**TP 2 : RS40**

**Connexion HTTPS : Compte Rendu**

**Pour le :** 19 Juin 2022

**Semestre :** Printemps 2022

**UV :** RS40

**Responsable UV :** Abdeljalil ABBAS-TURKI

**Réalise par :** ALBUZLU Gökdeniz & TRABOULSI Rawan

# Contexte :

Nous supposons 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 toute personne observant le trafic de lire le mot de passe du club.

# Objectif :

L’objectif du projet est de remplacer la connexion http par une connexion https.

# Introduction

Afin de comprendre l’objectif du projet, il faut tout d’abord définir la connexion HTTPS (HyperText Transfer Protocol Secure). La connexion https est une version sécurisée du http, en effet elle combine des couches de chiffrements comme SSL ou TLS. Cette connexion permet de vérifier le site web que l’utilisateur accède en utilisant un certificat d’autorité mais également un certificat de serveur. Pour remplacer la connexion http par la connexion https nous allons procéder en 3 étapes :

Nous allons créer une autorité de certification (ca) détenant une paire de clés (publique, privée) et un certificat (le tiers de confiance).

Ensuite, nous Générerons un certificat du serveur signé par la ca. Ceci se déroule en deux étapes : Génération du certificat de la requête de certification (CSR : Certificate Signing Request) et la signature du certificat par la ca.

Enfin, on remplace la connexion http par la connexion https.

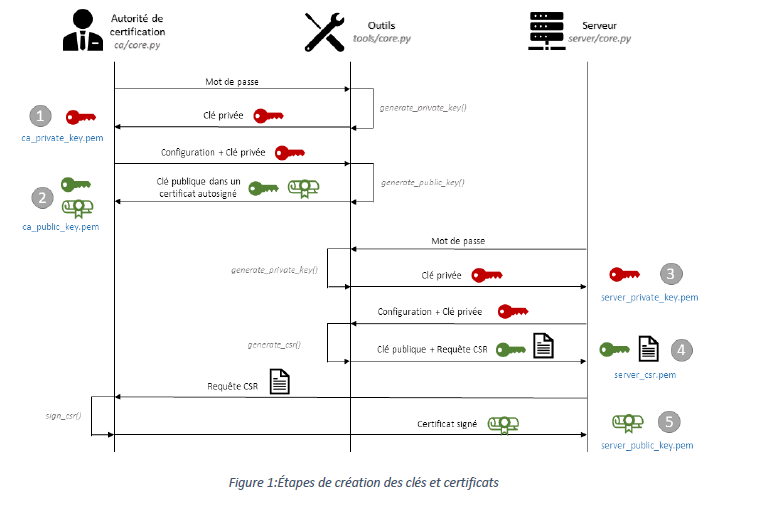


Table des matières

[Contexte : 2](#_Toc105588989)

[Objectif : 2](#_Toc105588990)

[Introduction 2](#_Toc105588991)

[Partie 1 : Vérification du serveur http 4](#_Toc105588992)

[Avant le changement de mot de passe 4](#_Toc105588993)

[Après le changement de mot de passe 5](#_Toc105588994)

[Commentaires 5](#_Toc105588995)

[Partie 2 : Génération du certificat de l’autorité de certification 6](#_Toc105588996)

[Partie 3 : Génération du certificat du serveur 7](#_Toc105588997)

[Partie 4 : Connexion https 8](#_Toc105588998)

[Partie 5 : Améliorations 10](#_Toc105588999)

[Conclusion 11](#_Toc105589000)

# Partie 1 : Vérification du serveur http

## Avant le changement de mot de passe

1. On exécute le programme run\_server.py
2. On se rend sur localhost :8081
3. Voici le mot de passe envoyé par le serveur

Une image contenant texte

Description générée automatiquement

Le mot de passe que le navigateur nous envoie

Une image contenant texte

Description générée automatiquement

Ce mot de passe est également visible sur WireShark comme on peut le constater sur l’image ci-dessus.

## Après le changement de mot de passe

Une image contenant texte

Description générée automatiquement

Le navigateur nous retourne le nouveau mot de passe que nous avons choisie

Une image contenant texte

Description générée automatiquement

Le nouveau mot de passe est aussi visible sur WireShark.

## Commentaires

Nous avons vue précédemment que peu importe le mot de passe utilisé, le protocole HTTP n’est pas sécurisé. En effet, en utilisant un outil simple tel que WireShark, un attaquant peut accéder facilement à nos identifiant car comme nous avons vue, avec HTTP les données ne sont pas chiffrées, c’est pourquoi il faut utiliser un protocole plus sécurisé : le HTTPS

# Partie 2 : Génération du certificat de l’autorité de certification

Tout d’abord nous complétons les fichiers ca/core.py et build.py afin de générer le certificat d’autorisation

Voici le fichier ca/core.py

Une image contenant texte

Description générée automatiquement

Une image contenant texte, capture d’écran, écran

Description générée automatiquementEt ci-dessous le fichier build.py ou on choisit ‘mdpca’ comme mot de passe de la clé privé de la ca. Puis, dans CA\_CONFIGURATION nous trouvons les données de la ca.

