COMPTE RENDU
TP4

Cours :

Développement avancé - Laurent Giustignano

Fait par :

Alexia Bence 304

SOMMAIRE

INTRODUCTION

L'objectif de ce TP est d'améliorer le niveau de sécurité de nos services web en passant par différentes étapes.

Dans un premier temps, nous nous concentrerons sur l'implémentation d'une authentification basique des utilisateurs. Pour ce faire, nous explorerons l'utilisation d'un plugin proposé par Fastify, un framework web, qui facilite cette tâche. Vous apprendrez à mettre en place ce mécanisme d'authentification et à l'intégrer à vos services existants.

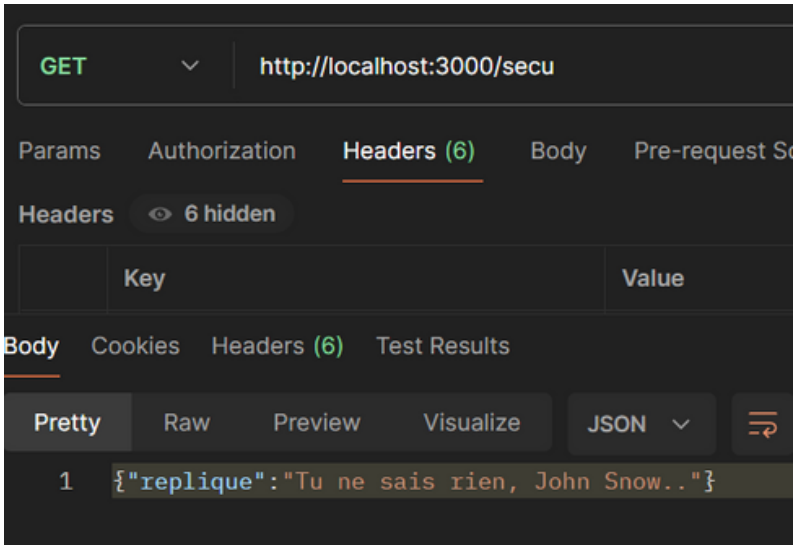
Ensuite, nous passerons à une étape essentielle : la transition de notre protocole de communication de HTTP à HTTPS. Cette migration vers HTTPS permettra de sécuriser toutes les échanges avec nos serveurs en utilisant des certificats que nous générerons nous-mêmes. Nous examinerons en détail le processus de génération de certificats et son impact sur la sécurité de nos services.

Enfin, nous explorerons l'utilisation des JSON Web Tokens (JWT) pour mettre en place une authentification plus robuste. Les JWT offrent une méthode sécurisée pour gérer l'authentification des utilisateurs, et nous verrons comment les intégrer à nos services web existants en utilisant les fonctionnalités avancées de Fastify, telles que les plugins et le dispositif decorate.

Chaque étape de ce TP sera accompagnée d'exercices pratiques visant à consolider vos connaissances et compétences. Vous aurez l'occasion de mettre en œuvre les concepts abordés dans des scénarios concrets, ce qui vous permettra de mieux appréhender les défis liés à la sécurisation des services web.
Nous sommes convaincus que ce TP vous fournira une base solide pour comprendre et mettre en œuvre des stratégies de sécurité efficaces dans vos projets de développement de services web. Préparez-vous à plonger dans le monde passionnant de la sécurité des services web avec Fastify !

