



UNIVERSITÉ DE TECHNOLOGIE DE
BELFORT-MONTBÉLIARD

TP2 : RS40

Passage du http au https

Chausson Thibault
Renaud Éléanore

Professeur : Abdeljalil Abbas Turki

8 juin 2022

Résumé

Nous imaginons un club privé qui distribue un mot de passe d'entrée aux adhérents une fois par mois. Ils utilisent ce mot de passe pour accéder au club. Les adhérents obtiennent le mot de passe en se connectant sur un lien URL secret. Le lien leur est communiqué lors de leur dernière rencontre physique. Actuellement, il s'agit simplement d'une connexion http, ce qui permet à toutes personnes observant le trafic de lire le mot de passe du club. L'objectif du projet est de remplacer la connexion http par une connexion https.

Exécution du programme

Il suffit d'ouvrir un terminal, de lancer le fichier « `run_server.py` » par la commande « `python3.8 run_server.py` » (ou une autre version). Un premier mot de passe est demandé : « auRevoir », et en deuxième mot de passe c'est aussi « auRevoir ».

Table des matières

1	Vérification du serveur http	1
2	Génération du certificat de l'autorité de certification	2
3	Génération du certificat du serveur	3
4	Connexion https	5
5	Améliorations	6
5.1	Présentation du site	6
5.1.1	Partie utilisateur	6
5.1.2	Partie personnelle du club	8
5.2	Le code	11
5.3	Idées d'améliorations	13
5.3.1	Premier essai	13
5.3.2	Certificat SSL piraté	13
5.3.3	Demander un certificat SSL authentique	13
5.3.4	Une solution qui ne fonctionne qu'en local	14
	Table des images	i
	Liste des codes sources	ii

1 Vérification du serveur http

Nous commençons ce tp en exécutant directement le code du fichier « run_serve.py », et nous remarquons qu’il y a le mot de passe affiché sur la page http :

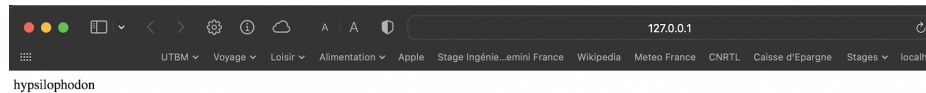


IMAGE 1 – Exécution du serveur

Maintenant utilisons Wireshark pour faire une attaque en écoutant les ports d’envoi des informations, comme sur le schéma suivant :

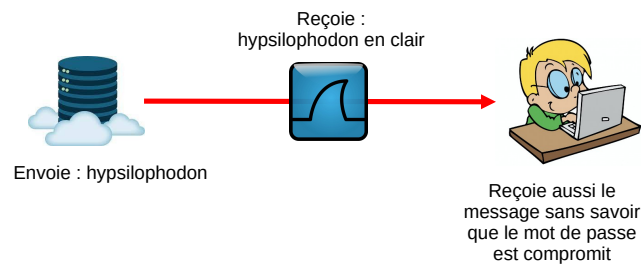


IMAGE 2 – Utilisation de Wireshark

Comme la communication en http, n’est pas chiffrée, nous pouvons facilement écouter le port : 8081, et ainsi intercepter le mot de passe pour rentrer dans le club.

Nous définissons le nouveau code secret comme suit :

```
1 SECRET_MESSAGE ="
    nouveaumotsdepassepersympatiquetastroplongenpluscestcool
"
```

Code source 1 – Nouveau code secret

Voici ce que nous voyons sur Wireshark :

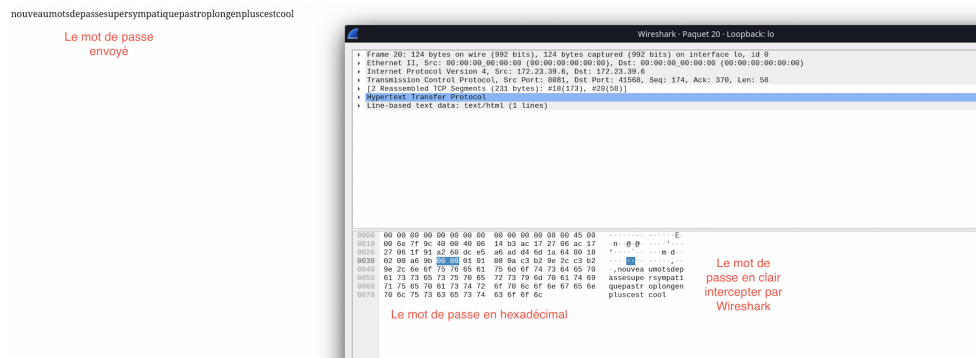


IMAGE 3 – Visualisation dans Wireshark

Ainsi nous visualisons bien que la connexion http n'est pas sécurisée.

