Florent Lelion - 302 15/02/2024

Compte rendu - Sujet 4
Authentification et Autorisation

Étape 1 – « De base… »

Je commence par accéder aux end-points http://localhost:3000/secu et http://localhost:3000/dmz à l'aide de Postman, qui me renvoient chacun respectivement deux répliques différentes :

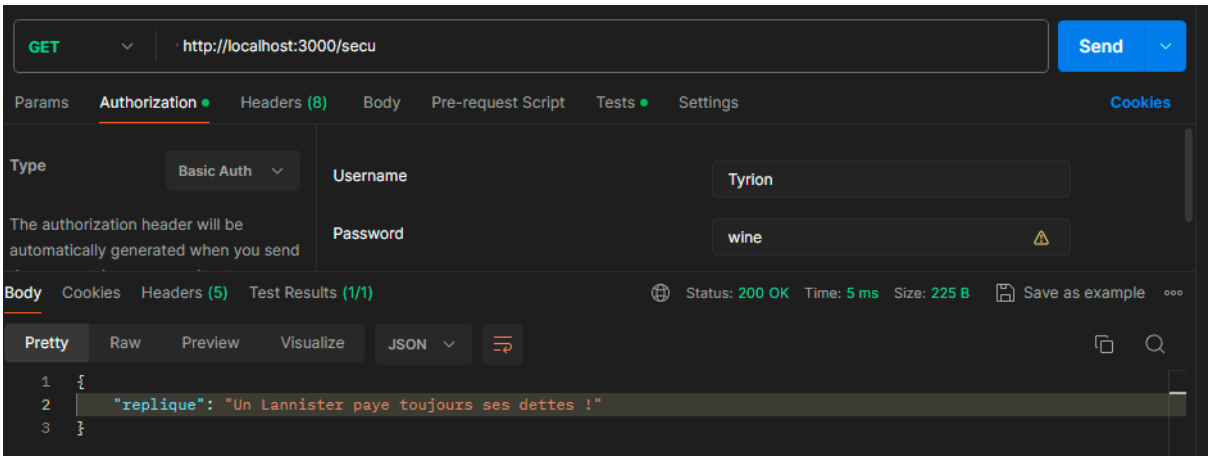
```
{
    "replique": "Tu ne sais rien, John Snow.."
}
```

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

Par la suite je me suis documenté sur l'authentification en base64 afin de mieux comprendre. J'ai suivi les étapes en encodant Tyrion:wine en base64 qui m'a donné VHlyaW9uOndpbmU=. J'ai ensuite ajouté Basic VHlyaW9uOndpbmU= dans la clé Authorization puis j'ai relancé l'end-points http://localhost:3000/secu qui m'a affiché :

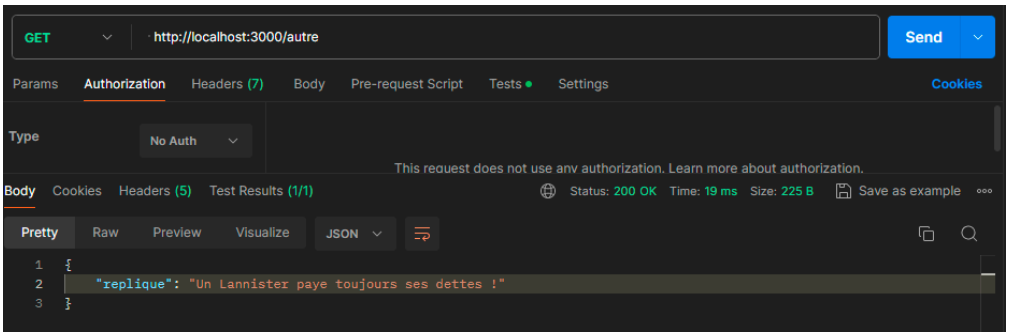
```
{
    "replique": "Un Lannister paye toujours ses dettes !"
}
```

Pour l'étape suivante j'ai ajouté les identifiants username et password en claire et j'obtiens bien la même réplique :



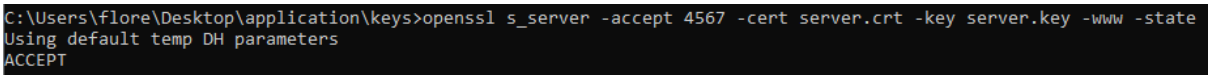
La fonction after() est utilisée pour vérifier que l'utilisateur passe bien par le Fastify Basic Auth (c'est à dire qu'il valide authentification et validate (qui vérifie le username et password)) et avant d'accéder à la route /secu.

