Avez-vous déjà vu …. ? … comment sécuriser un site web ?

Précédemment, vous avez créé un serveur Web. Vous avez créé différents sites internet et vous avez fait en sorte d’y accéder grâce à des noms de site différents, des adresses ip différentes, des ports différents aussi. Vous y avez accédé en faisant une résolution DNS dans le fichier hosts de Windows, ou directement sur un serveur DNS sur Linux.

Vous avez donc une très bonne idée du fonctionnement d’Apache.

Objectif :

Comprendre comment fonctionne une connexion web sécurisée.

Savoir mettre en place et configurer une connexion web sécurisée.

# Explications

## Fonctionnement du HTTPS

Le protocole HTTP, HyperText Transfert Protocole, port 80, permet de créer des connexions à la demande, en effet, il permet d’afficher les informations demandées sans faire attention à la façon dont ces infos se déplacent d’un endroit à un autre. Lorsque vous cliquez sur un lien internet, une connexion est demandée, et votre navigateur envoie cette demande de connexion au serveur qui répond en donnant le code la page. Le navigateur interprète le code et l’affiche.

La problématique de ce fonctionnement, c’est le fait que les informations qui circulent peuvent être interceptées et potentiellement détournées de leurs objectifs premiers, ce qui rend les infos et les destinataires vulnérables.

Pour pallier à cela, le protocole HTTPS, HyperText Transfert Protocole Secure, port 443, a été développé. Lui aussi permet d’initialiser une connexion vers un serveur, et de transformer le code source de la page en une page web lisible par l’utilisateur. Mais il a une fonctionnalité supplémentaire, le S, Secure. Cette sécurité est mise en place grâce au protocole TLS, Transport Layer Security.

## SSL/TLS ou TLS ?

Au début de la sécurité informatique on a créé le protocole SSL : Secure Socket Layer, il permettait la sécurité de la connexion et de la protection des données entre 2 systèmes échangeant en TCP en utilisant un algorithme de chiffrement pour brouiller les données en transit, donc empêche leurs lectures sur le réseau.

Le SSL a été maintenu et mis à jour, aujourd’hui la dernière version est la version 3.1, mais lors d’un nouveau projet de mise à jour, les changements attendus étaient très importants. Il a donc été décidé de faire un nouveau protocole : le TLS, Transport Layer Security.

De ce fait, le TLS est une version plus récente, plus sure que le SSL. Mais on parle toujours de sécurité SSL ou SSL/TLS, car ce sont les termes les plus connus et les plus utilisés.

Le protocole TLS peut utiliser plusieurs algorithmes de chiffrement : ECC, RSA ou DSA.

Dans tous les cas, il utilise le chiffrement asymétrique pour l’authentification des machines voulant communiquer et un chiffrement symétrique pour la protection des données.

Chiffrement Symétrique = clé privée partagée entre les utilisateurs/application. Les données sont chiffrées avec une même clé secrète, qui est transmise avec les messages. Ce qui, si elle est interceptée, permet de chiffrer et déchiffrer les infos.

C’est pour cela que la protection des données transmises, faites par chiffrement symétrique a d’abord était protégée par l’authentification de la communication, à l’aide du chiffrement asymétrique.

Chiffrement asymétrique = utilise 2 clés : 1 clé privée et 1 clé publique. Les deux clés sont générées par un logiciel. La clé privée est conservée à l’abri sur la machine qui les a générés, et on envoie la clé publique à la machine distante. La clé publique sert à chiffrer le message et la clé privée à le déchiffrer.

Là encore, on envoie une clé publique par le réseau, cette fois-ci, cette clé sert uniquement à chiffrer le message, mais pas à le déchiffrer ! De façon à bien indiquer qui est le propriétaire de cette clé publique, qui l’envoi à distance sur le réseau, on va utiliser un certificat.

## Certificat

Ce certificat est un document qui permet de fournir des informations sur l’identité de son propriétaire et donc de la personne qui fournit les données. Il va toujours de pair avec une clé publique de façon à ce que la communication entre la machine cliente et le propriétaire du certificat soit chiffrée.

