

ÉNONCE TRAVAUX PRATIQUE - SUJET 4

Authentification et Autorisation

Étape 1 - « De base... »

- ♦ A l'aide du site <https://www.base64encode.org/fr/>, effectuer l'encodage des identifiants de connexion tels qu'ils sont attendus. Dans l'onglet **Headers** de **Postman**, ajouter la clé

BUT informatique - R6.A.05 - Développement Avancé - Semaine 4

1



IUT de Paris - Rives de Seine
Université Paris Cité

laurent.giustignano@u-paris.fr

Authorization et placer la valeur obtenue en **base64**, précédé de la mention **Basic**. Retester ensuite les end-points.

Comme le demande le TP je suis aller sur le site pour me créer ma clé en mettant les information nécessaire :

Encodage au format Base64

Il suffit de saisir vos données et d'appuyer sur le bouton d'encodage.

Tyrion:wine

❗ Pour encoder des binaires (comme des images, des documents, etc.), utilisez le formulaire de téléchargement de fichiers un peu plus bas sur cette page.

UTF-8 Jeu de caractères de destination.

LF (Unix) Séparateur de nouvelle ligne de destination.

☐ Encodage chaque ligne séparément (utile lorsque vous avez plusieurs entrées).

☐ Divisez les lignes en segments de 76 caractères (utile pour MIME).

☐ Effectuez un encodage sûr pour les URL (utilisez le format Base64URL).

☒ Mode direct OFF Encodage en temps réel alors que vous tapez ou collez (prenant en charge uniquement le jeu de caractères UTF-8).

> ENCODAGE < Encodage de vos données dans la zone ci-dessous.

VHlyW9uOndpbmU=

J'ai choisies ces paramètres car c'est ce qui apparaît dans le TP :

```
if (username !== 'Tyrion' || password !== 'wine') {  
    return new Error('Winter is coming')  
}
```

The screenshot shows the Postman interface for a GET request to `http://localhost:3000/secu`. The **Headers** tab is selected, showing 8 headers:

Header	Value
Host	<calculated when request is sent>
User-Agent	PostmanRuntime/7.36.1
Accept	*/*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Authorization	Basic VHlyaW9uOndpbmU=
Key	Value

The **Body** tab is also visible, showing a JSON response in the **Pretty** view:

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

- ◆ Pour simplifier l'utilisation, décocher cette clé nouvellement créée. Dans l'onglet **Authorization**, choisissez **Basic Auth** et remplissez les valeurs en claires. Vous observerez que le **Header** est automatiquement complété avec la bonne valeur. Retester ensuite les end-points et confirmer aussi qu'une erreur dans la saisie du **username/password** rejette l'authentification.

The screenshot shows the **Authorization** tab in Postman. The **Type** is set to **Basic Auth**. A warning message states: "Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#)."

The configuration fields are:

- Username:** Tyrion
- Password:** wine

On peut remarquer que l'encodage se fait automatiquement après avoir rentrer les information

	Key	Value	Description	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Authorization	Basic VHlyaW9uOndpbmU=				

Si on change les paramètres des autorisations, l'authentification ne passe plus.

The screenshot shows the Postman interface for a GET request to `http://localhost:3000/secu`. The 'Authorization' tab is active, showing 'Basic Auth' with the username 'Tyrionne' and password 'wine'. A warning message states: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables.' The 'Body' tab shows the response: `{ "replique": "Tu ne sais rien, John Snow.." }`. The status bar at the bottom indicates 'Status: 401 Unauthorized', 'Time: 28 ms', and 'Size: 283 B'.

- ◆ Déterminer maintenant le rôle de la fonction `after()` pour la déclaration de la route `'/secu'`.

Les actions effectuées après le registre, ce qui signifie que toute les routes seront définies après que les autres configuration du plugin aient été enregistrées.

- ◆ Dupliquer le code de route `'/secu'` pour créer la route `'/autre'` à l'intérieur de `after()`, mais elle doit être accessible sans authentification.

The screenshot shows the Postman interface for a GET request to `http://localhost:3000/autre`. The 'Authorization' tab is active, showing 'Basic Auth' with the username 'Tyriong' and password 'wine'. A warning message states: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables.' The 'Body' tab shows the response: `{ "replique": "Un Lannister paye toujours ses dettes !" }`. The status bar at the bottom indicates 'Status: 200 OK', 'Time: 6 ms', and 'Size: 225 B'.


