Engineer

JWT

Jeton compact adapté aux URL, avec déclarations JSON

Ce que c’est

Un JWT est une courte chaîne qui transporte un ensemble de déclarations d’une partie à une autre sous une forme que le destinataire peut vérifier. Les lettres signifient JSON Web Token. La RFC 7519 l’a défini en mai 2015 comme une représentation compacte de déclarations adaptée à une URL, et précise que le nom se prononce «jot». Une déclaration est un nom et une valeur affirmés au sujet d’une entité : `"sub": "24400320"` indique de qui traite le jeton, `"exp": 1300819380` indique quand il cesse d’être valide.

Sur le réseau, un JWT est une suite de chaînes base64url jointes par des points. Un jeton signé en possède trois : un en-tête qui nomme l’algorithme, une charge utile contenant les déclarations et une signature calculée sur les deux premiers. Un jeton chiffré en possède cinq, et ses déclarations ne peuvent pas être lues sans la clé. La RFC 7519 n’effectue elle-même aucune opération cryptographique. Elle s’appuie sur JSON Web Signature (RFC 7515) pour les jetons signés ou protégés par un code d’authentification de message, et sur JSON Web Encryption (RFC 7516) pour les jetons chiffrés ; les algorithmes figurent dans la RFC 7518, et les clés peuvent être publiées au format JSON Web Key de la RFC 7517.

`HS256` est un HMAC avec SHA-256 : un même secret partagé crée et vérifie le jeton, si bien que tout service capable de le vérifier peut aussi en émettre. `RS256` et `ES256` utilisent une paire de clés : l’émetteur signe avec une clé privée et tous les autres vérifient avec la clé publique, librement distribuable. La RFC 7519 n’impose d’implémenter que `HS256` et le mode non signé `none`, et recommande `RS256` et `ES256`.

La section 4.1 enregistre sept noms de déclarations : `iss` (émetteur), `sub` (sujet), `aud` (destinataire), `exp` (instant d'expiration), `nbf` (pas avant), `iat` (instant d'émission) et `jti` (identifiant JWT unique pouvant empêcher le rejeu d'un jeton). Ils ont trois lettres parce que le format se veut compact, et tous sont facultatifs. L'application qui utilise le jeton choisit les déclarations qu'il doit contenir. Au-delà de cet ensemble enregistré, chacun peut définir des déclarations publiques, soit enregistrées auprès de l'IANA, soit dotées d'un nom sans collision possible avec celui d'autrui, comme une URL, ou des déclarations privées convenues entre deux parties.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce qu'un JWT ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Des comptes, mots de passe et droits attribués selon les besoins exacts de chaque poste, puis retirés le jour du départ.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. 1 000 règles RBAC pour sécuriser Linux — 1,000 règles d'accès écrites pour combler une faille de sécurité

1 000 règles RBAC pour sécuriser Linux Contactez-nous →

Pourquoi c'est important

L'intérêt d'un JWT est que le destinataire peut le vérifier seul. Avec la clé de l'émetteur et une horloge, un service peut déterminer si le jeton est authentique, à qui il est destiné et s'il reste valide, sans consulter de base de données ni appeler l'émetteur. La RFC 9068 (octobre 2021) constate que de nombreuses implémentations commerciales d'OAuth émettent des jetons d'accès que les serveurs de ressources analysent et valident directement. Ce schéma est courant lorsqu'une frontière sépare le serveur émetteur du serveur qui accepte les jetons, et beaucoup de ces implémentations utilisent JWT. Elle en a ensuite normalisé la structure pour permettre l'interopérabilité entre fournisseurs.

C'est aussi le format d'OpenID Connect : chaque jeton d'identité est un JWT adressé à l'application qui l'a demandé. Le fournisseur doit le signer sauf si les deux conditions permettant de fixer `alg` à `none` sont réunies : le type de réponse ne renvoie aucun jeton d'identité depuis le point de terminaison d'autorisation, comme dans le flux de code d'autorisation, et le client a explicitement demandé `none` lors de son enregistrement. Lorsque le type de média `application/jwt` a été enregistré en 2015, le registre citait déjà OpenID Connect, Salesforce, Google, Android, Windows Azure et Amazon Web Services parmi ses utilisateurs.

La RFC 7519 destine le format aux emplacements contraints en taille, dont l'en-tête Authorization et les paramètres de requête d'une URI : les noms des déclarations sont donc courts, et l'encodage peut figurer dans une URL sans échappement.

