Jeton JWT (pratique)

|  |  |
| --- | --- |
| Elément | Description |
| **Compétences** | Pouvoir mettre en œuvre des mécanismes d’authentification et d’autorisation. |
| **Objectifs** | A la fin de cette séquence, les apprentis seront capables   * De créer une session à l’aide d’un token JWT (symétrique et asymétrique) |
| **Durée estimée** | 45 min |
| **Répertoire de travail** |  |
| **Fichiers sources** | Nodejstoken.zip |
| **A produire** | Répondre aux questions directement dans ce document |
| **Moyens d’aide** |  |
| **Changelog** | Auteur original et version |

# Mise en place de l’environnement

Suivez les étapes :

1. (Si ce n’est pas déjà fait) Faites un backup de votre application nodejstoken
2. Eteignez et supprimer les containers nodejstoken
3. Purgez les images actuelles de Docker à l’aide de **docker system prune -a**
4. **Extraire nodejstoken.zip dans C:\docker\nodejstoken**
5. Montez votre stack à l’aide de la commande docker compose up -d

# Signature de jeton avec un chiffrement symétrique

Modifier le code fourni de telle sorte à inclure la signature d’un jeton JWT selon les contraintes suivantes :

## Jeton JWT + HS256 + expiration dans 1h

Notez ci-dessous l’extrait de code correspondant

      const token = jwt.sign({ userId: rows.idUser }, private\_key, {

        expiresIn: "1h",

        algorithm: "HS256"

      });

Prouvez par un screenshot que le jeton obtenu correspond bien à cette définition (indice vous aurez besoin d’un site de validation de token).

Quels sont les champs du token qui sont relatifs au timestamp ? Quel est leur format ?

Le champ expiresIn. ………………………………………………………………………..

## Jeton JWT + HS512 + expiration dans 1h

Notez ci-dessous l’extrait de code correspondant à la génération de votre token :

      const token = jwt.sign({ userId: rows.idUser }, private\_key, {

        expiresIn: "1h",

        algorithm: "HS512"

      });

# Protéger la route /user

Modifiez le contenu du fichier UserController.mjs de telle sorte à ce que l’utilisateur spécifié dans le jeton soit retourné (success http : 200 + body contenant le nom d’utilisateur du token).

Si le token n’est pas fourni, vous retournerez une erreur 40x (à vous de choisir la bonne).

Si le token est fourni, vous le testerez et en cas d’erreur d’authentification, retournez une erreur 40x (à vous de choisir la bonne). (indice : vous utiliserez un try…catch)

## Protection de la clé secrète

Attention, vous avez certainement écrit votre clé secrète directement dans le code. Faites en sorte que la clé secrète soit externe au code source de votre application.

Quand c’est terminé, montrez votre implémentation à votre enseignant.

# Signature de jeton avec un chiffrement asymétrique

## Génération d’une paire de clé (privée / publique)

Avec OpenSSL, (<https://www.cryptool.org/en/cto/openssl>), générez une paire de clé compatible RS256 et exportez ces clés vers votre ordinateur.

Quelles sont les commandes que vous avez utilisées ?

Décrivez les paramètres de ces commandes.

## Implémentation du token RS256 avec validité de 1h

Modifiez maintenant votre application de telle sorte à ce que le jeton JWT soit maintenant basé

# Aide sur Thunder Rest Client

Le terme **bearer authentification** peut être compris comme « Donne l’autorisation au porteur du token ». Le bearer est un schéma d’authentification.

Avec Thunder Rest Client, il est possible de choisir un schéma d’authentification. Pour JWT, nous choisirons donc **Bearer** (porteur).

A screenshot of a computer

Description automatically generated

# Pour les plus rapides

Maintenant que votre authentification et vos sessions sont en place, créez avec Vue.js une page de login pour que l’utilisateur puisse saisir son nom d’utilisateur et son mot de passe.

Vous mettrez aussi en place une page user pour afficher le nom de l’utilisateur connecté (ou une erreur si l’utilisateur n’est pas authentifié).