INSTITUT SUPERIEUR D'INFORMATIQUE

ET DES TECHNIQUES DE COMMUNICATION – HAMMAM SOUSSE

المعهد العالي للإعلامية وتقنيات الاتصال بحمام سوسة

JWT (JSON Web Token)

réalisé par :

Yosr Ben Jabeur

1ère année Master Professionnel en Services Web et Multimédia

Responsable du Cours : Mr. Haythem Saoudi

Année universitaire : 2023/2024

Introduction Générale

Dans un monde numérique où la sécurité et l'efficacité sont des enjeux majeurs, le JSON Web Token (JWT) s'impose comme une solution incontournable pour l'authentification et l'autorisation. De plus en plus d'applications web et mobiles distribuées adoptent JWT pour sa simplicité, sa flexibilité et sa robustesse. Ce rapport vise à fournir une compréhension approfondie de JWT, en explorant son fonctionnement, ses avantages, ses cas d'utilisation concrets et les meilleures pratiques pour son implémentation.

Un JWT est un jeton numérique compact et sécurisé composé de trois parties : l'en-tête, la charge utile et la signature. L'en-tête contient des métadonnées sur le jeton, la charge utile contient les données transmises, et la signature garantit l'intégrité et l'authenticité du jeton. Le serveur génère un JWT et le signe avec une clé secrète. Le client stocke le jeton et l'inclut dans chaque requête HTTP au serveur. Le serveur vérifie la signature et extrait les données de la charge utile.

JWT offre de nombreux avantages par rapport aux solutions d'authentification traditionnelles. Il est plus sécurisé, car il est signé numériquement et ne peut pas être modifié sans que la signature ne soit invalidée. Il est également plus compact, ce qui le rend idéal pour une utilisation dans les environnements où la bande passante est limitée. De plus, JWT est flexible et peut être utilisé dans une grande variété de cas d'utilisation, tels que l'authentification des utilisateurs, l'autorisation des API et le partage d'informations entre parties de confiance.

Ce rapport approfondira les aspects techniques de JWT. Il explorera en détail l'en-tête, la charge utile et la signature. Il abordera également les aspects sécuritaires et les bonnes pratiques à suivre pour garantir la sécurité et l'intégrité des jetons JWT. Enfin, il fournira des exemples concrets d'implémentation de JWT dans les applications web et mobiles.

En conclusion, JWT est une technologie prometteuse qui offre de nombreux avantages pour l'authentification et l'autorisation. Ce rapport permettra aux lecteurs de comprendre en profondeur le fonctionnement de JWT et de l'utiliser efficacement dans leurs propres applications.

I. Types de JWT (Stateful vs Stateless) :

La transition des architectures web d'une approche stateful à une approche stateless a été significative dans le développement des applications modernes, notamment en ce qui concerne l'utilisation des JSON Web Tokens (JWT).

La différence fondamentale entre les approches **stateful** et **stateless** réside dans la manière dont les serveurs gèrent les informations sur l'état des sessions utilisateur.

1. Stateful (avec sessions) :

- Les serveurs stockent les informations sur l'état de chaque session utilisateur.
- Ces informations peuvent inclure des données d'authentification, des préférences utilisateur, etc.
- Les sessions sont souvent conservées dans la mémoire du serveur ou dans une base de données associée.
- Les serveurs doivent gérer activement les sessions utilisateur et les données qui y sont associées.

  Une session avec état est créée du côté du backend et l'identifiant de référence de la session correspondante est envoyé au client. Chaque fois que le client adresse une requête au serveur, ce dernier localise la mémoire de session à l'aide de l'identifiant de référence du client et trouve les informations d'authentification.

  Dans ce modèle, vous pouvez facilement imaginer que si la mémoire de session est supprimée du côté du backend, l'identifiant de référence de la session, que le client détient, est complètement dépourvu de sens.