Pour visualiser le ***ca-public-key.pem*** j’utilise la fonction print\_perms comme ci-dessous

Une image contenant texte, mur, intérieur, orange

Description générée automatiquement

Voici ce qu’on obtient comme résultat :

# Partie 3 : Génération du certificat du serveur

Dans cette partie, nous compléterons ***server/core.py*** et ***build.py*** de telle sorte à ce qu’on génère la clé privée du serveur (***server-private-key.pem***) ainsi que le certificat csr (***server-csr.pem***), qui contient la clé publique du serveur.

Une image contenant texte, écran, capture d’écran

Description générée automatiquementOn complète la classe Server pour générer la clé privée du serveur ainsi que le certificat csr.

Ensuite, on complète le fichier build.py pour générer le serveur.

On choisit le mot de passe ‘mdpserver’ pour notre serveur



Voici les données de configuration de notre serveur :



Enfin on crée notre serveur comme ci-dessous



Pour la visualisation du certificat du serveur on utilise encore une fois la fonction print\_perms



Voici ce qu’il nous affiche :



# Partie 4 : Connexion https

Cette partie consiste à modifier le fichier ***run\_serveur.py*** afin de permettre aux membres du club de se connecter en https.

Voici ce qu’on change :

Une image contenant texte

Description générée automatiquement

Pour passer de http à https on doit utiliser ssl\_context qui contient les clés du serveur. Ainsi cela va permettre d’effectuer une connexion qui est plus sécurisée que le http puisque Le protocole SSL( Secure Socket Layer) consiste à chiffrer les données.

Lorsqu’on teste la connexion https, le programme nous demande le mot de passe du serveur. On saisie le mot de passe que nous avons définies. Pour notre cas c’est « mdpserver »

Voici ce que le terminal nous affiche après avoir saisie le bon mot de passe

Une image contenant texte

Description générée automatiquement

On se rends sur https :127.0.0.1 :8081 pour tester la connexion https.

Cependant le navigateur nous affiche un petit problème ;

Une image contenant texte

Description générée automatiquement

Nous pouvons accéder à notre serveur en cliquant sur paramètre avancés, puis en cliquant sur continuer vers 127.0.0.1

Ou encore on peut accéder aux paramètres de sécurité de google chrome (pour notre cas), cliquer sur gérer les certificats puis, importer notre certificat.

Une image contenant texte

Description générée automatiquement

# Partie 5 : Améliorations

Pour cette partie, nous avons décidé de créer une page HTML qui se trouve dans un dossier nommé ‘templates’. Cette page contient 2 champs ; identifiant et mot de passe. En effet, on invite l’utilisateur à saisir ses identifiants pour qu’il puisse accéder au message secret.

Pour réaliser cela nous avons utilisé la fonction render\_template de la bibliothèque flask.

Une image contenant texte

Description générée automatiquement

Ensuite, nous avons intégré également un fichier de style à notre page HTML (index.css). Pour cela nous avons créer un dossier appelé ‘static’.

Une image contenant texte, nature, ciel nocturne

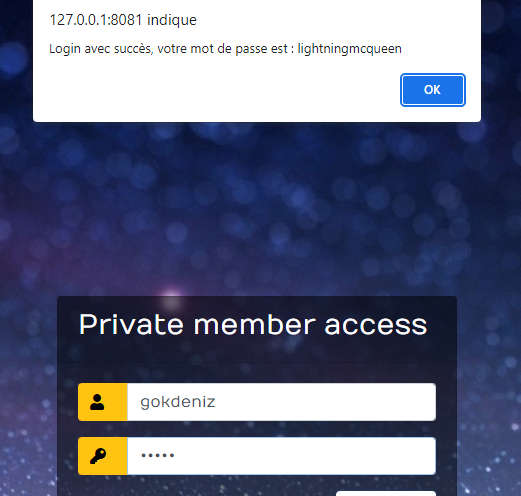
Description générée automatiquement

Enfin nous avons intégré une fonction JavaScript qui vérifie les identifiants saisis par l’utilisateur. Si les identifiants sont corrects on affiche le message secret sinon on affiche un message d’erreur.

Voici les identifiants que nous avons choisi :

-identifiant : gokdeniz

-mot de passe : rawan



# Conclusion

Ce projet de RS40, nous montre les faiblesses du protocole http, en effet http n’est pas sécurisé car les données ne sont pas chiffrées et peuvent donc être intercepter par les attaquants. C’est pourquoi il faut éviter la connexion http et utiliser la connexion https. Car la connexion https combine des couches de sécurités comme le protocole de SSL que nous avons utilisé durant ce projet (ssl\_context). Ce projet nous à également permis de voir en détails les différentes étapes du protocole de https mais nous à également montré en quoi ce protocole est sécurisée.