permet à l’appelé d’authentifier l’appelant.

La signature est la concaténation d'un certain nombre d'éléments de la requête, constituant une chaîne de caractères qui est chiffrée en utilisant une clé secrète, ce qui permet d'obtenir le HMAC (Hash Message Authentication Code) de cette chaîne (la signature).

Cette signature est ajoutée à la requête. Lors de la réception, l'appelé utilise la même clé secrète que l'appelant est censé avoir et calcule lui-même la signature, en utilisant les éléments de la requête. Il compare ensuite la clé calculée et la clé reçue pour vérifier que la clé utilisée était bien la même.

Cette solution permet d'assurer l'authentification et de protéger contre le rejeu. Le chiffrement du message et du flux en retour n'est pas assuré.

Solution

Le principe d'authentification est décrit dans le paragraphe ci-dessus. Les détails de l'implémentation sont définis ici :

- Les éléments d'authentification sont transférés dans un cookie dont le nom est unique quel que soit l'appelant ou l'appelé : **authentication**

Signature

- La signature est calculée en réalisant le HMAC (Base 64) d'une chaine de caractère composée de :
  - **Verbe REST (GET, POST, PUT , DELETE)**
  - **URI** , dans le cas d'un POST ou d'un PUT le contenu envoyé n'est pas présent dans l'URI

```
https://copiloteg.net-courrier.extra.laposte.fr/silodepot/depots/v2?q
=toto&champ=2
https://copiloteg.net-courrier.extra.laposte.fr/silodepot/depots
```

  - **date** (telle qu'elle est présente dans le champ *Date* en tête HTTP). La date est au format défini (le premier décrit comme le standard internet : **Sun, 06 Nov 1994 08:49:37 GMT**) dans la section 3.3 de la RFC 2616 sur HTTP 1.1
- Ces trois éléments sont séparés par "**\n**", code ASCII: 10

Le principe de construction de la chaîne d'authentification est le suivant :

```
StringToSign = VerbeREST  + "\n" + URI + "\n" + Date ;

Signature = Base64( HMAC-SHA256(clefSecrete, StringToSign) );

authentication = IdentifiantCleSecrete + ":" + Signature + ":" + Date;
```

- **L'algorithme SHA-256** sera utilisé pour réaliser le HMAC (attention hmac ne fonctionne pas sur sles sp2 : une alerte à été renvoyée à Novell).

Clé secrète et identifiant de la clé secrète

- La clé privée est un chaine de 64 caractères alphanumériques en minuscule.
- La clé secrète est identifiée par un identifiant de clé qui est fourni par le fournisseur du service et permet de savoir quelle clé utiliser pour déchiffrer la signature.
- **Une clé secrète ne peut pas être partagée par plusieurs appelants.**

- Un même appelant pourra optionnellement avoir plusieurs clés privées, que l'appelé pourra reconnaître grâce à leur identifiant. Cet identifiant est placé avant la signature dans le cookie.

    - L'identifiant de la clé secrète aura le format suivant : **<nomsilo>_<*nomressource*>_<nomdelappelant>_<id>**, avec :
        - **<nomsilo>** : nom du silo exposant la ressource (ces noms sont normalisés dans cette fiche) . Ex: entites, ressources
        - **<nomressource>**: **nom exact de la ressource demandée, comme dans l'URI du service**. Ex : utilisateur.
        - **<nom de l'appelant>** : code de l'application sans underscore (ex : T1U1) ou nom de l'application si elle n'a pas de code (Ex : exadelais)
        - **<id>** : identifiant de la clé pour le cas où il y en aurait plusieurs. Il s'agit d'un numéro incrémenté.
    - Exemple d'identifiant de signature : utilisateurs_utilisateur_T1U1_1
    - Exemple de contenu complet du cookie : utilisateurs_utilisateur_T1U1_1:24df843T4seerfrksp53n763ndff978634d7ghr:Sun, 06 Nov 1994 08:49:37 GMT

Le format de la chaîne d'authentification passée dans le cookie aura finalement le format suivant : **<IdentifiantCleSecrete>:<signature>:<Date>**

Stockage des clés secrètes

- Les clés secrètes d'un appelant sont placées dans un **fichier unique** ayant pour format :
    - **IdentifiantCleSecrete**=**clefSecrete**

```
utilisateurs_utilisateur_T1U1_1=24dfn843T4dfsererfgrksp53klnf986bns6d
nfer76390ndfnf978634d7g3hr9
utilisateurs_utilisateur_T1U2_1=avfhnjy45hdtcnks67jax1cknvbcgwgjvx4qp
gds709djklnq896lknfg07dfgrf
```

- Les clés secrètes seront stockées dans :
    - le répertoire racine de la version du module applicatif : **/<ccx>/<ccx>adm/<tm>/<xx><*yy*><zz>.000/.**
    - plus spécifiquement dans le sous répertoire : **web/config/secretkeys/**
    - le nom du fichier sera : **secretkeys_<ccx><tm>.ini.**
    - Pour le module u2 de l'application t5_, nous aurons par exemple : /t5_/t5_adm/u2/<xx><*yy*><zz>.000/web/config/secretkeys/secretkeys_t5_u2.ini
- Les droits sur ce répertoire doivent être définis pour n'autoriser que l'utilisateur Apache (www-data) et les gestionnaires d'instance Unix (batchs) à y accéder en lecture.