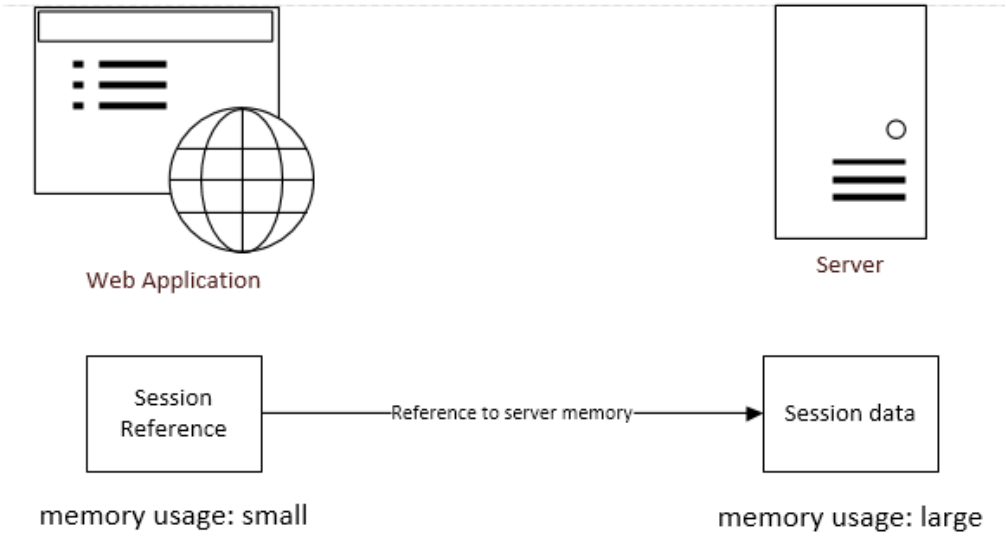


Figure 1:Modèle Stateful

2. Stateless (sans sessions) :

- Les serveurs ne stockent pas les informations sur l'état de chaque session utilisateur.
- Au lieu de cela, toutes les informations nécessaires pour identifier et autoriser l'utilisateur sont contenues dans le token lui-même.
- Les tokens, tels que les JSON Web Tokens (JWT), sont autoportants, ce qui signifie qu'ils contiennent toutes les informations nécessaires dans leur structure.
- Les serveurs n'ont pas besoin de conserver les sessions utilisateur, ce qui simplifie la gestion et réduit la charge de stockage et de traitement.

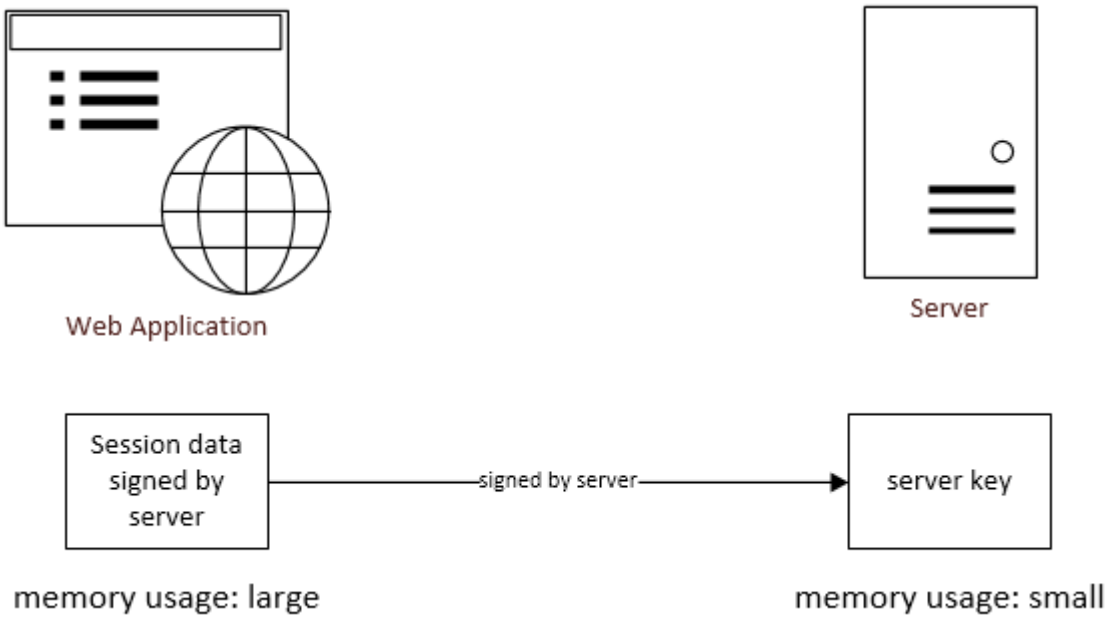


Figure 2:Modèle Stateless

II. Structure et contenu d'un JWT(Stateless)

L'utilisation des JWT a contribué à rendre les architectures web plus stateless, en éliminant la nécessité de stocker des sessions utilisateur sur les serveurs.