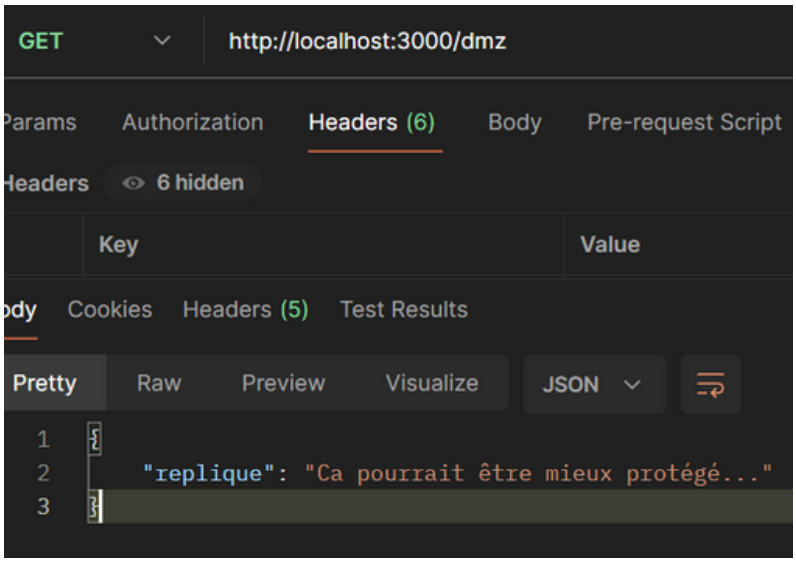
ETAPE 1

En essayant d'accéder à ce endpoint via l'application Postman *http://localhost:3000/secu*, nous obtenons la réponse suivante : {"replique":"Tu ne sais rien, John Snow.."}



Si cette réponse nous est envoyée, cela signifie qu'une erreur de non autorisation s'est produite. Etant donné que l'utilisateur ne s'est pas identifié, nous ne pourrons pas accéder aux informations protégées. Pour accéder à cette route, nous avons besoin d'être identifiés.

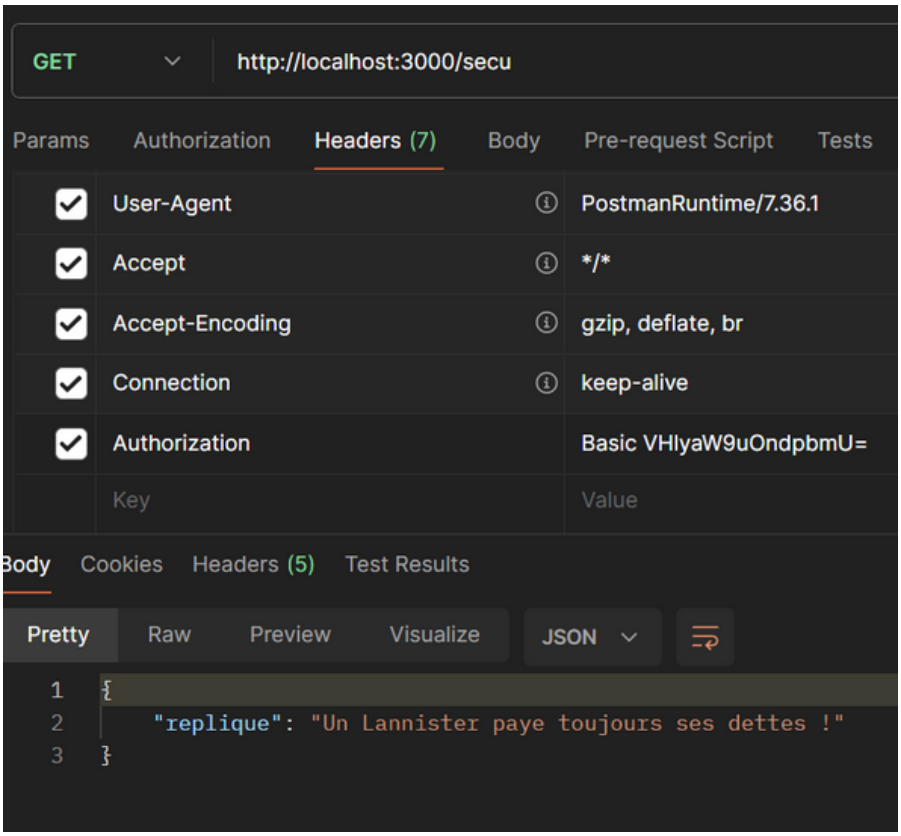
En essayant d'accéder à ce endpoint via l'application Postman *http://localhost:3000/dmz*, nous obtenons la réponse suivante : {"replique": "Ca pourrait être mieux protégé..."}



Pour accéder à cette route, nous n'avons pas besoin d'authentification. Elle est donc considérée comme publique et donc accessible à tout utilisateurs sans authentification.