Il comprend :

* N° série unique
* Algorithme de chiffrement
* Le CA (Certificate Authority = Autorité de Certification)
* Une période de validité
* Les infos sur la personne ou la société identifiée
* La clé publique
* La signature par la clé privée du CA

Si on a confiance dans le CA, le certificat assure que le propriétaire de la clé publique ou du serveur (certif serveur) sur lequel je me connecte est bien le contact que j’ai demandé.

Pour prouver l’authenticité d’un certificat, il faut le signer. Pour cela il existe deux possibilités :

* Le faire signer par une autorité de certification
* Faire un certificat auto-signé, c’est-à-dire que le propriétaire signe lui-même son certificat

La première solution est payante, donc, on utilise régulièrement la seconde solution. Cependant, le certificat ne sera pas totalement reconnu par les navigateurs web comme étant certifié, à ce moment-là vous aurez le fameux message vous demander d’ajouter une exception de sécurité.

## Récap

**HTTPS**, HyperText Transfert Protocole Secure, port 443 initialise une connexion sécurisée vers un serveur, et transforme le code source de la page en une page web lisible par l’utilisateur.

Cette sécurité est mise en place grâce au protocole **TLS**, Transport Layer Security. Il utilise un **certificat** pour fournir des informations sur son identité et transmettre la clé publique, de façon à ce que la communication entre la machine cliente et le serveur, propriétaire du certificat soit chiffrée avec cette clé publique, et déchiffrée avec la clé privée (principe des **clés asymétrique**). Ensuite, l’échange de données sera chiffré avec des **clés symétrique**.

# Installer Apache

Créez un site tout ce qu’il y a de plus simple, et faites sa résolution de nom en local, avec le fichier hosts de votre machine Windows.

Attention, l’extension du site ne doit pas être .lan ou .local, sinon le générateur de certificat ne permettra pas la création de ce dernier, car il n’y a pas lieu d’avoir un certificat de sécurité si on est en local.

Pour la suite de ce TP, le site paramétré est accessible sous le nom « exemple.fr ».

# Mise en place de la sécurité

## Activation du module ssl sur apache

Bien que nous allons utiliser le TLS, le nom du module sur apache n’a pas changé, il faut donc activer le module ssl.

sudo a2enmod ssl

Comme demandé, on **redémarre** le service apache pour prendre en compte la modification.

## Téléchargement de l’outil de certificat

Pour générer notre certificat signé par notre certificat personnel, on va utiliser l’outil mkcert. Nous allons devoir télécharger depuis votre machine Linux le paquet d’installation de cet outil. Pour cela, on va utiliser la commande wget. Celle-ci n’est peut-être pas installée sur votre machine, donc procédez à son installation.   
Mkcert va se servir de la commande certutils qui se trouve dans le paquet libnss3-tools, installez-le aussi.

sudo apt install wget libnss3-tools

Créer un dossier mkcert dans le dossier /etc.

mkdir /etc/mkcert

Et déplacez-vous dans ce dossier.

Allez chercher le chemin du paquet le plus récent de mkcert, en date du 28-06-21, il s’agit de la version 1.4.3. Pour cela, allez sur le site

<https://github.com/FiloSottile/mkcert/releases>

Et faites un clic droit, puis « Copier le lien » sur le paquet d’installation qui corresponds à votre besoin, pour nous celui qui se termine par « linux-amd64 », pour la version 1.4.3 le lien obtenu est donc : <https://github.com/FiloSottile/mkcert/releases/download/v1.4.3/mkcert-v1.4.3-linux-amd64>

Maintenant, il faut télécharger cet outil sur votre machine Linux, c’est là où rentre en jeu la commande wget en précisant le lien de l’élément à télécharger (l’option –O mkcert permet de préciser que l’élément téléchargé s’appellera mkcert) :

sudo wget -O mkcert https://github.com/FiloSottile/mkcert/releases/download/v1.4.3/mkcert-v1.4.3-linux-amd64

Il faut rendre ce fichier exécutable pour tout le monde :

chmod +x mkcert

À présent, on a bien l’outil de gestion de certificat au complet. Il ne reste plus qu’à l’installer et créer notre certificat.