- Un JWT est généralement composé de trois parties séparées par des points ('.').
- Les trois parties sont l'en-tête (header), la charge utile (payload) et la signature.
- La forme générale d'un JWT est : <header>.<payload>.<signature>

3. En-tête (Header)

C'est un JSON encodé en base64 contient des métadonnées sur le jeton, comme l'algorithme de signature utilisé et le type de jeton.

- **alg**: Algorithme de signature utilisé (par exemple, HS256, RS256)
- **typ**: Type de jeton (par exemple, "JWT")
- **cty**: Type de contenu de la charge utile (optionnel)

4. Charge utile (Payload) :

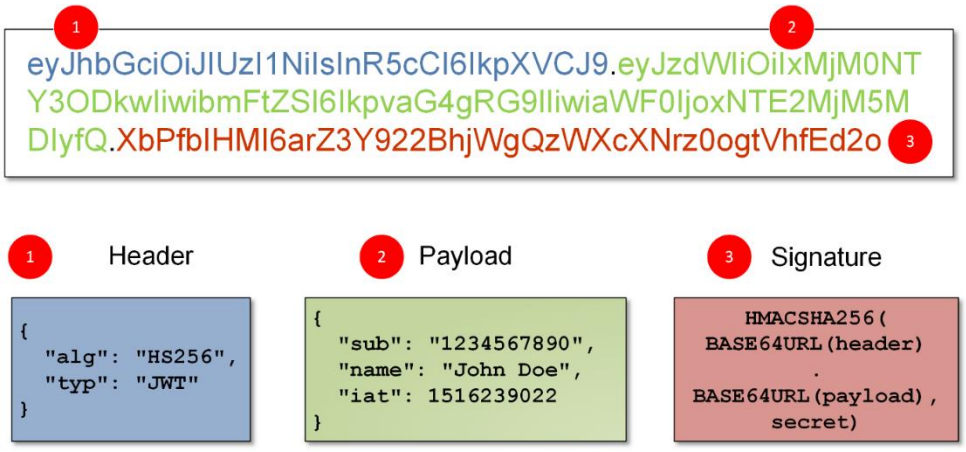
Contient les données que le jeton est censé transmettre. Elles sont généralement sous forme de paires clé-valeur JSON encodé en base64 qui contient les informations à transmettre. Ce JSON peut contenir des clefs prédéfinies (appelées claims) qui permettent de donner des informations supplémentaires sur le token (comme la date d'expiration, la cible, le sujet...).

- **ss**: Émetteur du jeton (par exemple, le nom du serveur)
- **sub**: Sujet du jeton (par exemple, l'identifiant de l'utilisateur)
- **aud**: Destinataire du jeton (par exemple, l'identifiant de l'application)
- **exp**: Date d'expiration du jeton
- **iat**: Date d'émission du jeton
- **nbf**: Date avant laquelle le jeton ne doit pas être accepté (optionnel)
- **jti**: Identifiant unique du jeton (optionnel)
- **Données spécifiques à l'application :** Champs supplémentaires définis par l'application

5. Signature :

Est la partie la plus importante du token JWT car elle permet de s'assurer de l'authenticité du token. Cette signature est générée à partir des deux premières clefs avec un algorithme particulier (HS256 par exemple avec une clef secrète).

Exemple :



III. Utilisation:

JWT peut être utilisé dans une grande variété de cas d'utilisation, notamment :

1. Authentification des utilisateurs :

- Authentification des utilisateurs dans les applications web et mobiles
- Implémentation de Single Sign-On (SSO)
- Authentification des API

2. Autorisation des ressources :

- Définir les droits d'accès des utilisateurs aux ressources
- Contrôler l'accès aux API
- Protéger les données sensibles

3. Partage d'informations :

- Partager des informations entre parties de confiance
- Échanger des données entre applications
- Mettre en œuvre des architectures microservices

4. Suivi des utilisateurs :

- Suivre les sessions des utilisateurs
- Analyser le comportement des utilisateurs
- Personnaliser l'expérience utilisateur

5. Implémentation de workflows :

- Gérer les processus métier
- Définir des règles d'autorisation pour les tâches
- Suivre l'avancement des tâches

IV. Avantages

- Sécurité: Les JWT sont signés numériquement, ce qui garantit l'intégrité et l'authenticité des données.
- Compacité: Les JWT sont compacts et peuvent être facilement transmis dans une URL ou un en-tête HTTP.
- Flexibilité: Les JWT peuvent être utilisés dans une grande variété de cas d'utilisation.
- Simplicité: Les JWT sont faciles à implémenter et à utiliser.