J'ai dupliqué le code fastify.route({}) en changeant la route en /autre et en retirant la sécurité d'authentification onRequest: fastify.basicAuth. Maintenant quand je lance l'endpoint http://localhost:3000/secu sans authentification j'obtiens la bonne réplique :

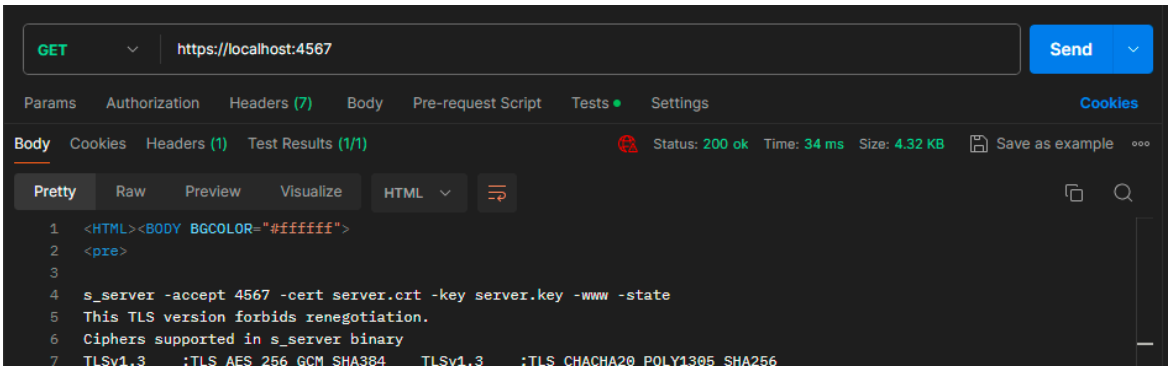


Étape 2 – Prouves qui tu es !

J'ai tout d'abord exécuté la commande *openssl genrsa -out server.key 2048* dans le OpenSSL Command Prompt qui m'a généré un fichier server.key dans mon dossier. Ensuite j'ai exécuté la commande *openssl req -new -key server.key -out server.csr* pour créer le certificat server.csr. Ensuite j'ai signé le certificat avec la clé privée à l'aide de la commande suivante *openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt*. Enfin, j'ai pu tester le certificat généré qui me renvoie bien ACCEPT :



Je test l'url https://localhost:4567 dans Postman qui me renvoie la réponse suivante :



Dans la réponse, *Verify return code: 0 (ok)* signifie que le certificat a bien été vérifié.

Je me suis inspiré de la documentation et j'ai rajouté ce code que j'ai adapté (key et cert) pour que l'on puisse accéder au service web en https :

```
const fastify : FastifyInstance<...> & PromiseLike<...>  = Fastify( opts: {
    logger: true,
    http2: true,
    https: {
        allowHTTP1: true,
        key: fs.readFileSync('./src/server.key'),
        cert: fs.readFileSync('./src/server.crt')
    }
})
```

Étape 3 – Un jeton dans la machine

Pour cette étape, j'ai commencé par me diriger dans le dossier .ssl du projet avec OpenSSL pour générés les clés de chiffrements (clé privée et publique) à l'aides des commandes suivantes :

```
C:\Users\flore\Desktop\application\R6.A.05-TP-Secu1-auth\.ssl>openssl ecparam -genkey -name secp256k1 -out ecdsa.key

C:\Users\flore\Desktop\application\R6.A.05-TP-Secu1-auth\.ssl>openssl ec -in ecdsa.key -pubout -out ecdsa.pub
read EC key
writing EC key
```

Ensuite pour compléter l'enregistrement du plugin fasitfyJwt, je suis allé voir la documentation puis j'ai complété le code du fichier jwt.js pour que le plugin configure Fastify pour utiliser les JWT signés avec l'algorithme ES256 en utilisant les clés privée et publique.

Pour continuer, j'ai complété le handler addUser() pour qu'il crée un nouveau user et qu'il le rajoute à la liste (dans la méthode addUser) :

```
const newUser : {...}  = {
    email,
    password: hashedPassword,
    role: 'utilisateur'
};

users.push(newUser);
```

Concernant la méthode loginUser, je la complète pour qu'elle reçoive les informations d'identification de l'utilisateur depuis la requête HTTP, puis vérifie leur validité, puis génère un jeton JWT si les informations sont correctes, et si ce n'est pas le cas elle renvoie un message d'erreur :

```
export const loginUser = async function (req, res) : Promise<...>  {
    const { email, password } = req.body;

    const user = users.find((u) : boolean  => u.email === email);

    if (!user) {
        return res.status(401).send({ message: "Utilisateur non trouvé" });
    }

    if (user.password !== hashedPassword) {
        return res.status(401).send({ message: "Mot de passe incorrect" });
    }

    const token = generateJWT(user);

    res.status(200).send({ token });
}
```