2 Génération du certificat de l'autorité de certification

Nous renseignons les arguments des fonctions, comme demandés dans le sujet du TP.

Voici le mot de passe de la CA :

```
1 CA_PASSWORD = "Salutttttt"
```

Code source 2 – Le mot de passe de la CA

Voici les données de la CA :

```
1 CA_CONFIGURATION = Configuration("FR", "Territoire de Belfort",
    , "Sevenans", "UIBM_CA", "localhost")
```

Code source 3 – Les données de la CA

Ainsi nous avons la clé publique suivante pour la CA :

```

1  -----BEGIN CERTIFICATE-----
2  MIIDbTCCAIWgAwIBAgIUC1nmnxTsm4YCI7Wj0INX2BHuBtAwDQY
3  JKoZlhvcNAQELBQAwwZjELMAkGA1UEBhMCRIxHjAcBgNVBAGMF
4  VRlcnJpdG9pcmUgZGUgQmVsZm9ydDERMA8GA1UEBwwIU2V2Z
5  W5hbnMxEDAQBgNVBAoMB1VUQk1fQ0ExEjAQBgNVBAMMCWxv
6  Y2FsaG9zdDAeFw0yMjA1MTgyMDA4NTBaFw0yMjA3MTcyMDA4N
7  TBaMGYxCzAJBgNVBAYTAkZSMR4wHAYDVQQQIDBVUZXJyaXRv
8  aXJlIGRlIEJlbGZvcnQxETAPBgNVBACMFNldmVuYW5zMRAwDg
9  YDVQQKDAAdVVEJNX0NBMRlEAYDVQQDDAIsb2NhbGhvc3Qwg
10 gEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCdTpRzY
11 oWX3YdJrTEACqb6y1erTuD18aBc0o+DzBifl6PtFxuhq2rDw6H1jCd
12 BNuaMYUfK8IgAa30Ygh8UnwIDprn8LJLlbNeRaxPg3iUIaDQCzelX
13 xhJAa1r0Jj1z/XouqyP/6
14   PTtEQKfjqOLHng4ugTyY6ueK4peAxebT9wYJ4Uu9JN
15   a5LB65uAxRvRbjrluv5aE8O35cxUNdiMjuguANO/qTNzXRy/7
16   mQXeJhdTyfrOTh7a5PuGk8L73LVHnQXNWGmeAb0FJE75gG9
17   fSKNjYBtI4rtXsmkQiISeaUC9+WFUcLZGBcJVIFa2GCEJ/1
18   wkmvn0DtKhphys2dBt
19   +KPIAgMBAAGjEzARMA8GA1UdEwEB/
20   wQFMAMBAf8wDQYJKoZIhvcNAQELBQADggEBAJWYNetFkg8QB
21 FMNhVYkNwaIHH326+lmOPb0mTOHj1ETbpvfxmdoshi4YMz6jZP
22 zESfRbIdx7DFnJOPcek7Mb0bYdcpV0a4gFGG9cxpI/
23   Ja3Cdv3iGBSDsoo2XloFrQbWVnwdAtF2YKo4L1ziGX6UTLvahppA
24   tzUJtqwht9AMLyxVoar3IDfofnMnd+mHUxKT03pqRNqUMf4q1hgNs
25   jgwzT4qAQAkAFrQZYNhlzP89sjOt+G3K1+9Py4qMeXh+Uc
26   NuRq22W2++JmRWiyzBIL2N3+aFTAY4dGmwdAV5c6A4wKMEZI/
27   TNpHdVEU0eoqAHb
28   8vLLoLDeX9RPCdUGL9S4MWw=
29   -----END CERTIFICATE-----