Ce qui rend un JWT peu coûteux à vérifier le rend difficile à révoquer. Un destinataire qui vérifie la signature et l'horloge acceptera le jeton jusqu'à `exp`, quoi qu'il soit arrivé au compte entre-temps.

À quoi cela ressemble en pratique

La RFC 7519 déroule un exemple complet. L'en-tête déclare que l'objet est un JWT et qu'il est protégé par HMAC SHA-256 :

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

Les déclarations nomment un émetteur, un instant d'expiration et une déclaration publique dont le nom est une URL, pour éviter une collision avec une déclaration définie par quelqu'un d'autre :

```
{"iss":"joe",
 "exp":1300819380,
 "http://example.com/is_root":true}
```

Chaque partie est encodée en base64url, le HMAC est calculé sur l'en-tête et la charge utile joints par un point, puis les trois résultats sont réunis en une chaîne. La RFC l'affiche avec des sauts de ligne pour la présentation seulement :

```
eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9
.
eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA6Ly9leGFt
cGxlLmNvbS9pc19yb290Ijp0cnVlfQ
.
dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
```

Trois choses découlent de cet exemple. D'abord, toute personne qui détient le jeton peut lire les déclarations : base64url est un encodage, et décoder la partie centrale restitue le JSON sans aucune clé. Ensuite, la signature couvre exactement les chaînes encodées telles qu'elles ont été envoyées. L'en-tête de la RFC contient un saut de ligne CRLF et une espace initiale, dont elle donne les octets exacts pour cette raison ; reformater le JSON puis le réencoder produirait une autre chaîne, et la signature ne correspondrait plus. La clé de cet exemple figure dans l'annexe A.1 de la RFC 7515 et permet de vérifier la signature. Enfin, le jeton a expiré depuis longtemps : `1300819380` secondes après le 1er janvier 1970 correspondent au 22 mars 2011 à 18:43 UTC, si bien qu'un validateur correct le rejette aujourd'hui.

Un destinataire qui reçoit un tel jeton effectue un ensemble fixe de vérifications avant de lire une seule déclaration. Il sépare la chaîne aux points, décode l'en-tête et confirme qu'il s'agit de JSON valide contenant uniquement des paramètres et valeurs dont il comprend et prend en charge la syntaxe et le sens, ou qui sont explicitement prévus pour être ignorés lorsqu'ils sont inconnus. Il compare `alg` aux algorithmes qu'il a choisi d'accepter pour cet émetteur et cette clé : la RFC 7519 précise qu'il s'agit d'une décision de l'application et qu'un jeton devrait être rejeté si son algorithme est inacceptable, même si la signature est valide. Il vérifie la signature avec une clé appartenant à l'émetteur. Puis il vérifie les déclarations : `iss` doit être un émetteur de confiance, `aud` doit désigner le destinataire lui-même, l'heure actuelle doit précéder `exp` , avec une tolérance de quelques minutes au plus pour le décalage des horloges, et suivre `nbf` si cette déclaration est présente. Ce n'est qu'alors qu'il agit sur les déclarations.

Un jeton d'accès JWT conforme à la RFC 9068 montre ce qu'ajoute un profil concret. Son en-tête dans cette RFC est `{"typ":"at+JWT","alg":"RS256","kid":"RjEwOwOA"}` : un type explicite, un algorithme asymétrique et l'identifiant de la clé dans l'ensemble publié par l'émetteur. Le profil rend `iss` , `exp` , `aud` , `sub` , `client_id` , `iat` et `jti` obligatoires, et un serveur de ressources doit vérifier que `typ` vaut `at+jwt` ou `application/at+jwt` et rejeter les autres types. Ces noms de type et de sous-type de média sont insensibles à la casse ; `at+JWT` dans l'exemple est une écriture équivalente.

Les déclarations d'un jeton sont un instantané du moment de la signature : tout ce qui peut changer avant `exp` est donc mieux décidé sur le serveur. Un des projets de ce site, le durcissement d'une application à chaque couche, revérifie

les droits — niveau, rôle, organisation et périmètre par navire — sur le serveur à chaque requête et ne traite les contrôles du navigateur que comme des éléments d'expérience utilisateur.

Où les problèmes surviennent

Le jeton choisit son propre algorithme. L'en-tête `alg` apporte de la souplesse, et la RFC 8725 décrit comment certaines bibliothèques lui ont fait trop confiance. Un attaquant pouvait le changer en `none`, et la bibliothèque acceptait le jeton sans aucune signature ; ou remplacer `RS256` par `HS256`, et la bibliothèque vérifiait un HMAC en utilisant comme secret partagé la clé RSA publique de l'émetteur, que l'attaquant peut télécharger. Configurez les algorithmes acceptés dans la bibliothèque, n'utilisez chaque clé qu'avec un seul algorithme et rejetez `none` sans exception.

