TP3 : développement avancé.

RABIARIVELO Ilo Andrianaly 303

Etape 1:

Après avoir initialisé le dépôt git j'ai lu le sujet, j'ai ouvert postman et j'ai essayé d'accéder au lien demandé dans l'énoncé.

Pour "http://localhost:3000/secu" j'obtiens

```
1  {"replique":"Tu ne sais rien, John Snow.."}
```

C'est un message d'erreur lorsque la connexion d'un identifiant est refusée.

Pour "http://localhost:3000/dmz" j'obtiens

```
1  {
2      "replique": "Ca pourrait être mieux protégé..."
3  }
```

Après avoir encodé "Tyrion:wine" en base 64 et l'avoir mis dans la valeur d'Autorization précédé d'un Basic, j'ai retenté d'accéder au lien "http://localhost:3000/secu" pour voir si j'obtenais toujours le message d'erreur ou bien si j'avais réussi à me connecter et j'ai obtenu le bon message de connexion.

```
{
    "replique": "Un Lannister paye toujours ses dettes !"
}
```

La fonction after() est invoquée après que tous les plugins ont fini de s'exécuter. Donc on peut en déduire que dans le cas de "/secu", elle exécute son code après la bonne authentification de l'utilisateur.

Après la création de la route "/autre", j'ai mis cette réplique là dans le bloc after() et j'ai enlevé le champ "onRequest : fastify.basicAuth" et j'ai obtenu la bonne réplique.

```
{
    "replique": "Celui qui est déjà mort ne peut pas mourir à nouveau!"
}
```

Etape 2 :

Pour l'utilisation d'openssl, j'ai utilisé la VM fournie par la clef USB car je suis à l'IUT. J'ai créé un nouveau fichier "TP4" et ensuite tapé la commande suivante.

```
openssl genrsa -out server.key 2048
```

Une fois que la clé a été initialisée, j'ai effectué la création du certificat et je l'ai autosigné avec ma clé privée.

```
linuxetu@linuxetu:~/TP4$ openssl req -new -key server.key -out server.req
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:Paris
Locality Name (eg, city) []:Paris
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IUT de Paris
Organizational Unit Name (eg, section) []:303
Common Name (e.g. server FQDN or YOUR name) []:monserveur.fr
Email Address []:rabiariveloilo@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:armanturc123
An optional company name []:Lcorp
linuxetu@linuxetu:~/TP4$ openssl x509 -req -days 365 -in server.req -signkey server.key -out server.
crt
```

J'ai ensuite démarré le server sur le port 4567 avec la commande suivante :

```
linuxetu@linuxetu:~/TP4$ openssl s_server -accept 4567 -cert server.crt -key server.key -www -state
```

Et j'ai ensuite accédé au site avec à la place de "localhost", "192.169.10.128" et avec https à l'avant. et j'ai obtenu le code HTML de la page :

```
<HTML><BODY BGCOLOR="#ffffff">
<pre>
```

Après, j'ai créé la clef publique à partir de ma clef privée avec la commande adéquate et je l'ai par la suite copié depuis le powershell avec la commande "SCP".

```
scp linuxetu@192.168.10.128:TP4/server.pub .
```

Et j'ai aussi récupéré de la même manière la clef privée et le certificat.

```
PS Z:\> scp linuxetu@192.168.10.128:TP4/server.key .
linuxetu@192.168.10.128's password:
server.key                                    100% 1704    57.8KB/s   00:00
```

```
PS Z:\> scp linuxetu@192.168.10.128:TP4/server.crt .
linuxetu@192.168.10.128's password:
server.crt                                    100% 1338    42.5KB/s   00:00
```

Et avec la documentation de Fastify, j'ai ensuite modifier le service web pour qu'il utilise ces nouveaux ustensiles en changeant sa définition avec la mention http2 et l'ajout de la clef privée et du certificat.

J'ai aussi fait la découverte de SSH et SCP pour la copie de fichiers dans la machine virtuelle.

Etape 3 :

Pour commencer cette étape j'ai git clone le repo marqué dans l'énoncé et je l'ai ensuite copié une deuxième fois pour avoir deux projet auth et data.

Il fallait que je crée des nouvelles clefs compatibles avec JWT j'ai donc tapé la commande suivante permettant la nouvelle génération.

```
openssl ecparam -genkey -name prime256v1 -out .ssl/private_key.pem
openssl ec -in .ssl/private_key.pem -pubout -out .ssl/public_key.pem
```

Après j'ai rajouté le chemin vers ma clef publique et ma clef privée dans le jwt.js

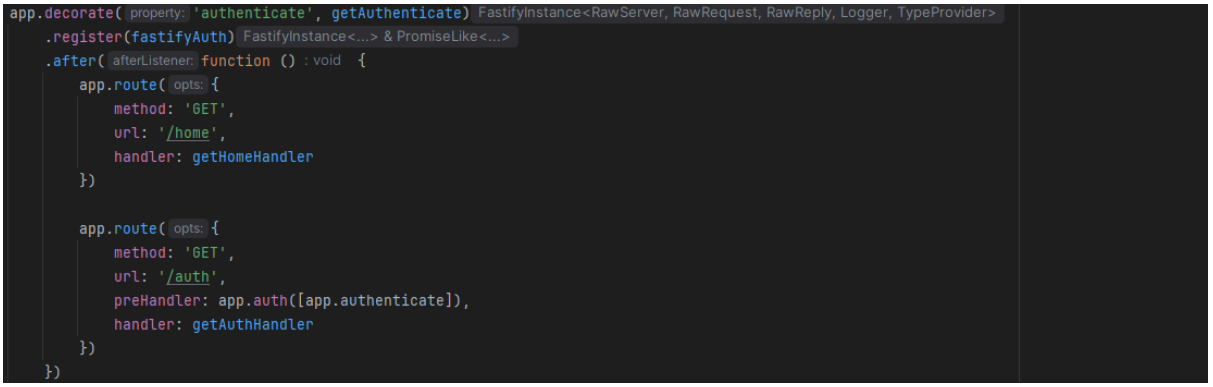
```
    secret: {
        allowHTTP1 : true,
        private : fs.readFileSync(path.join(__dirname,"..","..",".ssl","private-key.pem")),
        public : fs.readFileSync(path.join(__dirname,"..","..",".ssl","public-key.pem")),
    }
```

Et j'ai commencé à explorer le projet, j'ai commencé à remplir les méthodes loginUser et addUser.

Dans la méthode pour ajouter un nouvel utilisateur j'ai créé une méthode permettant de choisir aléatoirement le rôle admin ou simple utilisateur.

```
function randomizer(choix1, choix2) : any  {
    const randomNum : number  = Math.random();
    if (randomNum < 0.5) {
        return choix1;
    } else {
        return choix2;
    }
}
```

Et ensuite j'ai attaqué la création de la partie data. J'ai d'abord initialisée les routes en utilisant la fonction getAuthenticate codée dans le dossier middleware pour décorer les handler getHomeHandler et getAuthHandler. Et après j'ai codé ces méthodes dans le fichier data.js.



Les deux services “auth” et “data” devrait être fonctionnel sur le port 3000 et 4000.

Conclusion :

Pour conclure, j’ai trouvé les 2 premières étapes de ce TP assez simple. Mais à partir de la troisième étape j’ai rencontré un peu plus de difficulté avec l’utilisation de JWT et de comment gérer les tokens.

Dans l’ensemble je pense avoir fini le TP à 100% et je suis plutôt fier du travail que j’ai fourni. Pour une fois j’ai réussi à faire une grosse partie des étapes en cours.