```

Code source 4 – Clé publique de la CA

3 Génération du certificat du serveur

Voici le mot de passe du serveur :

```
1 SERVER_PASSWORD = "auRevoir"
```

Code source 5 – Le mot de passe du serveur

Voici les données du serveur :

```

1  SERVER_CONFIGURATION = Configuration("FR", "Territoire de
    Belfort", "Sevenans", "UTBM_SER", "localhost")

```

Code source 6 – Les données du serveur

Ainsi nous avons le certificat pour le serveur :

```

1  -----BEGIN CERTIFICATE REQUEST-----
2  MIICyDCCAbACAQAwwZzELMAkGA1UEBhMCRIxHjAcBgNVBAgMFVRlcnJpdG9pcm
3  Ug
4  ZGUgQmVsZm9ydDERMA8GA1UEBwwIU2V2ZW5hbnMxETAPBgNVBAoMCFVUQk1fU0
5  VS
6  MRIwEAYDVQQDDAIsb2NhbGhvc3QwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwgg
7  EK
8  AoIBAQc++Z/eNcMu3f3VLXk8c6UUI9LwqYZCJZ/kIIVDtGa5+yJtqzX/Y8iQ/
    BK8
9  Avex/vAgn9ToWuLnhqbKCwRcBzsH+amxhWtz8rOc7lpYeVNBoXw+
    S0EAt8wdyFXF
10 fm4g2sKBtuM3RXT5qScQf7PvoX2YQt/Ou7RtuEV3oX7yzTJZRi/J2HVQK+
    ZTFGi7
11 K3lRPqbvo5blP3BSl918DZ2rrJZw2vXTLn2UPX8DEsGDJC5dC9+
    EXSYeo0CDhqOD
12 +icyLykxPPgnHZ5xgs9zp/
    X9B6MDSuBb68CYYxMmXfsY8VFtn6ZIUGrMcnHNssXx
13 57txghQLcz5zzGs+9
    ik7HOwnfaHNAgMBAAGgHDAaBgkqhkiG9w0BCQ4xDTALMAkG
14 A1UdEQQCMAAwDQYJKoZIhvcNAQELBQADggEBAIpJWIQMIR0DoPqOamQyF6jT+O
    +J
15 +lItxVKuapsQVE2fIpGABOTUAoWnLeey8Ps+zYz7l6O70JNZ1ppZ1bJq+
    pVSWklS
16 qeq+GE91DsAsisee+EMSFv94cOWyOIH0aIQKE+i2Pf9PB+
    lhmTjfm7UCJ5FQ7PEN
17 etglEV33bke0g1jwLdGAT/
    GXMykfrqjlnFKpkYUWSWGWTNDGLYImCdKpr3LQ5Kw3
18 vf/4Scw2A2CfWhQTj5MVQ0xlfymJRNZQaUtPdEupuzrba8q0TtNkBTpzNi8TVx
19 g0
20 6qAfvomix7NH8W5P62b5eLua8h9k7ZTtnTSCSt6Ke19EPMUj1UybAogtV+8=
21 -----END CERTIFICATE REQUEST-----

```

Code source 7 – Le certificat du serveur

4 Connexion https

Voici la modification pour avoir un serveur en https, pour ce faire, nous devons mettre un tuple des clés du serveur. Dans le but, d'avoir un context SSL.

```
1 app.run(debug=True, host="0.0.0.0", port=8081, ssl_context=(  
    SERVER_PUBLIC_KEY_FILENAME, SERVER_PRIVATE_KEY_FILENAME))
```

Code source 8 – Pour avoir du https

Nous pouvons remarquer que nous sommes bien en https :



IMAGE 4 – https

Mais le problème une erreur arrive :