Le secret HMAC est un mot de passe. Un jeton signé avec `HS256` à l'aide d'un secret court et mémorisable peut être attaqué hors ligne : toute personne détenant un seul jeton peut essayer ses hypothèses à son rythme, sans toucher votre serveur. La RFC 8725 interdit d'utiliser directement un mot de passe mémorisable comme clé HMAC, et la RFC 7518 exige une clé `HS256` d'au moins 256 bits. Générez la clé à partir d'une source d'aléa cryptographique, conservez-la dans un coffre de secrets et préférez une paire de clés lorsque de nombreux services ont seulement besoin de vérifier.

La signature est valide, donc on croit tout. Une signature valide prouve qui a émis le jeton, pas à qui il était destiné. Si un émetteur sert plusieurs services et que le destinataire omet `aud`, un jeton légitimement envoyé à un service peut être rejoué contre un autre. La RFC 8725 appelle cela une substitution et exige de tout destinataire de tels jetons qu'il vérifie à qui ils sont destinés et rejette un jeton qui n'en indique aucun. Vérifiez `aud` par rapport à votre propre identifiant et vérifiez que la clé de signature appartient à l'émetteur nommé dans `iss`.

Un type de jeton passe pour un autre. Un jeton d'identité, un jeton d'accès et un jeton d'événement de sécurité du même émetteur peuvent beaucoup se ressembler ; un destinataire qui ne vérifie que la signature et l'émetteur peut accepter le mauvais type. La RFC 8725 appelle cela la confusion entre JWT et recommande un typage explicite par l'en-tête `typ` ; la RFC 9068 l'applique avec `at+jwt`. Attribuez à chaque catégorie de jetons un type ou un destinataire qui lui soit propre, et écrivez les règles de validation pour que chaque catégorie échoue aux vérifications des autres.

L'en-tête choisit la clé. L'en-tête `kid` indique quelle clé utiliser, et `jku` ainsi que `x5u` peuvent pointer vers une URL où la récupérer. Tous viennent de l'expéditeur. Un `kid` transmis sans contrôle à une requête de base de données ou à une recherche dans un répertoire devient une voie d'entrée dans ce système ; un serveur qui suit toute URL fournie dans `jku` ou `x5u` peut être amené à envoyer des requêtes là où l'expéditeur le souhaite. Ne recherchez `kid` que parmi les clés déjà en votre possession, et ne récupérez des clés que depuis une liste d'emplacements autorisés, comme le `jwks_uri` publié par l'émetteur.

Tout le monde peut lire la charge utile. Un JWT signé protège ses déclarations contre les modifications, pas contre la lecture, et il passe souvent par des navigateurs, journaux et proxies. La RFC 7519 précise qu'un JWT peut contenir des informations sensibles pour la vie privée et que le plus simple est de ne pas les inclure ; sinon, le jeton devrait être chiffré ou au moins transmis uniquement sur TLS. La RFC 9068 ajoute qu'un client ne doit jamais inspecter un jeton d'accès, car l'émetteur peut en changer le format. Placez dans le jeton des identifiants, pas des données personnelles, et chiffrez lorsque le contenu lui-même doit rester privé.

Pour aller plus loin

- RFC 7519 : JSON Web Token (JWT) — le format lui-même, les déclarations enregistrées, l'exemple détaillé et les étapes de validation.
- RFC 7515 : JSON Web Signature (JWS) — encodage et assemblage de l'en-tête, de la charge utile et de la signature, ainsi que la clé de l'exemple ci-dessus.
- RFC 8725 : bonnes pratiques actuelles de JSON Web Token — attaques connues contre JWT et règles des bibliothèques et applications pour les prévenir.
- RFC 7518 : algorithmes web JSON (JWA) — les noms d'algorithmes utilisés dans `alg` et la taille minimale de clé pour chaque HMAC.
- RFC 9068 : profil JWT pour les jetons d'accès OAuth 2.0 — la structure standard des jetons d'accès JWT et les vérifications qu'un serveur de ressources doit effectuer.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le parcours de déploiement, les risques opérationnels et les personnes d'astreinte. Nous pouvons définir un changement avec des vérifications, une procédure de reprise et une responsabilité claire.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce qu'un JWT ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/jwt/