ETAPE 1

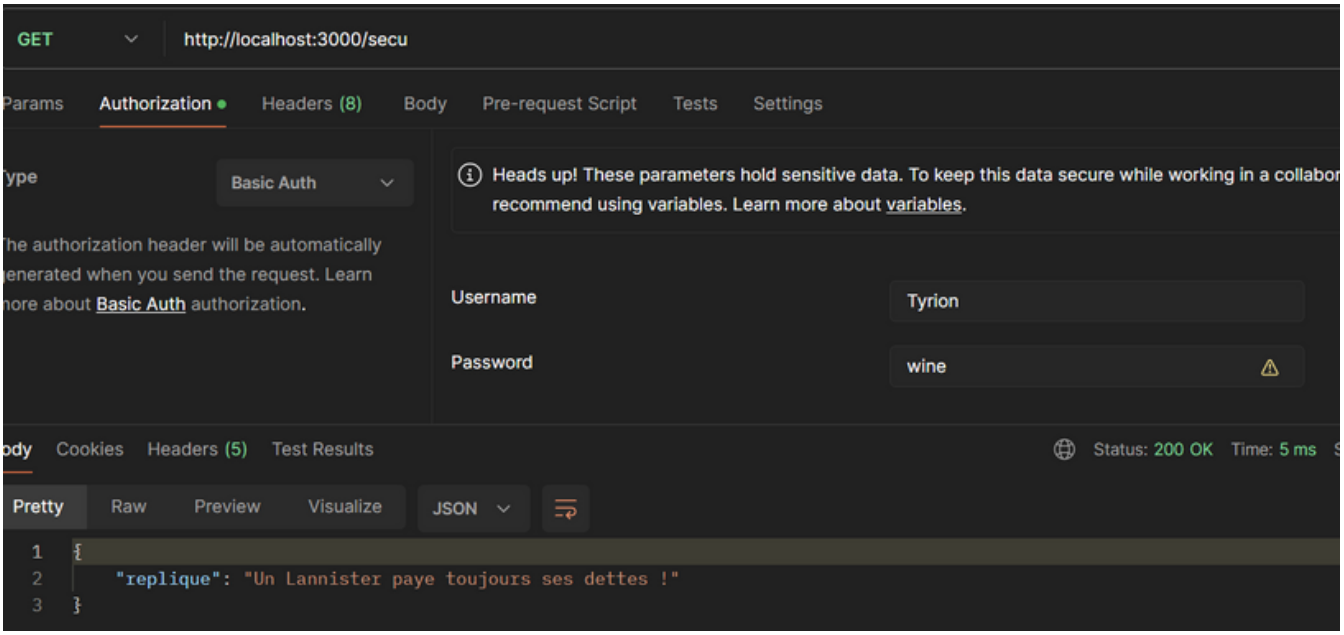
A présent, nous allons tenter d'accéder aux endpoint en se connectant en encodant ces données : Tyrion:wine
Tyrion étant le username authorisé et wine le mot de passe de cet utilisateur
L'encodage nous donne : VHlyaW9uOndpbmU=
Ensuite, nous allons dans Postman et dans nos Headers nous ajoutant une clé Authorization avec notre valeur encodée précédée de Basic qui nous indique que c'est une authentification "basique".



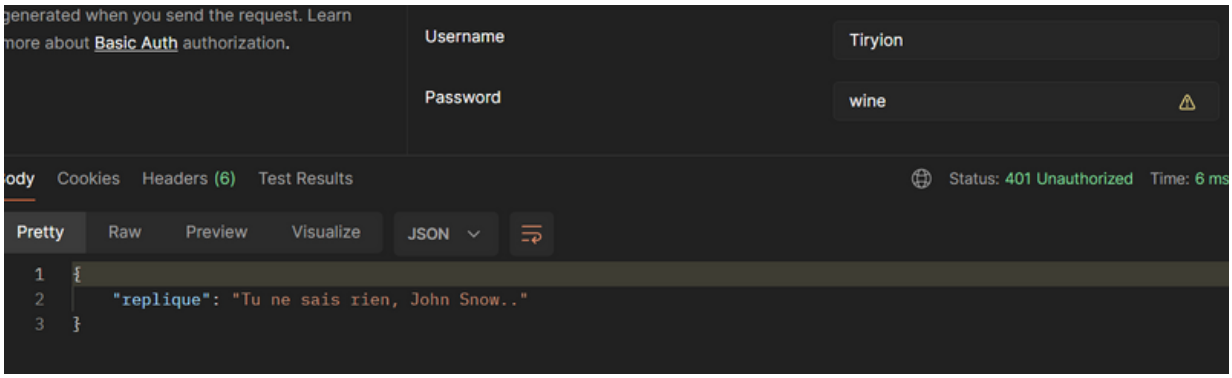
Cette réplique obtenue nous indique que l'autorisation a bien été effectuée et que l'utilisateur a bien accès aux données.

Concernant l'adresse : http://localhost:3000/dmz, la réplique est toujours la même étant donné qu'elle ne requiert pas d'authentification.