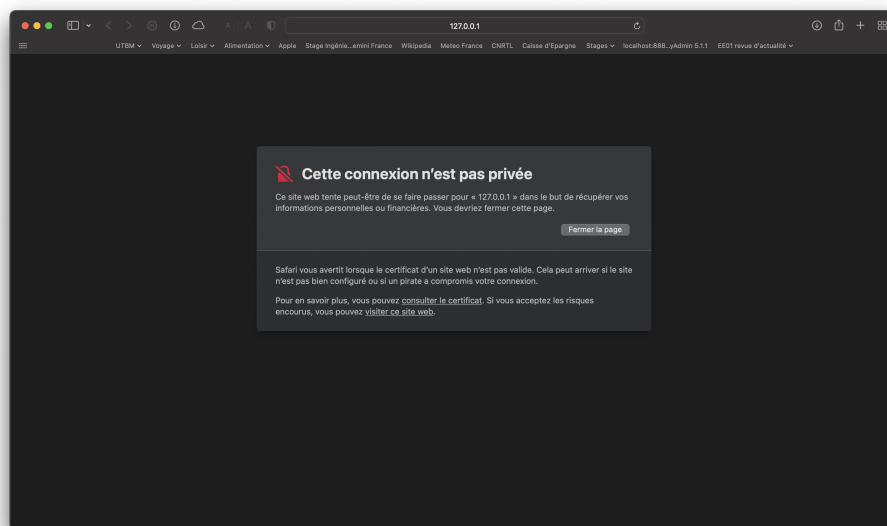


IMAGE 5 – Erreur avec le https

Voici la justification de l'erreur :

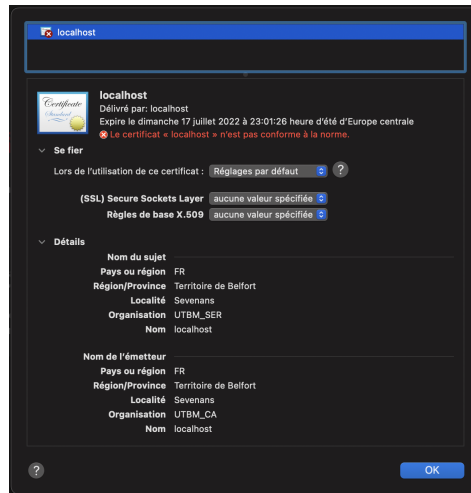


IMAGE 6 – Explication de l'erreur

Dans un premier temps le certificat est déjà expiré à l'ouverture de la page.

De plus, nous avons créé une entité CA, mais elle n'est pas reconnue par le navigateur en tant que tiers permettant de signer des certificats, car notre certificat est auto-signé.

5 Améliorations

5.1 Présentation du site

5.1.1 Partie utilisateur

Pour améliorer la diffusion de notre mot de passe nous pouvons créer une page de connexion, ou chaque utilisateur a un identifiant et un mot de passe fourni par notre club privé.

De ce fait, lors de la dernière visite à ce club les responsables m'ont fourni ce lien de connexion : <https://127.0.0.1:8081> et les informations suivantes, utilisateur : « jean » et le mot de passe : « pierre ».

En cliquant sur le lien j'arrive sur la page de connexion :

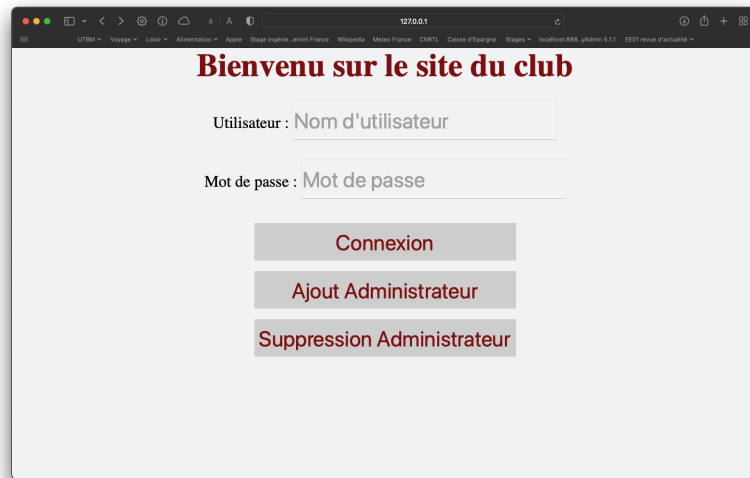


IMAGE 7 – Page de connexion

A mince! Je me suis trompé dans le mot de passe (idem pour le nom d'utilisateur) un message d'erreur s'affiche :

