

Développement Avancé : TP3

Etape 1 : “De base...”

Dans un premier temps on récupère ici les identifiants tyrion:wine

```
1 usage  Laurent Giustignano
async function validate(username, password, req, reply) : Promise<...> {
  if (username !== 'Tyrion' || password !== 'wine') {
    return new Error('Winter is coming')
  }
}
```

Ensuite on copie colle dans le site d'encodage puis on récupère les identifiants encodé

The screenshot shows the base64encode.org website interface. The browser address bar shows 'base64encode.org/fr/'. The website has a green header with 'BASE64' and tabs for 'Décodage' and 'Encodage'. Below the header, there's a language selector and a description of the site's purpose. The main section is titled 'Encodage au format Base64' and contains a text input field with 'Tyrion:wine'. Below the input field, there are several options for encoding: 'UTF-8' (selected), 'Jeu de caractères de destination', 'LF (Unix)' (selected), 'Séparateur de nouvelle ligne de destination', and checkboxes for 'Encodez chaque ligne séparément', 'Divisez les lignes en segments de 76 caractères', and 'Effectuez un encodage sûr pour les URL'. There's also a 'Mode direct OFF' checkbox. At the bottom, there's a green button labeled 'ENCODAGE' and a text box showing the encoded result: 'VHlyaW9uOndpbmU='.

Ensuite, j'ajoute la clé d'autorisation et je mets l'identifiant encodé. Ensuite, la réplique change en "Un Lannister paie toujours ses dettes"

The screenshot shows the Postman interface for a GET request to `http://localhost:3000/secu`. The 'Headers' tab is active, displaying a table of headers:

Key	Value
<input checked="" type="checkbox"/> Postman-Token	<calculated when request is sent>
<input checked="" type="checkbox"/> Host	<calculated when request is sent>
<input checked="" type="checkbox"/> User-Agent	PostmanRuntime/7.36.1
<input checked="" type="checkbox"/> Accept	*/*
<input checked="" type="checkbox"/> Accept-Encoding	gzip, deflate, br
<input checked="" type="checkbox"/> Connection	
<input checked="" type="checkbox"/> Authorization	Basic VHIyaW9uOndpbmU=

A context menu is open over the 'Authorization' header, showing the option 'Set as variable'.

The 'Body' tab is also visible, showing a JSON response in 'Pretty' format:

```
1 {
2   "replique": "Un Lannister paye toujours ses dettes !"
3 }
```

Lorsque nous saisissons l'identifiant en clair dans l'onglet 'Type d'autorisation' comme 'authentification de base', nous pouvons constater que l'en-tête est automatiquement complété.

The screenshot shows the Postman interface with the 'Authorization' tab selected. The 'Type' dropdown 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, variables." Below this, the 'Username' field contains 'Tyrion' and the 'Password' field contains 'wine'.

The 'Body' tab is also visible, showing the same JSON response as the previous screenshot:

```
1 {
2   "replique": "Un Lannister paye toujours ses dettes !"
3 }
```

GET http://localhost:3000/secu

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Headers Hide auto-generated headers

Key	Value
<input checked="" type="checkbox"/> Authorization	Basic VHlyaW9uOndpbmU=
<input checked="" type="checkbox"/> Postman-Token	<calculated when request is sent>
<input checked="" type="checkbox"/> Host	<calculated when request is sent>
<input checked="" type="checkbox"/> User-Agent	PostmanRuntime/7.36.1
<input checked="" type="checkbox"/> Accept	*/*
<input checked="" type="checkbox"/> Accept-Encoding	gzip, deflate, br
<input checked="" type="checkbox"/> Connection	keep-alive
<input type="checkbox"/> Authorization	Basic VHlyaW9uOndpbmU=
Key	Value

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```

1 {
2   "replique": "Un Lannister paye toujours ses dettes !"
3 }

```

Et lorsque l'on se trompe dans le mot de passe, la réplique change en 'Tu ne sais rien, Jon Snow'.

http://localhost:3000/secu

GET http://localhost:3000/secu

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Type Basic Auth

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

Username Tyrion

Password falsepassword

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, avoid using sensitive data in variables.

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON

```

1 {
2   "replique": "Tu ne sais rien, John Snow.."
3 }

```

La fonction after sert à afficher la réplique 'Un Lannister paie toujours ses dettes !' lorsque

les identifiants d'authentification sont corrects. Lorsque les identifiants sont faux, la fonction `setErrorHandler` est appelée et la réplique devient alors 'Tu ne sais rien, Jon Snow'.

