CUBE
Plan de sécurisation API

Juin 2022

Mails et chiffrement des identifiants

Vérification de l'adresse mail

Lors de la création d'un compte utilisateur, une confirmation par e-mail est effectuée avec un token haché permettant de vérifier l'identité de l'utilisateur et d'éviter la création massive de comptes via l'utilisation de scripts.

Hachage des mots de passe

Lors de la création de compte, le mot de passe utilisateur est haché grâce à l'algorithme SHA512 et une clef secrète sur le serveur modifiée régulièrement. En base de données ne sont stocké que le résultat du hachage ainsi qu'un « sel » unique supplémentaire à chaque utilisateurs (une chaîne de caractères générée aléatoirement nécessaire au déchiffrage des mots de passes).

Il suffit ensuite de hacher à nouveau les mots de passes utilisateurs avec la même procédure afin de vérifier la concordance des données. Ainsi, si le serveur est compromis par une cyber-attaque, les mots de passes ne peuvent pas être rattachés à leurs identifiants respectifs.

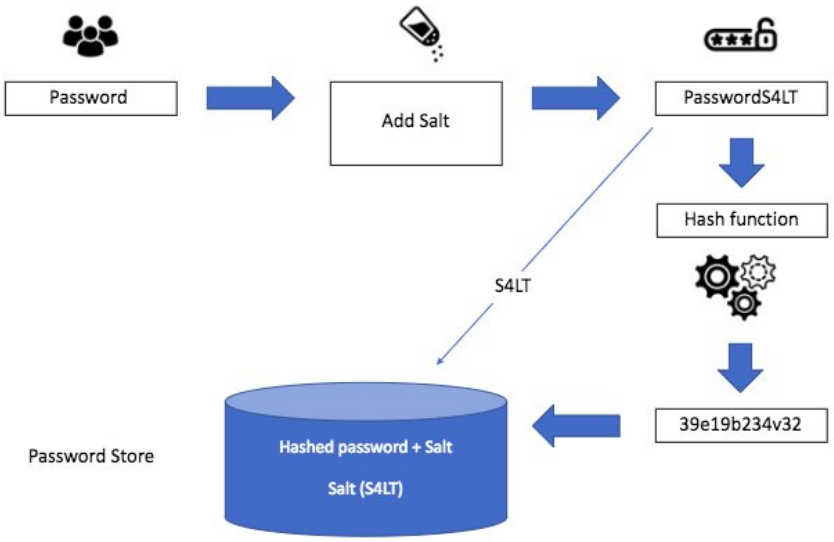

Figure 1- Schéma d'un système classique de hachage avec sel

Token et permissions

Token JWT

A chaque interaction de l'utilisateur après son authentification, un JWT (JSON Web Token) est lui est envoyé ; celui-ci doit être renvoyé à chaque requête qui nécessite un droit d'utilisateur connecté.

Le JWT est composé de 3 parties : le « header » qui permet d'en connaître le type, le « payload » qui contient les informations utilisables par le serveur, et enfin le contenu du token haché qui permet de vérifier que le token est authentique et bien produit par le serveur.

Le payload du token contient trois informations actuellement :

- L'identité de l'utilisateur (via l'ID renseigné en base de données)
- La date d'expiration du token (augmente la sécurité car crypter ou decrypter un token prend tu temps)
- Le niveau de permission de l'utilisateur

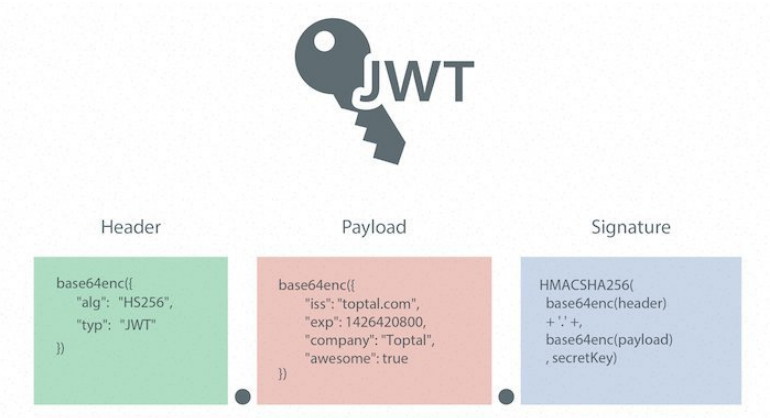

Figure 2 Exemple d'utilisation d'un token JWT

Système de permission

Le système de permission comporte quatre catégories d'utilisateur : Non-connecté, utilisateur, modérateur et Administrateur. Les requêtes peuvent nécessiter un de ces niveau d'autorisation, et le traitement des données et la réponse du serveur peut changer en fonction de celui-ci.

Mises à jour des dépendances

Les dépendances du serveur peuvent facilement devenir une source de failles majeures dans la sécurité des serveurs, au fur et à mesure que des failles sont trouvées dans les dépendances de nouvelle mises à jour.

Le code étant hébergé sur GitHub, le service d'analyse du code automatique fournie par le site permet d'alerter en cas de failles majeures et offre une première barrière de sécurité.

Autre outil permettant d'assurer un code à jour : Dependabot présent nativement dans le service GitHub permet de proposer automatiquement une mise à jour des paquets.

Pour encore réduire les chances d'attaque, un système de reverse proxy et une configuration du serveur qui empêche d'accéder facilement aux propriétés du serveur est à concevoir.

Intégrité des données et bonne pratique de code

Enfin, en plus des outils et des méthodes permettant de limiter la vulnérabilité du système, la bonne pratique lors de l'écriture du code est essentielle.

Le système de validation des entrée et sortie de donnée au sein des contrôleurs permet d'empêcher tout dysfonctionnement interne qui pourrait compromettre le fonctionnement attendu de l'API. Pour se faire, chaque objet prévu pour intégrer la base de données est strictement vérifié, de par les attributs qui doit comprendre mais aussi leur type et leurs limites.

De même pour les réponses données aux requêtes, notamment si elle comporte des messages d'erreurs, il faut certes offrir une qualité de débogage et d'information aux utilisateurs, mais ne pas trop dévoiler le fonctionnement interne de l'api, cela pourrait offrir une aide précieuse au cyber- attaquant en quête de failles.