Présentation des JSON Web Token (JWT)

Pendant longtemps, les cookies ont été utilisés pour identifier les utilisateurs. Cela fonctionne encore parfaitement aujourd'hui pour certaines applications. Il est parfois cependant nécessaire de disposer d'un peu **plus de flexibilité**. C'est là qu'interviennent les JSON Web Token. Ce **nouveau standard ouvert** est de plus en plus pris en charge par les principaux sites Web et applications. Découvrez ce qu'est le standard JWT, son fonctionnement et ses applications

Sommaire

1 – Qu'est-ce qu'un JSON Web Token (JWT) ?

JWT (JSON Web Token) est une norme ouverte qui définit une procédure globale et adaptée aux URL pour transmettre en toute sécurité des informations sous la forme d'un objet JSON entre des parties. Un jeton JWT est souvent utilisé pour sécuriser des API RESTful car il peut être utilisé pour authentifier un client qui veut accéder aux API.

Un JSON Web Token est un **access token** (jeton d'accès) aux normes RFC 7519 qui permet un échange sécurisé de donnée entre deux parties. Il contient **toutes les informations importantes** sur une entité, ce qui rend la consultation d'une base de données superflue et la session n'a pas besoin d'être stockée sur le serveur (stateless session).

Les JSON Web Token sont particulièrement appréciés pour les opérations d'identification. Les messages courts peuvent être chiffrés et fournissent alors des informations sûres sur l'identité de l'expéditeur et si celui-ci **dispose des droits d'accès requis**. Les utilisateurs eux-mêmes ne sont qu'indirectement en contact avec les token, par exemple lorsqu'ils entrent un nom d'utilisateur et un mot de passe dans un masque. La véritable communication se fait entre les différentes applications du côté serveur et client.

2 – Structure d'un JSON Web Token

Un JWT signé se compose de trois parties codées en base64 et séparées par un point :

```
HEADER.PAYLOAD.SIGNATURE
```

Que signifient ces trois composantes ?

En-tête (Header)

L'en-tête, ou header, est en général composé de deux parties et fournit des informations essentielles sur le token. Il contient le type de token et l'algorithme de signature et/ou de chiffrement utilisé. Un exemple de header de JWT :

```
{ "alg": "HS256", "typ": "JWT" }
```

Il est recommandé d'entrer JWT pour le type. Il correspond au type de média « application/jwt » de l'IANA. Dans l'exemple ci-dessus, l'en-tête indique que HMAC-SHA256, abrévié « HS256 », est utilisé comme signature du token. Il existe **d'autres méthodes courantes de chiffrement**, comme RSA avec SHA-256 (« RS256 ») et ECDSA avec SHA-256 (« ES256 »). Il est recommandé de toujours utiliser un chiffrement. Si les données ne requièrent pas un haut niveau de protection, il est possible d'indiquer « none » pour le chiffrement. Les valeurs possibles sont standardisées par la JSON Web Encryption **(JWE)** selon le **RFC 7516.**

Le paramètre « cty » pour « Content Type » vient s'ajouter pour les JSON Web Token avec signature ou chiffrement complexe. Il contient également la valeur « JWT ». Dans les autres cas, ce paramètre est laissé de côté.

2-1- charge utile (Payload)

La charge utile du JSON Web Token est la partie qui contient les informations qui doivent être transmises à l'application. C'est là que sont définis certains standards qui déterminent quelles données doivent être transmises. Les informations sont fournies en **"paire clé/valeur"**, les clés sont appelées **« claims »** dans les JWT.