```
± Laurent Giustignano
fastify.after(() : void => {
  fastify.route({
    method: 'GET',
    url: '/secu',
    onRequest: fastify.basicAuth,
    handler: async (req : FastifyRequest<RouteGeneric, http.Server, http.IncomingMessage, SchemaCompiler, FastifyTypeProviderDefault, ContextConfig, boolean> , reply : FastifyReply<http.Server, http.IncomingMessage, http.ServerR
    return {
      reponse: 'Un Lannister paye toujours ses dettes !'
    }
  })
})

± Laurent Giustignano
fastify.setErrorHandler({ handler: function (err : TError , req : FastifyRequest<RouteGeneric, http.Server, RawRequestDefaultExpression<...>, SchemaCompiler, TypeProvider> , reply : FastifyReply<http.Server, RawRequestDefaultExpr

if (err.statusCode === 401) {
  console.log(err)
  reply.code( statusCode: 401).send( payload: {reponse: 'Tu ne sais rien, John Snow..'})
}
reply.send(err)
})
```

Ici, c'est la création de la route /autre :

```
± Laurent Giustignano
fastify.after(() : void => {
  // Route '/secu' avec authentification
  fastify.route({
    method: 'GET',
    url: '/secu',
    onRequest: fastify.basicAuth,
    handler: async (req : FastifyRequest<RouteGeneric, http.Server, http.IncomingMessage, SchemaCompiler, FastifyTypeProviderDefault, ContextConfig, boolean> , reply : FastifyReply<http.Server, http.IncomingMessage, http.ServerR
    return {
      reponse: 'Un Lannister paye toujours ses dettes !'
    }
  })
});

// Route '/autre' sans authentification
fastify.route({
  method: 'GET',
  url: '/autre',
  handler: async (req : FastifyRequest<RouteGeneric, http.Server, http.IncomingMessage, SchemaCompiler, FastifyTypeProviderDefault, ContextConfig, boolean> , reply : FastifyReply<http.Server, http.IncomingMessage, http.ServerR
  return {
    message: 'un Lannister paye sans authentification.'
  }
});
});
```

J'ai créé une nouvelle clé RSA de 2048 bits en utilisant la commande du TD de la semaine 4 'openssl genrsa -out server.key' en ajoutant '2048' à la fin.

VMDebian7x64-bit - VMware Workstation 17 Player (Non-commercial use only)

Player | || | |

Applications Raccourcis mer. 14 févr., 11:15

Parcourir et lancer les applications installées sim@VMDebian: ~/rep1/rep1_1

Fichier Édition Affichage Rechercher Terminal Aide

```
sim@VMDebian:~$ cd Desktop
bash: cd: Desktop: Aucun fichier ou dossier de ce type
sim@VMDebian:~$ ls
bin                krb5.conf          rep1                Téléchargements
Bureau             krb5.conf.sav      rep2                tmp
Documents          liste.txt           rep3                TP3
FusionInventory-Agent.pkg  Modèles            rep4                Vidéos
Images             Musique             sys_env_prog        visu_param
insulte.txt        Public              TD2_seance2_Arbre  wireshark
sim@VMDebian:~$ cd Bureau
sim@VMDebian:~/Bureau$ ls
TD2_seance1_Arborecence.pdf  wireshark
sim@VMDebian:~/Bureau$ cd
sim@VMDebian:~$ cd rep1
sim@VMDebian:~/rep1$ ls
fic1.txt  rep1_1  rep1_2  Sauv
sim@VMDebian:~/rep1$ cd rep1_1
sim@VMDebian:~/rep1/rep1_1$
sim@VMDebian:~/rep1/rep1_1$ openssl genrsa -out server.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
sim@VMDebian:~/rep1/rep1_1$ SSS
```

Ensuite je rentre la commande `openssl req -new -key server.key -out server.csr` :

```

sim@VMDebian:~$ cd Desktop
bash: cd: Desktop: Aucun fichier ou dossier de ce type
sim@VMDebian:~$ ls
bin                krb5.conf          repl               Téléchargements
Bureau             krb5.conf.sav      rep2              tmp
Documents          liste.txt          rep3              TP3
FusionInventory-Agent.pkg  Modèles           rep4              Vidéos
Images             Musique            sys_env_prog      visu_param
insulte.txt        Public            TD2_seance2_Arbre wireshark
sim@VMDebian:~$ cd Bureau
sim@VMDebian:~/Bureau$ ls
TD2_seance1_Arborescence.pdf  wireshark
sim@VMDebian:~/Bureau$ cd
sim@VMDebian:~$ cd repl
sim@VMDebian:~/repl$ ls
fic1.txt  repl_1  repl_2  Sauv
sim@VMDebian:~/repl$ cd repl_1
sim@VMDebian:~/repl/repl_1$
sim@VMDebian:~/repl/repl_1$ openssl genrsa -out server.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
sim@VMDebian:~/repl/repl_1$ openssl req -new -key server.key -out server.csr
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 Descartes
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IUT
Organizational Unit Name (eg, section) []:IUT PARIS
Common Name (e.g. server FQDN or YOUR name) []:Mehdi
Email Address []:mehdi.zaoui@etu.u-paris.fr