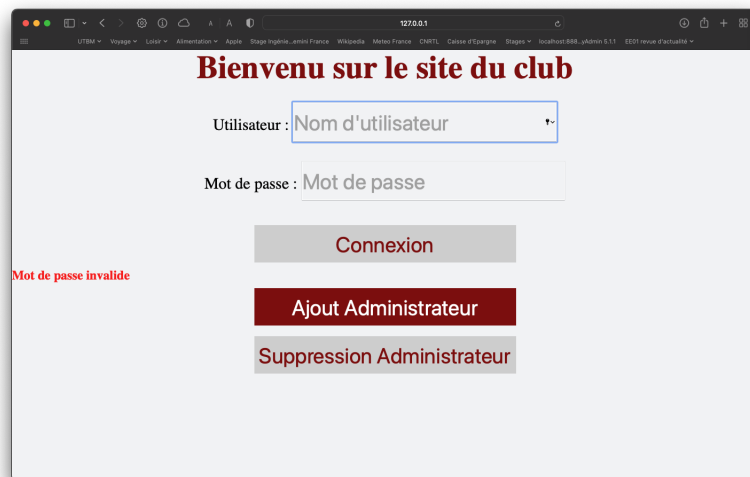


IMAGE 8 – Gestion de l'erreur d'un mauvais nom d'utilisateur

Je viens de me connecter :



IMAGE 9 – Partie utilisateur

Une fois connecté j'ai la possibilité de me déconnecter.

5.1.2 Partie personnelle du club

1. Supposons qu'un nouveau membre arrive, nous pouvons lui créer un compte en se connectant en tant qu'administrateur.

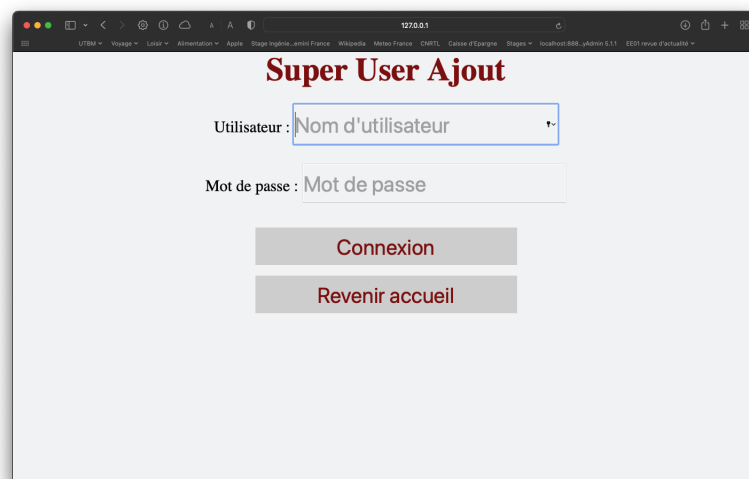



IMAGE 10 – Connexion pour ajouter un utilisateur

Nous pouvons créer un nom d'utilisateur et un mot de passe.



The screenshot shows a web browser window with the address bar displaying '127.0.0.1'. The page has a title 'Bienvenu super User' in green. On the left, there is a red link 'Ajouter un membre'. The main form has two input fields: 'Utilisateur : Nom d'utilisateur' and 'Mot de passe : Mot de passe'. Below these fields are two buttons: 'Ajoute' and 'Déconnexion'. On the left side of the form, the text 'Utilisateur ajouté' is visible.

IMAGE 11 – Ajout d'un utilisateur

Si le nom d'utilisateur est déjà pris, un message d'erreur est affiché.



The screenshot shows the same web browser window as Image 11. The 'Utilisateur' input field now has a red border, indicating an error. Below the form, the message 'Un utilisateur a déjà le même nom d'utilisateur' is displayed. The 'Ajoute' and 'Déconnexion' buttons are still present.

IMAGE 12 – Si deux utilisateurs ont le même nom

2. Supposons que nous devons supprimer un membre.



IMAGE 13 – Connexion pour supprimer un utilisateur

On rentre le nom à supprimer et un message nous informe de la suppression, et un autre message peut nous informer si l'utilisateur est inconnu.