Il existe trois types de claims :

- **Les claims réservées** sont des claims qui sont enregistrées dans le IANA JSON Web Token Claim Register. Leur objet est défini dans une norme. Comme exemple, on peut citer l'émetteur du token (« iss » pour Issuer), le domaine cible (« aud » pour Audience) et la date d'expiration (« exp » pour Expiration Time). On utilise des noms de claims courts afin de limiter au maximum la longueur des token.
- **Les claims publiques** peuvent être définies librement. Il n'y a aucune limitation. Afin que la sémantique de la clé n'occasionne pas de collision, il est nécessaire d'enregistrer publiquement la claim dans le IANA JSON Web Token Claim Register ou de lui attribuer un nom résistant aux collisions.
- **Les claims privées** sont conçus pour les informations qui doivent être échangées spécifiquement avec une application donnée. Les claims publiques contiennent des informations comme « Name » ou « E-mail » tandis que **les claims privées sont plus spécifiques**. Parmi les informations standards pour ce genre de claim, on retrouve par exemple un « identifiant utilisateur » ou un « nom de département » concret. Il est nécessaire de faire attention à éviter les collisions avec les claims réservées ou publiques dans le domaine de nom.

Toutes les claims sont optionnelles. Vous n'avez donc pas à utiliser toutes les claims réservées. En règle générale, la charge utile peut contenir autant de claims que nécessaire, il est cependant recommandé **de limiter les**

informations du JWT au strict nécessaire. Plus le JWT est gros, plus il demande de ressources pour être (dé)codé.
La charge utile peut donc être structurée comme suit :

```
{ "sub": "123", "name": "Alice", "exp": 30 }
```

2-2- Signature

La signature d'un JSON Web Token est créée grâce au codage base64 de l'en-tête et de la charge utile et la méthode de signature/cryptage spécifiée. La structure est définie par la JSON Web Signature (JWS), une norme standardisée selon **le RFC 7515**. Pour que la signature fonctionne, il est **nécessaire d'utiliser une clé secrète** connue uniquement de l'application source. Cette signature vérifie d'une part que le message ne sera pas modifié pendant le transfert. D'autre part, dans le cas d'un jeton signé avec une clé privée, il authentifie également l'expéditeur du JWT.

Plusieurs procédés sont disponibles en fonction de la sensibilité des données :

1. **Aucune sécurité** : comme expliqué plus haut, lorsque les données ne nécessitent pas de protection, la valeur « none » peut être indiquée dans l'en-tête. Dans ce cas, aucune signature n'est générée. Le JSON Web Token est alors uniquement composé d'un en-tête et d'une charge utile. Sans sécurité, la charge utile est lisible en texte clair après le déchiffrage de la base64 et il n'est pas vérifié si le message vient du bon expéditeur ou s'il a été modifié lors du transfert.
2. **Signature (JWS)** : en règle générale, il suffit de vérifier si les données viennent du bon expéditeur et si elles ont été modifiées. C'est là qu'intervient le schéma JSON Web Signature (JWS) qui assure que les messages n'ont pas été modifiés pendant le transfert et proviennent du bon expéditeur. Grâce à cette procédure, la charge utile peut également être lue en texte clair après le décryptage base64.
3. **Signature (JWS) et chiffrement (JWE)** : il est possible d'ajouter une JSON Web Encryption (JWE) au JWS. JWE chiffre le contenu de la charge utile qui est ensuite signé par JWS. Un mot de passe commun ou une clé privée est utilisé pour le chiffrement du contenu. L'expéditeur est également vérifié, le message est fiable et authentique et la charge utile n'est plus lisible en texte clair après le décryptage base64.

   Le cryptage crée une séquence de caractères apparemment aléatoire :

```
{ 7WK5T79u5mIzjIXXi2oI9Fglmgivv7RAJ7izyj9tUyQ }
```

3 – Comment fonctionne un JSON Web Token ?

Il est très aisé de comprendre le fonctionnement d'un JSON Web Token en utilisant l'exemple d'une connexion utilisateur. Une **clé secrète est déterminée** avant l'utilisation du JWT. Dès qu'un utilisateur a entré avec succès ses données de connexion, le JWT est renvoyé avec la clé et stocké localement. Le transfert se fait par HTTPS afin de mieux protéger les données.

Lorsque l'utilisateur veut accéder à des ressources protégées comme une API ou un chemin d'accès protégé, le JWT sera envoyé par l'agent utilisateur comme paramètre (par exemple « jwt » pour les GET-Requests) ou comme en-tête d'autorisation (pour POST, PUT, OPTIONS, DELETE). **L'interlocuteur peut déchiffrer le JSON Web Token** et si le contrôle est réussi, exécuter la demande.