V. Limites de l'utilisation de JWT

- Taille : Les JWT ont une taille limite, ce qui peut limiter la quantité de données qu'ils peuvent contenir.
- Révocation : Il n'est pas possible de révoquer un JWT une fois qu'il a été émis.
- Sécurité de la clé secrète : La sécurité des JWT dépend de la sécurité de la clé secrète utilisée pour les signer.

VI. Sécurité des tokens (encodage, hashage, cryptage) :

1. Encodage :

- L'encodage est le processus de conversion de données dans un format spécifique, souvent utilisé pour la transmission ou le stockage de données.
- Il ne sécurise pas les données mais facilite leur transmission en convertissant les caractères spéciaux en une forme acceptée par les protocoles de communication.
- Par exemple, Base64 est une méthode couramment utilisée pour encoder des données binaires en texte ASCII.

2. Hashage :

- Le hashage est un processus qui convertit des données de taille variable en une chaîne de longueur fixe, appelée hachage, à l'aide d'une fonction de hachage.

- Les fonctions de hachage sont unidirectionnelles, ce qui signifie qu'il est difficile de retrouver les données d'origine à partir du hachage.
- Les hachages sont utilisés pour vérifier l'intégrité des données, stocker les mots de passe de manière sécurisée et générer des identifiants uniques.

3. Cryptage :

- Le cryptage est le processus de conversion de données en un format illisible (chiffré) à l'aide d'une clé de chiffrement.
- Contrairement au hashage, le cryptage est réversible, ce qui signifie que les données chiffrées peuvent être déchiffrées en utilisant la clé appropriée.
- Il est largement utilisé pour assurer la confidentialité des données sensibles pendant le stockage ou la transmission, notamment dans les protocoles SSL/TLS pour sécuriser les communications sur Internet.

VII. Types d'attaques et sécurité de JWT

Les attaques CSRF (Cross-Site Request Forgery) et XSS (Cross-Site Scripting) sont deux types de vulnérabilités de sécurité courantes rencontrées dans les applications web. Bien qu'ils partagent certains traits, ils se distinguent par leurs mécanismes d'attaque et les conséquences pour les utilisateurs et les applications.

1. CSRF VS XSS :

Caractéristique	**CSRF (Cross-Site Request Forgery)**	**XSS (Cross-Site Scripting)**
Mécanisme	Exploite l'authentification de l'utilisateur sur un site pour effectuer des actions non désirées sur ce site.	Exploite les failles dans le traitement des données sur le site pour injecter et exécuter du code malveillant sur les navigateurs des utilisateurs
Cible	Utilisateur authentifié sur un site web	Visiteurs du site web, généralement les utilisateurs non authentifiés
Impact	Modification non autorisée des données de l'utilisateur, exécution d'actions indésirables en son nom.	Redirection vers des sites malveillants, vol de données sensibles des utilisateurs, exécution de scripts nuisibles sur le navigateur de l'utilisateur.
Source de l'attaque	Un site malveillant ou compromis peut envoyer des requêtes forgées à un site cible au nom de l'utilisateur authentifié.	Les données non sécurisées fournies par l'utilisateur sont incorporées dans le code HTML retourné par le serveur, permettant l'injection et l'exécution de code malveillant.
Méthodes de prévention	Utilisation de jetons anti-CSRF (CSRF tokens), vérification de l'en-tête Referer, validation des requêtes côté serveur.	Filtrage et validation des données d'entrée, échappement des caractères spéciaux, utilisation de CSP (Content Security Policy), mise à jour régulière des frameworks et bibliothèques utilisées

2. Utilisation de HMAC

Dans le contexte de JSON Web Tokens (JWT), HMAC (Hash-based Message Authentication Code) joue un rôle essentiel dans la sécurisation de l'intégrité des tokens. HMAC est un algorithme de cryptographie qui garantit l'authenticité et l'intégrité des données échangées entre le client et le serveur.