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:mehdi.zaoui
An optional company name []:IUT
sim@VMDebian:~/repl/repl_1$ █

```

Puis la commande `openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt` :

```

bin          krb5.conf      repl         Téléchargements
Bureau       krb5.conf.sav  rep2         tmp
Documents    liste.txt      rep3         TP3
FusionInventory-Agent.pkg Modèles        rep4         Vidéos
Images       Musique        sys_env_prog visu_param
insulte.txt  Public         TD2_seance2_Arbre wireshark

sim@VMDebian:~$ cd Bureau
sim@VMDebian:~/Bureau$ ls
TD2_seance1_Arborescence.pdf  wireshark
sim@VMDebian:~/Bureau$ cd
sim@VMDebian:~$ cd repl
sim@VMDebian:~/repl$ ls
ficl.txt  repl_1  repl_2  Sauv
sim@VMDebian:~/repl$ cd repl_1
sim@VMDebian:~/repl/repl_1$
sim@VMDebian:~/repl/repl_1$ openssl genrsa -out server.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
sim@VMDebian:~/repl/repl_1$ openssl req -new -key server.key -out server.csr
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 Descartes
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IUT
Organizational Unit Name (eg, section) []:IUT PARIS
Common Name (e.g. server FQDN or YOUR name) []:Mehdi
Email Address []:mehdi.zaoui@etu.u-paris.fr

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:mehdi.zaoui
An optional company name []:IUT
sim@VMDebian:~/repl/repl_1$ openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.c
rt
Signature ok
subject=/C=FR/ST=Paris/L=Paris Descartes/O=IUT/OU=IUT PARIS/CN=Mehdi/emailAddress=mehdi.zaoui@etu.u-pari
s.fr
Getting Private key
sim@VMDebian:~/repl/repl_1$ █

```

Et enfin la commande `openssl s_server -accept 4567 -cert server.crt -key server.key` :

```

sim@VMDebian:~/repl/repl_1$ openssl s_server -accept 4567 -cert server.crt -key server.key
Using default temp DH parameters
Using default temp ECDH parameters
ACCEPT
█

```

Pour tester le certificat généré, on peut utiliser la commande suivante : `openssl s_server -accept 4567 -cert server.crt -key server.key -www -state` :

```

sim@VMDebian:~/repl/repl_1$ openssl s_server -accept 4567 -cert server.crt -key server.key -www -state
Using default temp DH parameters
Using default temp ECDH parameters
ACCEPT
█

```

Puis on test avec Postman <https://localhost:4567/>

https://10.125.133.200:4567

GET https://10.125.133.200:4567

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Headers

Key	Value	Description
Postman-Token	<calculated when request is sent>	
Host	<calculated when request is sent>	
User-Agent	PostmanRuntime/7.36.1	
Accept	*	
Accept-Encoding	gzip, deflate, br	
Connection	keep-alive	
Authorization	Basic VHYattW9uOndpbnU=	

Body Cookies Headers (1) Test Results

Status: 200 ok Time: 22 ms Size: 4.46 KB

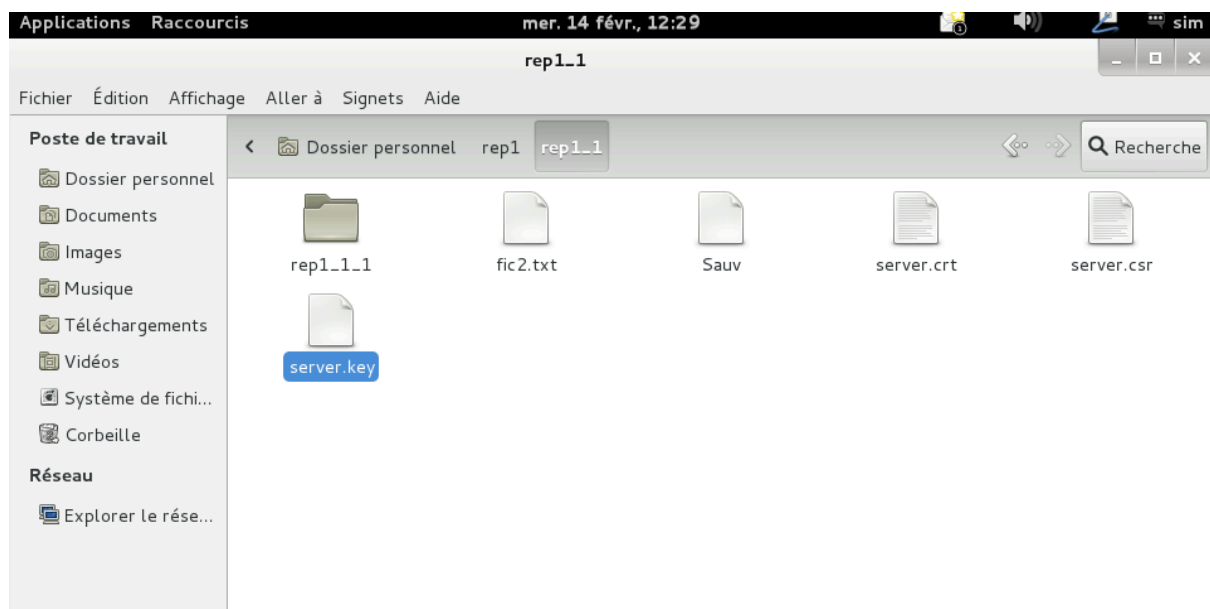
```