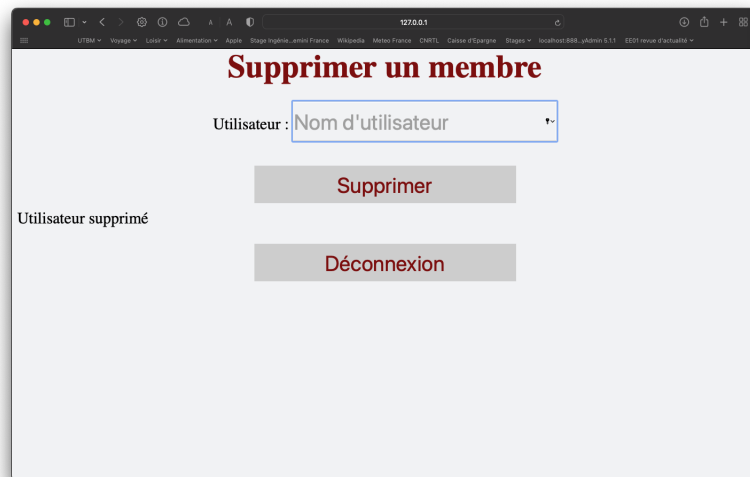


IMAGE 14 – Suppression d'un utilisateur

5.2 Le code

Maintenant, expliquons le code utilisé pour faire ce petit site d'authentification.

Commençons par les fonctions de salage :

1. Le sel : cette fonction crée une chaîne de caractères de manière aléatoire.

```
1 def getSel(length):  
2     """Générer une chaîne aléatoire de longueur fixe"""  
3     str = string.ascii_lowercase  
4     return ''.join(random.choice(str) for i in range(length))
```

Code source 9 – Création du sel

2. Le hash : concatène le mot de passe et le sel puis réalise un hashi grâce à sha256.

```
1 def passPlusSel(pwd, sel):  
2     tout = pwd+sel  
3     print(tout)  
4     res = ha.sha256(tout.encode(encoding='UTF-8', errors='strict')).hexdigest()  
5     return res
```

Code source 10 – Le hash avec le sel

Pour ce qui concerne la partie HTML et CSS, ce ne sont que des « form », donc la compréhension est assez facile.

Lors d'une connexion le HTML envoie :

```
1 <form action="/motDePasse" method="post">  
2     Utilisateur :  
3     <input type="text" name='username' autofocus placeholder="  
4         Nom d'utilisateur"><br><br>  
5     Mot de passe :  
6     <input type="password" name='password' placeholder="Mot de  
7         passe"><br><br>  
8     <input class="bouton" type="submit" value="Connexion">  
9
```

```
10 </form>
11
12 <h2>{{info}}</h2>
```

Code source 11 – Le HTML d'un form de connexion

Le « password » et le « username » par la méthode « post » qui redirige vers l'action « /motDePasse ».

Passons maintenant côté serveur :

```
1 @app.route('/motDePasse', methods=['POST', 'GET'])
2 def login():
3     name1 = request.form['username']
4     pwd = request.form['password']
5     conn2 = sqlite3.connect('myDB.db')
6     cur2 = conn2.cursor()
7     cur2.execute("SELECT COUNT(pass) FROM log WHERE userName=?"
8                  ", (name1,))
9     rows2 = cur2.fetchall()
10    conn2.close()
11    if rows2[0][0] ==0:
12        return render_template('login.html', info='Utilisateur
13        inconnu')
14    else:
15        conn3 = sqlite3.connect('myDB.db')
16        cur3 = conn3.cursor()
17        cur3.execute("SELECT pass, sel FROM log WHERE userName
18        =?", (name1,))
19        rows3 = cur3.fetchall()
20        conn3.close()
21        if rows3[0][0] != passPlusSel(pwd,rows3[0][1]):
22            return render_template('login.html', info='Mot de
23            passe invalide')
24        else:
25            return render_template('home.html', name=name1,
26            code=SECRET_MESSAGE)
```

Code source 12 – Code de la connexion

En recevant l'action : « /motDePasse », le code récupère les données du form, il regarde si un utilisateur existe et il vérifie si le mode de passe avec son salage est le bon. Si oui il renvoie sur la page suivante, sinon il affiche un message d'erreur avec la variable « info ».

5.3 Idées d'améliorations

Suppression du pop up de problème de sécurité.