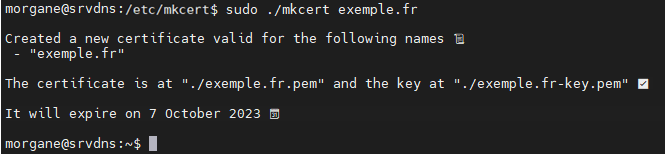
## Installation et création du certificat

Exécutons l’installation de mkcert, cela va créer un certificat local :



Générons le certificat pour notre site internet « exemple.fr »

sudo ./mkcert exemple.fr



On constate que mkcert à générer deux documents :

* Le certificat nommé « exemple.fr.pem »
* Et la clé publique associée nommée « exemple.fr-key.pem »

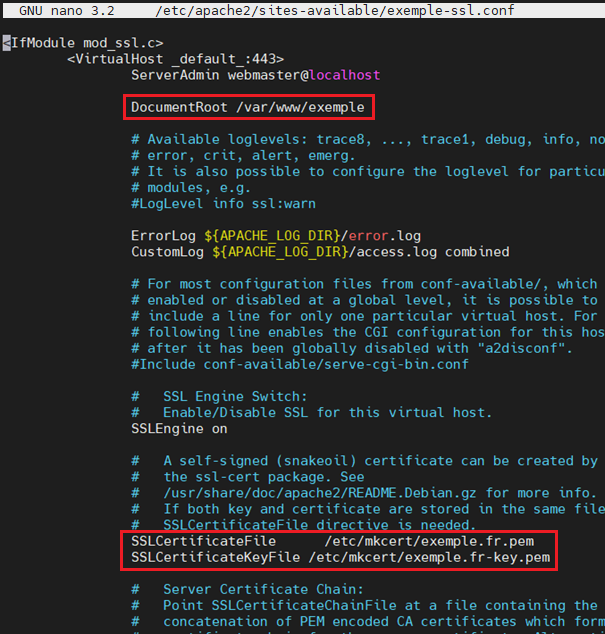
## Activation du HTTPS

Précédemment, le module ssl d’Apache a été activé, puis on a généré un certificat et la clé publique qui va avec. Maintenant, il faut configurer le serveur web pour qu’il utilise ces nouveaux éléments.

Pour cela, il faut copier le fichier de configuration par défaut du HTTPS (default-ssl.conf) pour en faire un fichier de configuration pour notre site exemple.fr, on le nommera exemple-ssl.conf pour le reconnaitre et le différencier du fichier de conf pour le HTTP.

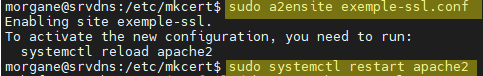


Dans ce fichier, il faut qu’on précise le dossier de notre site qui sera sécurisé, sur la ligne DocumentRoot, puis on doit indiquer le chemin du dossier où sont stockés la clé publique et le certificat qu’on a généré précédemment.



À présent, le site est créé, et il est sécurisé grâce à la clé publique et au certificat.

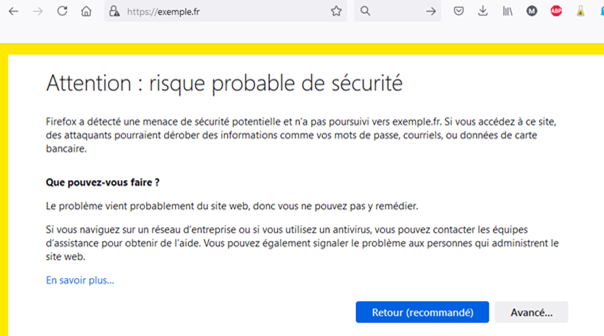
Il reste à le rendre actif.