ETAPE 1



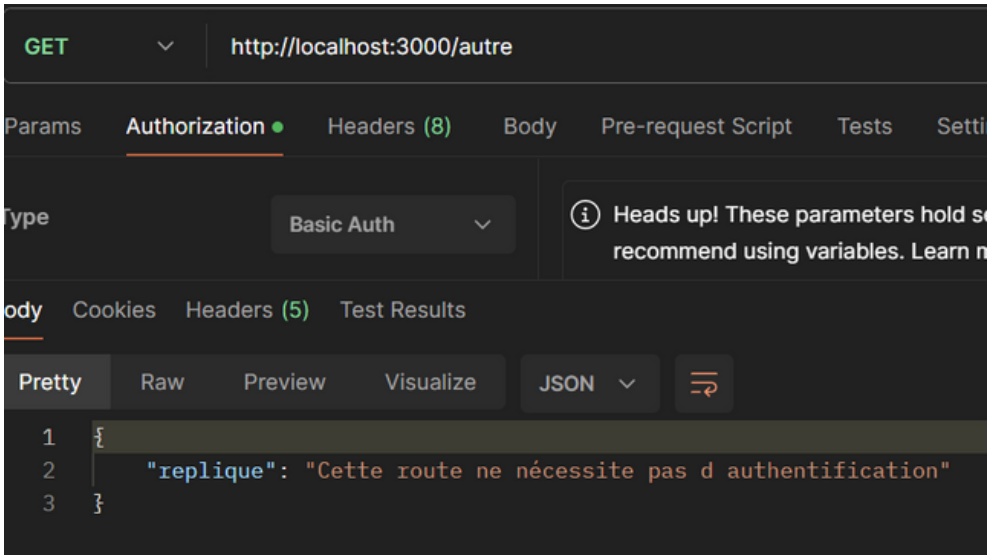
Avec un bon identifiant et un bon mot de passe, l'authentification a bien lieu. Au contraire, avec une erreur, elle n'a pas lieu.



Ensuite, après analyse du code, la fonction after() a pour rôle d'ajouter une nouvelle route, dans notre cas, la route /secu. Elle est effectuée après que toutes les autres routes aient été enregistrées dans notre application Fastify.

ETAPE 1

Après création de la route /autre ne nécessitant pas d'authentification, nous obtenons bien cette réplique sans s'être identifié. J'utilise le même modèle de code que la route /secu mais sans utiliser fastify qui demande une basicAuth.



ETAPE 2

Pour générer une nouvelle clé RSA de 2048 bits appelée server.key, j'utilise OpenSSL.
Pour ce faire j'utilise la commande suivante que nous avons vu en TD : openssl genrsa -out server.key 2048