Pour retirer l'authentification j'ai retiré la ligne `onRequest: fastify.basicAuth`, ce qui fait que dès ce moment je n'avais plus besoin d'authentification.

Étape 2 - Prouves qui tu es !

- ◆ Créer une nouvelle clé RSA de 2048 bits appelé `server.key`.

Après avoir fais de la documentation j'ai d'abord créer ma clé avec la commande suivante

```
C:\Users\calli\OneDrive\Documents\IUT\troisième année\C\devavance\TP0penSSL\R6.A.05-TP-Secu1\clés>openssl genrsa -out server.key 2048
```

 server.key

15/02/2024 16:53

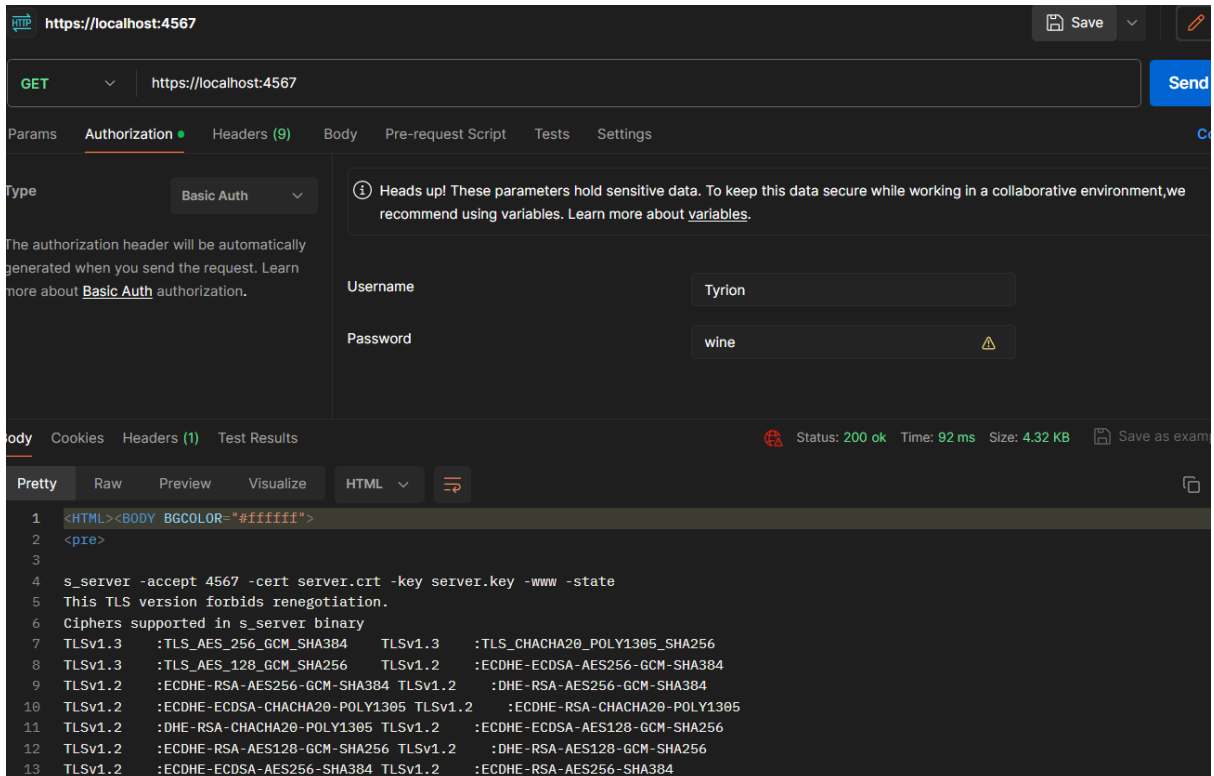
Fichier KEY

2 Ko

J'ai ensuite créé les autres composant nécessaire :

```
C:\Users\calli\OneDrive\Documents\IUT\troisième année\C\devavance\TP0penSSL\R6.A.05-TP-Secu1\src>openssl x509 -signkey s_server.key -in server.csr -req -days 365 -out server.crt
Certificate request self-signature ok
subject=C=FR, ST=cjeiz, L=grnei, O=bfe, OU=rhae, CN=khibae, emailAddress=bjier

C:\Users\calli\OneDrive\Documents\IUT\troisième année\C\devavance\TP0penSSL\R6.A.05-TP-Secu1\src>openssl s_server -accept 4567 -cert server.crt -key server.key -www -state
Using default temp DH parameters
ACCEPT
```



https://localhost:4567

GET https://localhost:4567

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Type Basic Auth

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#).

