TP RS40

Objectif

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

Résumé des étapes

Pour ce faire, le projet consiste à :

1. Créer une autorité de certification (ca) détenant une paire de clés (publique, privée) et un certificat (le tiers de confiance)
2. Générer un certificat du serveur signé par la ca. Ceci se déroule en deux étapes :
    a. Générer le certificat de la requête de certification (CSR : Certificate Signing Request)
    b. Faire signer le certificat par la ca.
3. Remplacer la connexion http par la connexion https

Le rendu

Plusieurs parties sont nécessaires pour réaliser le projet. Le projet consiste à accomplir chacune de ces parties. Chaque partie nécessite une réponse sous la forme d'un code ou d'une capture d'écran accompagné d'une remarque. L'ensemble vous permettra de constituer un rapport accompagné de fichiers de code en python. Le rapport à rendre est au format pdf. Le nom du fichier est composé des noms des auteurs (vous pouvez réaliser le projet en binôme). Le rapport sera accompagné de tous vos fichiers *.py et *.pem. **Le projet est à rendre avant le dimanche 12 juin minuit Teams**.

Prérequis

Vous avez besoin d'un environnement de développement python tel que ***Anaconda***, ***PyCharm***, ou tout autre environnement auquel vous êtes habitués.

Vous aurez besoin des librairies python suivants :

- Flask : pour lancer un serveur http ou https
- cryptography : pour les méthodes de génération des clés et certificats
- pem : pour *parser* le contenu d'un fichier pem

Pour installer une librairie python il suffit d'utiliser l'installeur de python pip : ***pip install [nom_du_package]***, ou directement les outils intégrés tel que l'outil « Python Packages » dans PyCharm.

Fichiers

Ce projet est fourni avec un ensemble de fichiers *.py . Certains fichiers doivent être complétés pour fonctionner. Ceci est indiqué dans l'étape correspondante. Le tableau ci-dessous donne le descriptif de chaque fichier *.py.

Fichier	**Contenu**
run_server.py	Permet de lancer le serveur pour fournir le mot de passe du club. Doit évoluer de http vers https.
tools/core.py	Contient des classes et méthodes utilitaires : • Classe ***Configuration*** : définition des paramètres de certificat • Méthode ***generate_private_key***() : génère une clé privée • Méthode ***generate_public_key***() : génère un certificat autosigné de l'autorité de certification (ca) • Méthode ***generate_csr***() :génère un certificat csr de la requête de certification • Méthode ***sign_csr***() : signe un certificat csr par la clé privée du ca
ca/core.py	Contient des classes et méthodes du CA : • Classe ***CertificateAuthority*** : définit une autorité de certification qui crée une clé privée et une clé publique dans un certificat autosigné • Méthode ***sign***() : signe un certificat avec la clé privée du ca
server/core.py	Contient des classes et méthodes du Server : • Classe ***Server*** : définit un serveur qui crée une clé privée et une requête csr • Méthode ***get_csr***() : renvoie la requête csr
build.py	Utilise tous les fichiers core.py pour effectuer toutes les étapes de création des clés et des certificats
print_pems.py	Affiche le contenu des fichiers pems générés par build.py

La figure ci-dessous décrit les étapes avec les fichiers pythons et méthodes correspondants.