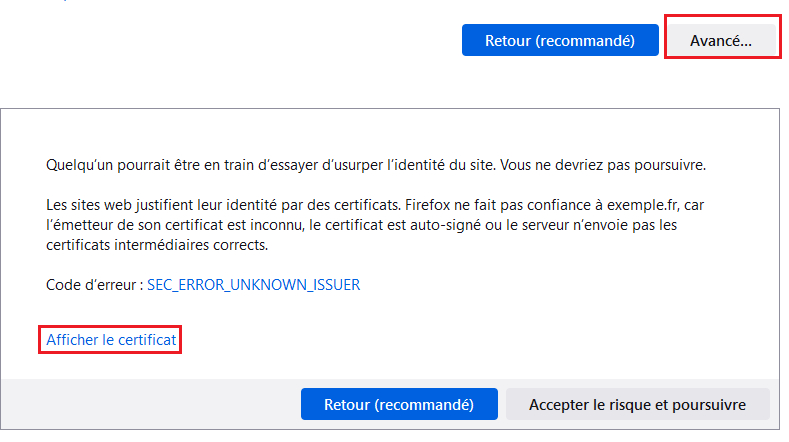
# Tests d’accès

Attention, il faut que la résolution de nom de votre site soit faite, au minimum dans le fichier hosts de votre machine Windows (Rappel : C:\Windows\System32\drivers\etc\hosts).

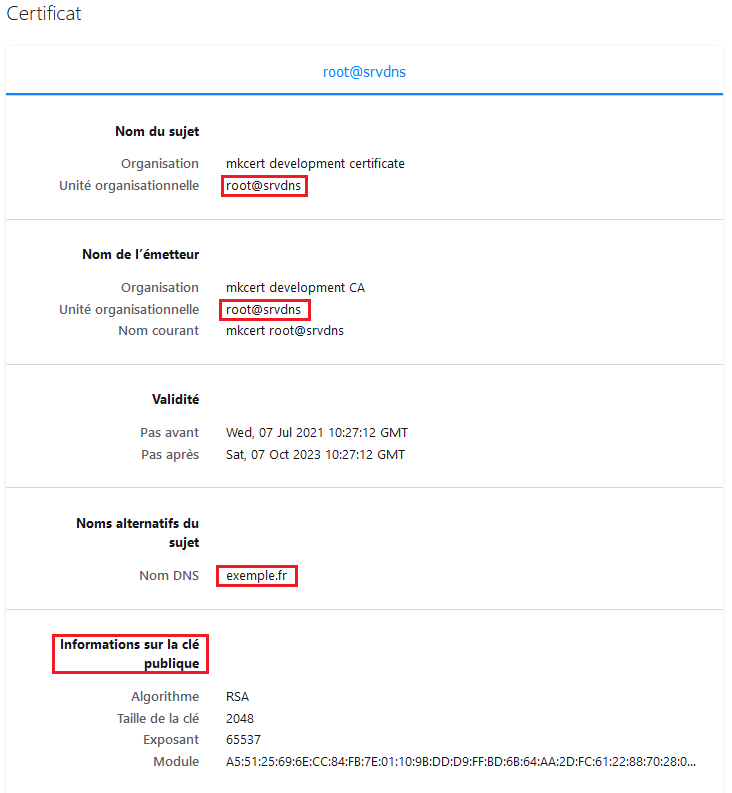
Depuis un navigateur, essayez d’accéder à votre site internet, en utilisant bien le protocole HTTPS :



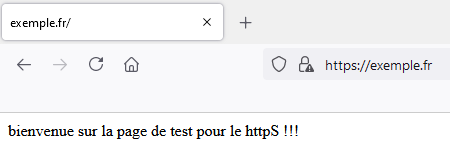
Le certificat étant autosigné, le navigateur nous prévient d’un risque probable de sécurité. Regardons un peu plus en détail, cliquez sur « Avancé » et sur « Afficher le certificat »



Dans un nouvel onglet, il ouvre le certificat, on apprend le nom de l’émetteur du certificat, les informations sur sa validité, le nom du site, les informations sur la clé publique …



Maintenant qu’on a vérifié le certificat, on peut accepter le risque de connexion à ce site, et poursuivre, et nous voilà connectés sur le site, en HTTPS.



Pour éviter cette erreur de sécurité sur le navigateur, il faudrait utiliser un certificat qui ne soit pas autosigné, donc qu’il soit donné par une Autorité de Certification, cela exige d’acheter votre nom de domaine sur internet et de faire appel à un service dédié.