|  |  |
| --- | --- |
|  | **Configuration  d'un serveur HTTPS** |

# Création du certificat

**Prérequis**Pour cette configuration, on doit disposer d'un hôte Linux avec un serveur web nginx et un accès sudo (root).   
On peut utiliser une machine virtuelle locale ou un hôte hébergé dans le nuage.

## Installation de nginx

Si tel n’est pas le cas, pour installer nginx …  
**>> sudo apt install nginx**

### Vérification de la bonne installation

Pour vérifier si nginx est installé et démarré …  
**>> sudo nginx -v  
nginx version: nginx/1.18.0**

**>> sudo systemctl status nginx.service  
● nginx.service - A high performance web server and a reverse proxy server  
 Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)  
 Active: active (running) since Sat 2021-11-06 13:41:07 EDT; 1min 36s ago  
 Docs: man:nginx(8)**

# Création de la paire de clés et du certificat

On utilise le paquet openssl pour la génération du certificat.

**Remarque** …  
Le paquet peut ne pas être déjà installé sur le serveur.  
On peut le vérifier …  
**>> openssl version  
OpenSSL 1.1.1d 10 Sep 2019**

Sinon, il faut l’installer, pour ce faire …  
**>> sudo apt install openssl**

La commande pour la création du certificat est la suivant …  
**>> sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/nginx/ssl/serveur.key \  
 -out /etc/nginx/ssl/serveur.crt  
Generating a RSA private key  
......................................+++++  
.....................................+++++  
…  
-----  
Country Name (2 letter code) [AU]:CA  
State or Province Name (full name) [Some-State]:Québec  
Locality Name (eg, city) []:Montréal  
Organization Name (eg, company) [Internet Widgits Pty Ltd]:squidly.info  
Organizational Unit Name (eg, section) []:  
Common Name (e.g. server FQDN or YOUR name) []:www.squidly.info  
Email Address []:squidly@squidly.info  
-----BEGIN CERTIFICATE-----  
MIID9TCCAt2gAwIBAgIUV2hA6eDYQQ6jlr1tSWa6yoySRfkwDQYJKoZIhvcNAQEL  
BQAwgYkxCzAJBgNVBAYTAmNhMRIwEAYDVQQIDAlRdcODwqliZWMxEzARBgNVBAcM  
Ck1vbnRyw4PCqWwxFTATBgNVBAoMDHNxdWlkbHkuaW5mbzEZMBcGA1UEAwwQd3d3  
…  
DomAxpICeLkc  
-----END CERTIFICATE-----**

Quelques explications sur la commande …

* **openssl req -x509**  
  Demande à OpenSSL de créer un certificat de type x509 autosigné ;
* **nodes**  
  Option pour ne pas chiffrer la clé privée avec un mot de passe, pour qu'Apache puisse utiliser la clé privée sans intervention utilisateur ;
* **days 365**Durée de validité du certificat, d'un an ;
* **newkey rsa:2048**Option pour générer une nouvelle paire de clés pour le certificat, en utilisant le système RSA avec une clé de   
  2 048 bits ;
* **keyout /etc/nginx/ssl/serveur.key**Emplacement où sera stockée la clé privée générée ;
* **out /etc/nginx/ssl/serveur.crt**Emplacement où sera stocké le certificat, qui contient la clé publique générée.

On peut laisser les informations par défaut demandées par OpenSSL lors de la génération du certificat en appuyant sur Entrée.

# Configuration du site Web

Pour configurer un serveur HTTPS, dans le fichier nginx.conf, on inclue le paramètre ssl dans la directive listen dans le bloc serveur.  
On spécifie les emplacements du certificat du serveur et des fichiers de clé privée.

On modifie le fichier de configuration du site …  
**>> sudo nano /etc/nginx/sites-available/default   
server {  
 listen 443 ssl;  
 server\_name www.tux.info;  
 ssl\_certificate /etc/nginx/ssl/serveur.crt;  
 ssl\_certificate\_key /etc/nginx/ssl/serveur.key;  
 ssl\_protocols TLSv1 TLSv1.1 TLSv1.2;  
 ssl\_ciphers HIGH:!aNULL:!MD5;  
 #...  
}**

Quelques explications …

* Le **certificat du serveur est une entité publique**.   
  Il est **envoyé à chaque client qui se connecte au serveur nginx** ;
* La **clé privée est une entité sécurisée** et doit être **stockée dans un fichier à accès restreint**.   
  Cependant, le processus maître nginx doit être capable de lire ce fichier ;
* Les **directives ssl\_protocols et ssl\_ciphers** peuvent être utilisées pour exiger que les clients n'utilisent que les versions et les **chiffrements forts de SSL/TLS lors de l'établissement de connexions** ;
* Depuis la version 1.9.1, nginx utilise ces valeurs par défaut …  
  **protocoles\_ssl TLSv1 TLSv1.1 TLSv1.2** ;  
  ssl\_**ciphers HIGH :!aNULL:!MD5**;

Comme dernière étape, on redémarre le service Apache …  
**>> sudo systemctl restart nginx.service**

Il est maintenant possible de tester le bon fonctionnement en ouvrant votre navigateur, par exemple Firefox, et en visitant **https://localhost**.