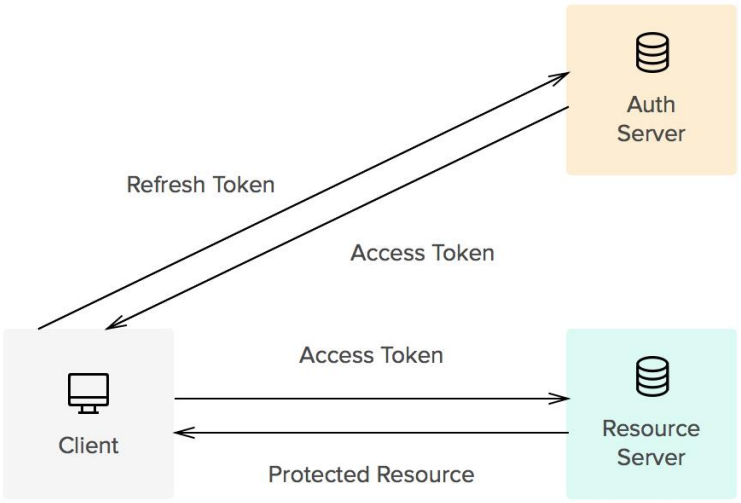
- Fonctionnement de HMAC : HMAC combine une clé secrète avec un algorithme de hachage cryptographique pour produire un code d'authentification de message. Ce code est ensuite joint au message et transmis au destinataire. À réception, le destinataire réapplique la fonction HMAC avec la même clé et vérifie si le code reçu correspond à celui recalculé localement.

- Sécurisation des signatures JWT :Dans JWT, la signature est générée en utilisant HMAC avec une clé secrète connue uniquement du serveur. Cette signature est basée sur le contenu du Header et du Payload du JWT, et est utilisée pour vérifier l'intégrité des données lors de la transmission.
- Avantages de l'utilisation de HMAC dans JWT :Sécurité renforcée : HMAC assure que les tokens JWT n'ont pas été altérés entre le moment où ils ont été émis et le moment où ils sont reçus.
- Protection contre la falsification : Les signatures HMAC rendent extrêmement difficile pour un attaquant de modifier les données du token sans que cela soit détecté.
- Clé secrète unique : La sécurité de HMAC dépend fortement de la sécurité de la clé secrète. Il est donc essentiel de générer et de stocker cette clé de manière sécurisée.

VIII. Accès Token et Refresh Token

Aspect	Access Token	Refresh Token
Fonctionnement	-Utilisé pour accéder aux ressources protégées contient des informations sur l'utilisateur et ses autorisations.	-Utilisé pour obtenir un nouvel Access Token -Utilisé pour prolonger la validité de la session de l'utiulisateur.
Mécaniseme de sécurité	Généralement court et renouvelé fréquemment Utilisation de signatures numériques pour assurer l'intégrité et l'authenticité	Plus long que l'Access Token et a une durée de vie plus longue, souvent associée à une politique de rafraîchissement
Gestion des Permissions et des Autorisation	Contient des informations sur les autorisations de l'utilisateur, telles que les rôles et les privilèges	N'a généralement pas de revendications d'autorisation
Renouvellement et Révocation des Tokens	Ne peut pas être rafraîchi directement, nécessite un nouveau processus d'authentification	Utilisé pour obtenir un nouveau Access Token lorsqu'il expire, évitant ainsi à l'utilisateur de se reconnecter
nteropérabilité et Standards	Impliqué dans des protocoles d'authentification modernes comme OAuth 2.0 et OpenID Connect	Souvent utilisé avec des protocoles tels que OAuth 2.0

Exemples d'Utilisation	Utilisé dans les applications mobiles, les services web et les API pour sécuriser les échanges de données	Joue un rôle essentiel dans la gestion des sessions utilisateur dans divers contextes applicatifs



IX. Comparaison entre les tokens JWT et les cookies sécurisés

Dans le paysage de l'authentification et de l'autorisation des utilisateurs, deux approches principales émergent : l'utilisation de JSON Web Tokens (JWT) et celle des cookies sécurisés. Chacune de ces méthodes présente des caractéristiques uniques ainsi que des avantages et des inconvénients spécifiques.

1. Tokens JWT

Les JSON Web Tokens (JWT) sont des tokens auto-contenus qui contiennent des informations sur l'utilisateur et des données de session.

- **Avantages :**

-Stateless : Les tokens JWT sont stateless, ce qui signifie que le serveur n'a pas besoin de stocker l'état de session côté serveur. Cela réduit la charge du serveur et améliore la scalabilité.

-Flexibilité : Les tokens JWT peuvent contenir des informations arbitraires sous forme de charges utiles (payloads), ce qui les rend flexibles pour divers cas d'utilisation.

-Cross-Domain : Les tokens JWT peuvent être utilisés dans des environnements cross-domain, ce qui les rend appropriés pour les architectures distribuées et les services d'API.

