SECURISER SON API REST

SSL

Une connexion SSL crypte le transfert des données. En utilisant une connexion https pour vos webservices vous diminuez le risques de fuite de données.

Si vous travaillez avec des données personnelles liées au RGPD, vous devez impérativement opter pour une connexion https.

Contraintes :
1 - nom de domaine
2 - Certificat SSL (avec let's encrypt)

FILTRAGE IP

Si votre API est destinée en interne vous pourrez filtrer les IP autorisées.

La fonction WebserviceAdresseIPClient() vous permettra de récupérer cette adresse IP.
Il suffit ensuite de la vérifier que l'IP est autorisée.

Contraintes :
1 - Avoir une base de données à jour des IP
2 - Les utilisateurs doivent avoir des IP Fixes

QUOTA

Si votre API est en accès tout public, vous devez contrôler le nombre d'exécution par IP par heure ou minutes.
Vous déterminez un quota d'utilisation ou Rate Limit.

Vous comptez les appels pour une même IP ou une même clé API (ou un token) dans un lapse de temps.
Vous temporisez ainsi le nombre des appels.

Contrainte :
1 - Sauvegarder et compter les appels

CRYPTAGE

A utiliser si vous transférer des données privées ou sensibles. Ajouté au https, vous diminuez davantage les risques de fuite.

Que faut-il crypter ?
Les paramètres envoyés qui se situent après la partie fixe de l'URL de l'API.

THIERRY TILLIER

SECURISER SON API REST

CLÉ API

C'est une pratique courante sur de nombreux systèmes publiques.
Ces clés permettent de savoir qui se connecte.

En pratique :
1 - on crée un compte (éventuellement avec une authentification oAuth)
2 - On récupère une clé API unique (dont l'usage peut être limité dans le temps).
3 - On insère cette clé dans l'URL pour être identifié et autorisé.

ex d'url :
https://mondomaine.com/LaCleAPI/v1/clients

Inconvénient, la clé est passé en clair.

TOKEN

Le token est un jeton d'identification temporaire. Certains services publics nécessitent un compte pour utiliser les API. Ensuite un jeton est crée et envoyer à l'utilisateur (le programme).

En pratique :
1 - On demande un Token (en fournissant son compte et son mot de passe - souvent fournis par le créateur du webservice)
2 - On récupère le Token dans le code et on le transmet à chaque appel REST

Contraintes :
Le token est transmis dans l'entête de la requête. Pour le récupérer :
sMonToken = WebserviceLitEnteteHHTP("NomDuToken")

CERTIFICAT

A partir de WINDEV 24 on peut insérer un certificat. Cette technique est un peu plus compliquée à mettre en place car il faut un certificat.

En pratique :
1 - Créer le certificat avec une clé privée (*.p12)
2 - le client : signe la chaine avec CertificatSigneChaine
3 - La chaine signée est passé dans l'entête (et pas dans l'URL)
4 - le webservice : récupère la signature (WebserviceLitEnteteHHTP)
5 - Vérification de la chaîne avec CertificatVérifieChaine(..)

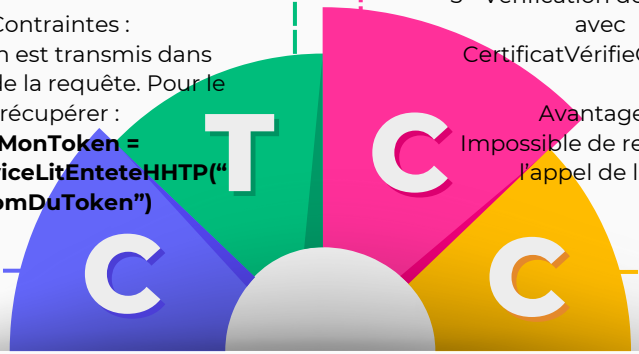
Avantage :
Impossible de reproduire l'appel de l'API

BONNE PRATIQUE

Vous pouvez choisir une technique ou bien en combinée plusieurs complémentaires.

Ex :
- SSL + Cryptage + Quota
- SSL + Cryptage + Clé API + Quota
- SSL + Cryptage + Token + Quota
- SSL + Certificat + Quota

Il est impossible de garantir une parfaite étanchéité des données mais on peut largement diminuer la surface d'attaque.



THIERRY TILLIER