1 <HTML><BODY BGCOLOR="#ffffff">
2 <PRE>
3
4 #_server -accept 4567 -cert server.cit -key server.key -www -state
5 Secure Renegotiation IS supported
6 (OpenSSL supports the 's_server' binary)
7 TLSv1/SSLv3:ECDH-RSA-AES256-GCM-SHA384 TLSv1/SSLv3:ECDH-ECDSA-AES256-GCM-SHA384
8 TLSv1/SSLv3:ECDH-RSA-AES256-SHA TLSv1/SSLv3:ECDH-ECDSA-AES256-SHA
9 TLSv1/SSLv3:ECDH-RSA-AES256-SHA TLSv1/SSLv3:ECDH-ECDSA-AES256-SHA
10 TLSv1/SSLv3:SRP-DSS-AES-256-CBC-SHA TLSv1/SSLv3:SRP-RSA-AES-256-CBC-SHA
11 TLSv1/SSLv3:SRP-AES-256-CBC-SHA TLSv1/SSLv3:DHE-DSS-AES256-GCM-SHA384
12 TLSv1/SSLv3:DHE-RSA-AES256-GCM-SHA384 TLSv1/SSLv3:DHE-RSA-AES256-SHA256
13 TLSv1/SSLv3:DHE-RSA-AES256-SHA256 TLSv1/SSLv3:DHE-RSA-AES256-SHA
14 TLSv1/SSLv3:DHE-DSS-AES256-SHA TLSv1/SSLv3:DHE-RSA-CAMELLIA256-SHA
15 TLSv1/SSLv3:DHE-DSS-CAMELLIA256-SHA TLSv1/SSLv3:ECDH-RSA-AES256-GCM-SHA384
16 TLSv1/SSLv3:ECDH-ECDSA-AES256-GCM-SHA384 TLSv1/SSLv3:ECDH-RSA-AES256-SHA
17 TLSv1/SSLv3:ECDH-ECDSA-AES256-SHA TLSv1/SSLv3:AEAD-AES256-GCM-SHA384
18 TLSv1/SSLv3:ECDH-ECDSA-AES256-SHA TLSv1/SSLv3:AEAD-AES256-GCM-SHA384

```

La réponse que j'obtiens montre les options de configuration de mon serveur OpenSSL, les types de chiffrement pris en charge, et les détails de la connexion sécurisée établie entre mon serveur et Postman, comme le protocole SSL/TLS utilisé et le succès de la vérification de la connexion SSL. Cela confirme que la communication entre mon serveur et Postman est sécurisée et fonctionne correctement.

J'ai importé les fichiers :



Puis avec la commande :

```
const myFastify : FastifyInstance<...> & PromiseLike<...> = Fastify( opts: { https: {  
  key: readFileSync('server.key'),  
  cert: readFileSync('server.crt')  
}});
```

Et j'obtiens :

The screenshot shows a web browser window with the address bar displaying `http://localhost:3000/autre`. The browser's developer tools are open, showing the 'Network' tab. A single request is visible, which is a `GET` request to `http://localhost:3000/autre`. The response status is `200 OK`, with a response time of `3 ms` and a size of `225 B`. The response body is displayed in the 'Body' tab, showing a JSON object: `{ "message": "un Lannister paye sans authentication." }`. The browser's address bar shows a `401` status, indicating an unauthorized request.