- **Inconvénients :**

-Sécurité dépendante de la mise en œuvre : La sécurité des tokens JWT dépend largement de la façon dont ils sont mis en œuvre, y compris la génération, le stockage et la validation des tokens.

-Taille des tokens : Les tokens JWT peuvent devenir volumineux s'ils contiennent trop d'informations, ce qui peut augmenter la taille des requêtes HTTP.

2. Cookies sécurisés

Les cookies sécurisés sont des fichiers texte stockés dans le navigateur de l'utilisateur et envoyés automatiquement avec chaque requête vers le serveur. Voici quelques points clés concernant les cookies sécurisés :

- **Avantages :**

-Simplicité d'utilisation : Les cookies sécurisés sont gérés automatiquement par le navigateur, ce qui simplifie leur utilisation pour les développeurs.

-Contrôle côté serveur : Les cookies peuvent être manipulés côté serveur, ce qui permet un contrôle plus granulaire sur les sessions utilisateur.

-Gestion des politiques de sécurité : Les cookies sécurisés peuvent être configurés avec des politiques de sécurité strictes, telles que HTTPOnly et Secure, pour renforcer la sécurité.

- **Inconvénients :**

-Complexité de gestion de l'état : Les cookies sécurisés nécessitent une gestion de l'état côté serveur, ce qui peut entraîner une complexité accrue, en particulier dans les applications distribuées.

-Sensibilité aux attaques : Les cookies peuvent être sujets à des attaques telles que le vol de session et les attaques de falsification de requête intersite (CSRF) s'ils ne sont pas correctement sécurisés.

X. Stockage des JWT

L'utilisation de JSON Web Tokens (JWT) pour le stockage des informations liées à l'authentification et à l'autorisation présente plusieurs avantages et considérations importantes.

1. Stockage local côté client

Les tokens JWT peuvent être stockés localement côté client, généralement dans le navigateur web, afin de maintenir l'état de session de l'utilisateur entre les requêtes. Les méthodes de stockage local incluent :

- Stockage dans le local storage : Le local storage offre une capacité de stockage plus grande que les cookies et persiste même après la fermeture du navigateur. Cependant, il est vulnérable aux attaques XSS (Cross-Site Scripting) si les données sensibles y sont stockées sans précaution.

- Stockage dans les cookies : Les cookies peuvent être utilisés pour stocker les JWT, offrant ainsi une compatibilité avec les requêtes HTTP et permettant de spécifier des options de sécurité telles que HTTPOnly et Secure pour renforcer la sécurité. Cependant, les cookies sont limités en termes de taille de stockage et peuvent être vulnérables aux attaques de vol de session si mal configurés.

2. Contenu du JWT

Les JWT stockent des informations d'identification et d'autorisation sous forme de charges utiles (payloads) encodées en base64url. Les informations stockées peuvent inclure :

- Identifiant de l'utilisateur
- Rôles et autorisations
- Durée de validité (expiration)
- D'autres métadonnées pertinentes

3. Gestion de l'expiration

Les JWT peuvent être configurés avec une date d'expiration pour limiter leur validité dans le temps. Une fois expirés, les JWT ne sont plus considérés comme valides et doivent être renouvelés ou réémis pour continuer à accéder aux ressources protégées.

4. Sécurité et bonnes pratiques

Il est essentiel de mettre en œuvre des pratiques de sécurité appropriées lors du stockage et de la gestion des JWT :

- Chiffrement des données sensibles : Les données sensibles stockées dans les JWT doivent être chiffrées pour éviter toute exposition en cas de vol de token.

- Rotation des clés : La rotation régulière des clés secrètes utilisées pour signer les JWT renforce la sécurité et réduit les risques liés à la compromission des clés.
- Gestion des revocations : En cas de besoin de révoquer un JWT avant son expiration, un mécanisme de gestion des révocations peut être mis en place pour maintenir l'intégrité du système.

Conclusion

En conclusion, JSON Web Tokens (JWT) simplifient l'authentification et l'autorisation dans les applications web et les services API grâce à leur structure stateless et leur signature HMAC assurant l'intégrité des données. Leur modularité permet une personnalisation étendue, mais leur sécurité nécessite une gestion appropriée des clés secrètes et une validation rigoureuse des tokens. En comprenant leurs avantages et défis, les développeurs peuvent exploiter pleinement JWT pour créer des applications sécurisées et évolutives.