4- Dans quel cas utilise-t-on un JSON Web Token

Les JSON Web Token offrent un certain nombre d'avantages comparés aux méthodes traditionnelles d'authentification et d'autorisation avec des cookies et sont pour cela utilisé dans les scénarios suivants :

1. Applications **REST** : dans les applications REST, le JWT sécurise le protocole sans état en envoyant les informations pour l'authentification directement lors de la requête.
2. **Cross origin resource sharing** : le JSON Web Token envoie les informations lors du Cross Origin Resource Sharing. Cela présente un énorme avantage par rapport aux cookies, qui ne sont généralement pas envoyés dans cette procédure.
3. **Utilisation de plusieurs frameworks** : les JSON Web Token sont standardisés et donc polyvalents. Lors de l'utilisation de plusieurs Framework, ils permettent de partager facilement les données d'authentification.

5 – Exemple d'implémentation

À l'aide d'un exemple de JWT, découvrez à quoi ressemble le token final. Pour ce faire, nous utilisons l'exemple d'en-tête que nous avons mentionné au début :

```
{
        "alg": "HS256",
        "typ": "JWT"
}
```

Une charge utile de JSON Web Token peut ressembler à ceci :

```
{
        "sub": "0123456789",
        "name": "Jean Dupont",
        "admin": true
}
```

Pour obtenir la structure réelle du JWT (trois parties séparées par des points), l'en-tête et la charge utile doivent être codés en base64. Pour l'en-tête, cela ressemble à ceci :

```
base64Header = base64Encode(header)
// eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
```

Il en va de même pour la charge utile :

```
base64Payload = base64Encode(payload)
//
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9
```

Il faut encore créer la signature. Dans l'en-tête, nous avons indiqué qu'il doit être signé avec HMAC-SHA256

```
signature = HS256(base64Header + '.' + base64Payload, 'secret')
// dyt0CoTl4WoVjAHI9Q_CwSKhl6d_9rhM3NrXuJttkao
```

Pour finir, il faut rassembler ces trois composantes et les séparer par un point.

```
Token = base64Header + '.' + base64Payload + '.' + signature
//
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.dyt0CoTl4WoVjAHI9Q_CwSKhl6d_9rhM3NrXuJttkao
```

La plupart des langages de programmation disposent généralement d'une bibliothèque pour générer les JSON Web Token, ce qui rend la mise en œuvre manuelle superflue.

:

```
signature = HS256(base64Header + '.' + base64Payload, 'secret')
// dyt0CoTl4WoVjAHI9Q_CwSKhl6d_9rhM3NrXuJttkao
```

Pour finir, il faut rassembler ces trois composantes et les séparer par un point.

```
Token = base64Header + '.' + base64Payload + '.' + signature
//
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.dyt0CoTl4WoVjAHI9Q_CwSKhl6d_9rhM3NrXuJttkao
```

La plupart des langages de programmation disposent généralement d'une bibliothèque pour générer les JSON Web Token, ce qui rend la mise en œuvre manuelle superflue.

Document extraut de ;

IONOS (www.ionos.fr)

Présentation et exemples de JSON Web Token (JWT) - IONOS

ANNEXE

La spécification de JWT propose différents champs (ou paramètres) standards, appelés *Claims*[5] :

- iss : créateur (*issuer*) du jeton
- sub : sujet (*subject*) du jeton
- aud : audience du jeton
- exp : date d'expiration du jeton
- nbf : date avant laquelle le jeton ne doit pas être considéré comme valide (*not before*)
- iat : date à laquelle a été créé le jeton (*issued at*)
- jti : identifiant unique du jeton (*JWT ID*)

Tous ces paramètres sont optionnels. Ils permettent simplement de définir plus précisément un jeton et de renforcer sa sécurité (e*.g.* en limitant la durée de vie d'un jeton).

Vous pouvez voir une liste complète des réclamations enregistrées à l'adresseRegistre des revendications de jetons Web IANA JSON.

https://www.iana.org/assignments/jwt/jwt.xhtml#claims

https://www.iana.org/assignments/jwt/jwt.xhtml#claims