5.3.1 Premier essai

Nous avons eu l'idée d'exécuter une ligne de commandes bash dans le code Python, dans le but d'ouvrir une version de Google Chrome qui ne vérifie pas les certificats SSL en utilisant :

```
1 import subprocess
2 subprocess.run("C:\Program Files (x86)\Google\Chrome\
  Application\chrome.exe", "--ignore-certificate-errors")
```

Code source 13 – Exécuter une ligne de commandes dans Python

Mais cette méthode a été patchée par Google :

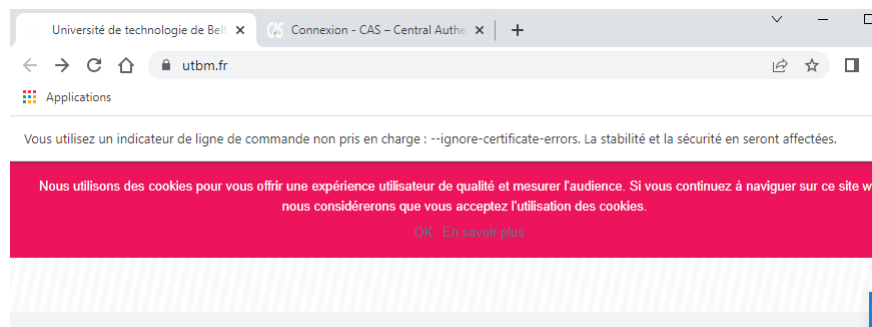


IMAGE 15 – Patche Google

5.3.2 Certificat SSL piraté

Après quelques recherches sur internet, nous nous sommes rendu compte qu'il est très compliqué de trouver un certificat piraté, mais de plus ce dernier devrait correspondre à un url pour un localhost.

5.3.3 Demander un certificat SSL authentique

Il est strictement impossible de demander un certificat SSL pour une URL en localhost, ou 127.0.0.1.

De ce fait, nous pourrions être tenté de contourner ces limitations en créant un nom de domaine dans le DNS 127.0.0.1. Donc, nous obtiendrions un certificat pour notre nouveau nom de domaine. Mais, cela mettrait nos utilisateurs en danger et notre certificat pourrait être révoqué.

De plus, en introduisant un nom de domaine au lieu d'une adresse IP, nous permettrons à un attaquant d'utiliser la technique du "Man in the Middle" (MitM) sur le DNS et injecter une réponse qui pointe vers une adresse IP différente. (Source : letsencrypt.org)

5.3.4 Une solution qui ne fonctionne qu'en local

Nous pouvons créer notre propre context SSL en utilisant la commande suivante :

```
1 openssl req -x509 -out localhost.crt -keyout localhost.key \  
2 -newkey rsa:2048 -nodes -sha256 \  
3 -subj '/CN=localhost' -extensions EXT -config <( \  
4 printf "[dn]\nCN=localhost\n[req]\ndistinguished_name = dn\  
[EXT]\nsubjectAltName=DNS:localhost\nkeyUsage=  
digitalSignature\nextendedKeyUsage=serverAuth")
```

Code source 14 – Création de notre propre context SSL pour notre localhost

Mais cette façon de faire est inutile, car elle ne fonctionne que sur notre machine.

Table des images

1	Exécution du serveur	1
2	Utilisation de Wireshark	1
3	Visualisation dans Wireshark	2
4	https	5
5	Erreur avec le https	5
6	Explication de l'erreur	6
7	Page de connexion	7
8	Gestion de l'erreur d'un mauvais nom d'utilisateur	7
9	Partie utilisateur	8
10	Connexion pour ajouter un utilisateur	8
11	Ajout d'un utilisateur	9
12	Si deux utilisateurs ont le même nom	9
13	Connexion pour supprimer un utilisateur	10
14	Suppression d'un utilisateur	10
15	Patche Google	13

Liste des codes sources

1	Nouveau code secret	1
2	Le mot de passe de la CA	2
3	Les données de la CA	2
4	Clé publique de la CA	3
5	Le mot de passe du serveur	3
6	Les données du serveur	4
7	Le certificat du serveur	4
8	Pour avoir du https	5
9	Création du sel	11
10	Le hash avec le sel	11
11	Le HTML d'un form de connexion	12
12	Code de la connexion	12
13	Exécuter une ligne de commandes dans Python	13
14	Création de notre propre context SSL pour notre localhost . .	14