Vérification de l'horodatage

Afin de s'assurer qu'un appel de service ne peut être rejoué dans le temps par un tiers malveillant, le fournisseur de service vérifie à la réception de l'appel l'horodatage transmis, en prenant une marge de temps de 20 secondes (convention prise dans le cadre du projet mais ajustable en fonction des contraintes) correspondant à la durée max de désynchronisation possible entre l'appelant et l'appelé.
Plus précisément, à la réception de l'appel, le fournisseur de service compare la date passée dans l'en tête HTTP par l'appelant avec la date de réception de la demande.
La requête est validée si et seulement si : **<date+heure de l'appelé>-20 secondes <= <date+heure de la requête> <= <date+heure de l'appelé>+20 secondes**

Remarque importante: Les transferts de fichiers HTTP devront utiliser ce principe d'authentification.

Annexe : exemple de cookie authentication valide

Données en entrée	Valeur
VerbREST	GET
URI	http://ute/UTE/v1
Date	Tue, 05 Jun 2012 13:58:19 GMT
Identifiant de la clé secrète	tae_enveloppe_T1U1_1
Valeur de la clé secrète	419bed03be8d19f04d25fbea99353bd0

Données de la chaîne d'authentification	Valeur
StringToSign	GET\nhttp://ute/UTE/v1\nTue, 05 Jun 2012 13:58:19 GMT
Signature	B3oGnF0jxArv5s8aHy8YjDph9NQ7w186HLx0dpaaL8U=
Cookie authentication	tae_enveloppe_T1U1_1:B3oGnF0jxArv5s8aHy8YjDph9NQ7w186HLx0dpaaL8U=:Tue, 05 Jun 2012 13:58:19 GMT

Logs côté silo en mode DEBUG

```
[LaPoste::HMAC][DEBUG] Begin handler
[LaPoste::HMAC][DEBUG] GET /UTE/v1 HTTP/1.1
User-Agent: PECL::HTTP/1.7.0-dev (PHP/5.3.1)
Host: ute
Accept: */*
Cookie:
authentication=tae_enveloppe_T1U1_1:B3oGnF0jxArv5s8aHy8YjDph9NQ7w186HLx0dpaaL8U=:
Tue, 05 Jun 2012 13:58:19 GMT;
Accept-Encoding: gzip,deflate, gzip;q=1.0,deflate;q=0.5
Keep-Alive: 300
Connection: keep-alive

HTTP/1.1 (null)


[LaPoste::HMAC][DEBUG] Keystore = C:/wamp/www/secretkeys / Delay = 20
[LaPoste::HMAC][DEBUG] Exclude URI : /ping
[LaPoste::HMAC][DEBUG] Asked: http://ute/UTE/v1
[LaPoste::HMAC][DEBUG] key = authentication, value =
tae_enveloppe_T1U1_1:B3oGnF0jxArv5s8aHy8YjDph9NQ7w186HLx0dpaaL8U=:Tue, 05 Jun
2012 13:58:19 GMT
[LaPoste::HMAC][DEBUG] Orig Cookie =
tae_enveloppe_T1U1_1:B3oGnF0jxArv5s8aHy8YjDph9NQ7w186HLx0dpaaL8U=:Tue, 05 Jun
2012 13:58:19 GMT
[LaPoste::HMAC][DEBUG] Cookie=3
[LaPoste::HMAC][DEBUG] Passed URI Filter
[LaPoste::HMAC][DEBUG] m=GET u=http://ute/UTE/v1 d=Tue, 05 Jun 2012 13:58:19 GMT
[LaPoste::HMAC][DEBUG] now=1338904701 (2012-06-05T13:58:21)
[LaPoste::HMAC][DEBUG] req - delay=1338904679
[LaPoste::HMAC][DEBUG] req + delay=1338904719
[LaPoste::HMAC][DEBUG] passed date 1
[LaPoste::HMAC][DEBUG] passed date 2
[LaPoste::HMAC][DEBUG] Chargement fichier clés
[LaPoste::HMAC][DEBUG] Input key=tae_enveloppe_T1U1_1
[LaPoste::HMAC][DEBUG] Testing key tae_enveloppe_T1U1_1 value
419bed03be8d19f04d25fbea99353bd0
[LaPoste::HMAC][DEBUG] Key found (no ts) : tae_enveloppe_T1U1_1 val =
419bed03be8d19f04d25fbea99353bd0
[LaPoste::HMAC][DEBUG] Key to sign : |GET
http://ute/UTE/v1
Tue, 05 Jun 2012 13:58:19 GMT|
[LaPoste::HMAC][DEBUG] key= B3oGnF0jxArv5s8aHy8YjDph9NQ7w186HLx0dpaaL8U=
[LaPoste::HMAC][DEBUG] Return OK
```