J'ai pu obtenir cette clé RSA :

```
alexia@Ordi-ThOmas:~$ cat /home/alexia/server.key
-----BEGIN PRIVATE KEY-----
MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQDHcErAwbIbm8mh
wQ6sA4S0MXYbFadODzClspzMjRsgt3NM/t7crbTI49vebcuEuyarEVSi0t8Pv4+O
ZJJ47tNy0DHl1rDaFDysUpzT+Q3IAwsb2RKvGsNip8MBDsq9L2QRzUQPMBB+y24I
B1ulf8TaXt9eACCrK9jI0ijNtC0teAN5txbi+j+pScYTdgW2GC2DQKIWAtZFE2Zq
JrcpXPDjWAZ0ODyAH7qMS2u+Hmj87jMFsHyN9aLX63cpbbfo9vGWvIwrTquAKJD1
t2aanoxW1Gnv/iRYb0nAHYqA/Q9Bm2OaEZuXhKi+XcD8GSkRuC1E80LEgbE845K7
XFiSU60xAgMBAAECggEAB/NBf5qZE4Gr0X9lKtU/EoHQP/cmgrWXTlum5X6QgMbf
WUflsW1ZQhLXPnApB0MAoRwhQBZg3vtXaSk4L4rxsed7vbpHtYHOA16NsuOAOaNp
a3RFSuapuEKQwveIfG8psY35sxNK7QsF+Vr1u8yyl7PlepVNuBUY4LSD/OM8U84d
Ucsqj9uY4iLRySM9MsZ6ADEyZE7rE5L8b8YMGP2nd7P4dlkmarTnGSXeDzcXO7hh
+kbss87qyYl6YzkGVh3qOwAEwE3p/rF8/MUeTeQBPIjTFFilnNdxURty75ImQZdk
eH+PeXmdu1qndjx+i4zcHjVEpRI2jOVvWWvTdkzUgwKBgQDKfVSotsC4MObC96Cs
08nEoJkYSAK5e/iGB2RiqND1H9oocI6ecsQkDgDA1qrW1IdT3Fy3lKcIjXsq6gIK
hnCxPLtPHswRZ4QrWg+o8auO+HNelZxM7vrZxd+VSuwvd7UbyzmuEpuf9y3otGlb
mDRDpfjjxcolfoz/DoIaet40bwKBgQD8JI/c8tXygIAGZUZYozt4cwYjy9Bzjoyw
+2F6qmm9aB8kruPMOD8K8zKXMy7vBbOtRMYs3rDgs45v8hDVmJcrcF8KKQJaJaCn
0wJseVgzjbPVyNges3su6C+MjjMwEXeLlD38zV2GToXZd3YbkZTMqH6GjMSdnbV0
3w4wXFFIXwKBgGyq9BsS8Lwoqj7pq9X0VuzqDx8jziqC6G3tW6dC19kk/ewD89X+
21pKFYdn7Qkbx4Bu22c+JKIHY5k1OmExoxrK7+7XJUcV2a12SL8u6xXwKeIRwm7b
Nc5XTdUN/1C+GnjPCKZtoA0qFv4yi/Q2651+XeLHC+QxVTOwMoxJciSLAoGAOCTo
B9nmu+3TFBl+9YO7XVG3f0cF7oVCXLJM/WRu0ScbMC7lOHcZypVmoXZbMAwD+0QC
IMGhfxdcUx8Ckp08pAaMkxWYmXwhbt00f/TxJ+GYtrTXMtbiSgqXc+PCECUDiogC
2YTdG2fmtAq+uNrM2Fl4evtjWHlii1iTrz8PNl8CgYB1AsBEiAKmCl0C+FeBCgyT
b69O6BPd5yYc97TH/m853kHullSB6+t3nnXKlLGTNNvu1cYVyW+cMGuc2eJAzAEN
lb8MRSwj9C/snLFSAz7XETp37kRKEXHf8vrdmJloGAYuGrMB6cMpgorFw2RyLLmU
ux5xLMv+KgwofOnqGSTSSg==
-----END PRIVATE KEY-----
alexia@Ordi-ThOmas:~$
```

Je crée ensuite mon certificat.

ETAPE 2

```
alexia@Ordi-ThOmas:~$ 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]:
Locality Name (eg, city) []:Paris
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:alexia.fr
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
String too short, must be at least 4 bytes long
A challenge password []:
String too short, must be at least 4 bytes long
A challenge password []:.
An optional company name []:.
alexia@Ordi-ThOmas:~$
```

Ensuite, je le signe avec ma clé RSA que j'ai généré grâce à cette commande :

```
alexia@Ordi-ThOmas:~$ openssl x509 -req -days 365 -in server.req -signkey server.key -out server.crt
Certificate request self-signature ok
subject=C = FR, ST = " ", L = Paris, O = " ", OU = " ", CN = alexia.fr
alexia@Ordi-ThOmas:~$
```

En testant mon certificat, il a bien été accepté.

```
alexia@Ordi-ThOmas:~$  openssl s_server -accept 4567 -cert server.crt -key server.key -www -stateer -accept 4567 -cert server.crt -ke
Using default temp DH parameters
ACCEPT
```

En effectuant la requete *https://localhost:4567/* sur Postman, nous obtenons les résultats d'exécution de la commande OpenSSL 's-server' qui est un outils de tests pour démarrer notre serveur SSL. Cela nous affiche tout d'abord les détails de la configuration de notre serveur, ensuite des informations sur notre session avec le protocole utilisé ainsi que le chiffrement et enfin nous avons des statistiques sur les connexions qui ont été faites.
Cela nous permet de savoir que notre serveur SSL a bien été démarré et que la connexion est sécurisée.

CONCLUSION

Difficultés rencontrées :

Je n'ai pas eu le temps de réaliser l'étape 3 mais je n'ai pas eu de difficultés particulière concernant le TP.

Ce que j'ai appris :

J'ai pu apprendre le fonction des JWT et comment sécuriser un site. J'ai trouvé ca très intéressant et en l'occurrence nous avons réutilisés ces JWT dans notre saé suite au cours que nous avons eu avec vous.