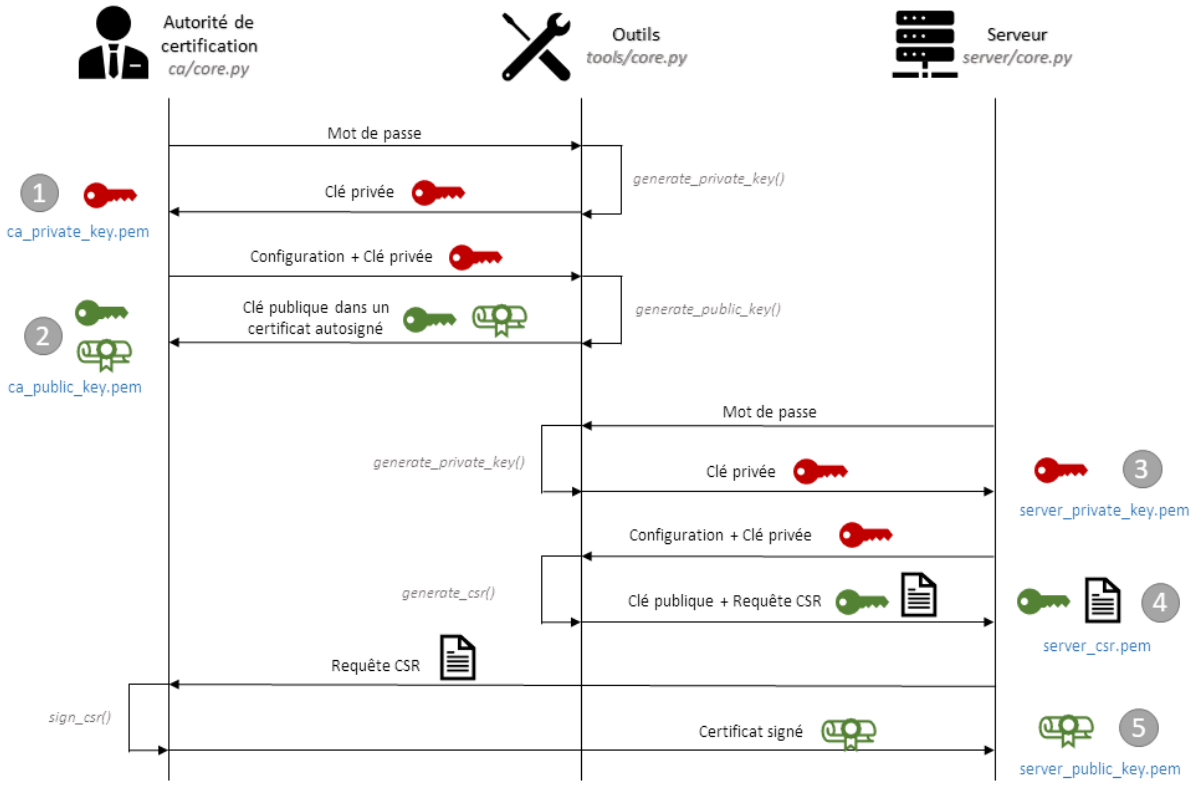

Figure 1:Étapes de création des clés et certificats

Partie 1 : Vérification du serveur http

Voici comment exécuter le serveur http :

1. Exécuter le programme ***run_server.py***
2. Aller sur votre navigateur et tapez «*localhost:8081*».
3. Vérifier que vous obtenez bien le mot de passe envoyé par le serveur
4. N'oubliez pas d'arrêter le serveur quand vous avez terminé

Pour se convaincre que tout le monde peut observer le mot de passe, utiliser ***wireshark***. Il est possible de filtrer les messages capturer selon l'interface, le protocole et le port. Pour «écouter l'hôte local, il suffit d'écouter le Loopback. Pour pouvoir filtrer selon le port vous tapez tcp.port==8081. Vous cherchez par la suite la ligne contenant dans la partie information « text/html ».

Pour cette étape, vous devez changer le mot de passe fourni et présenter une copie d'écran commentée du navigateur et de wireshark.

Partie 2 : Génération du certificat de l'autorité de certification

Cette partie correspond aux étapes 1 et 2 de la figure.

Il faut compléter les fichiers ***ca/core.py*** et ***build.py*** de telle sorte à générer le certificat autosigné de la ca (voir Figure). Ainsi vous avez besoin de générer le fichier contenant la clé privée de la ca : ***ca-private-key.pem***. Ce fichier n'est accessible qu'à l'aide d'un mot de passe que vous avez défini (introduit en paramètre de la fonction ***generate_private_key***). Par la suite vous complétez les fichiers ***ca/core.py*** et ***build.py*** pour pouvoir générer le certificat autosigné nommé ***ca-public-key.pem***. Il faut renseigner les paramètres de la fonction ***generate_public_key*** avec les données liées à l'autorité de certification (country, state…). Afin de visualiser le certificat autosigné, vous compléterez le fichier ***print_pem.py*** qui affiche le contenu des fichiers ****.pem***. Pour l'affichage vous pouvez utiliser la librairie pem de python.

Dans le rapport, vous mettez le résultat de la visualisation du ***ca-public-key.pem***, vous indiquez le mot de passe de la clé privée de la ca ainsi que les données de la ca. Vous êtes libre dans le choix de ces données.

Partie 3 : Génération du certificat du serveur

Cette partie correspond aux étapes 3, 4 et 5 de la figure.

Dans cette étape vous complétez ***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. Vous modifiez ensuite ***ca/core.py*** et ***build.py*** pour obtenir le certificat définitif du serveur (***server-public-key.pem***).

Le rapport doit contenir le mot de passe permettant la lecture du ficher contenant la clé privée du serveur. Il doit aussi contenir le résultat de la visualisation du certificat du serveur.

Partie 4 : Connexion https

Modifier le fichier ***run_serveur.py*** afin de permettre aux membres du club de se connecter en https. Tester le serveur https sur votre navigateur et lire le message affiché.

Pour cette partie vous devez commenter le message d'erreur en fournissant une copie d'écran avec la lecture du mot de passe du club. Discutez aussi les solutions viables et sécurisées à adopter pour éviter le message d'erreur du navigateur.

Partie 5 : Améliorations

Créer au moins un compte utilisateur avec un login et un mot de passe. La manière de traiter le mot de passe est notée.

Ajouter quelques améliorations qui vous semblent intéressantes au ***tools/core.py***. Présentez les améliorations en les justifiants dans le rapport.