The authorization header will be automatically generated when you send the request. Learn more about [Basic Auth](#) authorization.

Username Tyrion

Password wine

body Cookies Headers (1) Test Results

Status: 200 ok Time: 92 ms Size: 4.32 KB Save as exam

Pretty Raw Preview Visualize HTML

```
1 <HTML><BODY BGCOLOR="#ffffff">
2 <pre>
3
4 s_server -accept 4567 -cert server.crt -key server.key -www -state
5 This TLS version forbids renegotiation.
6 Ciphers supported in s_server binary
7 TLSv1.3 :TLS_AES_256_GCM_SHA384 TLSv1.3 :TLS_CHACHA20_POLY1305_SHA256
8 TLSv1.3 :TLS_AES_128_GCM_SHA256 TLSv1.2 :ECDHE-ECDSA-AES256-GCM-SHA384
9 TLSv1.2 :ECDHE-RSA-AES256-GCM-SHA384 TLSv1.2 :DHE-RSA-AES256-GCM-SHA384
10 TLSv1.2 :ECDHE-ECDSA-CHACHA20-POLY1305 TLSv1.2 :ECDHE-RSA-CHACHA20-POLY1305
11 TLSv1.2 :DHE-RSA-CHACHA20-POLY1305 TLSv1.2 :ECDHE-ECDSA-AES128-GCM-SHA256
12 TLSv1.2 :ECDHE-RSA-AES128-GCM-SHA256 TLSv1.2 :DHE-RSA-AES128-GCM-SHA256
13 TLSv1.2 :ECDHE-ECDSA-AES256-SHA384 TLSv1.2 :ECDHE-RSA-AES256-SHA384
```

- ♦ En vous inspirant de la documentation sur le site de Fasify (<https://fastify.dev/docs/latest/Reference/HTTP2>), faites évoluer le service web pour que l'accès s'effectue en **https** avec la clé privée et le certificat que vous venez de générer.

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

Étape 3 - Un jeton dans la machine

J'ai tout d'abord Générer les clés de chiffrement ECDSA :

```
C:\Users\calli\OneDrive\Documents\IUT\troisième année\C\devavance\TP0penSSL\R6.A.05-TP-Secu1-auth>openssl ecparam -name prime256v1 -genkey -noout -out .ssl/private.pem  
  
C:\Users\calli\OneDrive\Documents\IUT\troisième année\C\devavance\TP0penSSL\R6.A.05-TP-Secu1-auth>openssl ec -in .ssl/private.pem -pubout -out .ssl/public.pem  
read EC key  
writing EC key
```

Puis j'ai modifier le fichier plugin :

```
import fp from 'fastify-plugin'  
import fastifyJwt from "@fastify/jwt";  
  
1+ usages  Laurent Giustignano *  
export default fp( fn: async function (app, opts) : Promise<void> {  
  app.register(fastifyJwt, {  
    secret: {  
      private: './.ssl/private.pem',  
      public: './.ssl/public.pem'  
    },  
    sign: {  
      algorithm: 'ES256',  
      issuer: 'info.iutparis.fr'  
    },  
  },  
})  
})
```

J'ai ensuite complété les fonction du fichier.js :

```
import { sign } from 'jsonwebtoken'
```

```
export const addUser = async (req, res) : Promise<void> => {
  const { email, password } = req.body
  const hashedPassword : string = createHash("sha256").update(password).digest().toString("hex")

  let user = users.find((u) : boolean => u.email === email)
  if (user) {
    res.status(401).send({
      message: "Utilisateur déjà enregistré",
      user
    })
  } else {
    const newUser : {...} = { email, password: hashedPassword, role: role[Math.floor(Math.random() * role.length)] }
    users.push(newUser)
    res.status(201).send({
      message: "Utilisateur enregistré avec succès",
      user: newUser
    })
  }
}
```

```
1+ usages  Laurent Giustignano *
export const loginUser = async function (req, res) : Promise<void> {
  const { email, password } = req.body
  const user = users.find(u => u.email === email && u.password === createHash("sha256").update(password).digest().toString("hex"))

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

  const token = sign({ payload: { email: user.email, role: user.role }, secretOrPrivateKey: 'secretKey', options: { algorithm: 'ES256' }})
  res.status(200).send({ token })
}
```