Le **fureteur affiche un message d'erreur**.

Cela est normal… Le fureteur est configuré pour ne pas faire confiance aux certificats autosignés.   
Il est néanmoins possible d’**ajouter** **une exception pour ce site** en cliquant sur Avancé puis Ajouter une exception et Confirmer l'exception.

# Optimisation du serveur HTTPS

Les opérations SSL consomment des ressources CPU supplémentaires.   
L'opération la plus gourmande en CPU est l'établissement de liaison SSL.   
Il existe deux manières de **minimiser le nombre de ces opérations par client** …

* **Permettre aux connexions keepalive** d'envoyer **plusieurs requêtes à l’aide d’une seule connexion** ;
* **Réutilisation des paramètres de session SSL** afin d’**éviter** **les poignées de main SSL** pour les connexions parallèles et suivantes.

Les **sessions sont stockées dans le cache de session SSL partagé** **entre les processus de trav**ail et configuré par la **directive ssl\_session\_cache**.   
Un **mégaoctet de cache contient environ 4000 sessions**.   
Le **délai d'expiration du cache par défaut est de 5 minutes**.   
Ce **délai d'attente peut être augmenté** à l'aide de la **directive ssl\_session\_timeout**.

Voici un exemple de configuration optimisée pour un système multicœur avec un cache de session partagé de 10 Mo …  
**>> sudo nano /etc/nginx/sites-available/default   
worker\_processes auto;**

**http {  
 ssl\_session\_cache shared:SSL:10m;  
 ssl\_session\_timeout 10m;**

**server {  
 listen 443 ssl;  
 server\_name www.tux.info;  
 keepalive\_timeout 70;  
 ssl\_certificate /etc/nginx/ssl/serveur.crt;  
 ssl\_certificate\_key /etc/nginx/ssl/serveur.key;  
 ssl\_protocols TLSv1 TLSv1.1 TLSv1.2;  
 ssl\_ciphers HIGH:!aNULL:!MD5;  
 #...  
 }  
}**

# Chaînes de certificats SSL

Il est maintenant possible de visiter https://localhost grâce au certificat autosigné.

Lors de l’accès au serveur Web, le **fureteur affiche un message d'erreur**.   
Cela est normal… Le fureteur est configuré pour ne pas faire confiance aux certificats autosignés.   
Il est néanmoins possible d’**ajouter** **une exception pour ce site** en cliquant sur Avancé puis Ajouter une exception et Confirmer l'exception.

Certains fureteurs (*browsers*) peuvent se plaindre d'un certificat signé par une autorité de certification bien connue, tandis que d'autres fureteurs peuvent accepter le certificat sans problème. Cela se produit parce que l'autorité émettrice a signé le certificat de serveur à l'aide d'un certificat intermédiaire qui n'est pas présent dans la base d'autorités de certification de confiance bien connues qui est distribué dans un fureteur particulier.

Utiliser un **certificat autosigné** (ou expiré) **ne diminue pas la force du chiffrement des données** **HTTP** par rapport à un certificat valide signé par une autorité de certification. En revanche, un certificat autosigné ne permet pas d'authentifier le serveur web, ce qui signifie qu'un attaquant exécutant une attaque MITM peut intercepter et remplacer le certificat du serveur par le sien, puis déchiffrer et modifier toutes les informations échangées entre le client et le serveur, réduisant à néant la sécurité de TLS.

Un **certificat autosigné est généralement utilisé en environnement local** de développement, de test ou de préproduction.

Par ailleurs, le certificat racine d'une autorité de certification racine est autosigné, puisque c'est elle-même qui est chargée de signer les certificats. Cependant, ce certificat fait partie des certificats de confiance des machines clientes, et est donc reconnu comme valide.

# Redirection HTTP vers HTTPS

Si tous les sites Web hébergés sur le serveur sont configurés pour utiliser HTTPS et que l’on ne souhaite pas créer un bloc de serveur HTTP distinct pour chaque site, on peut créer un seul bloc de serveur HTTP fourre-tout.   
Ce bloc redirigera toutes les requêtes HTTP vers les blocs HTTPS appropriés.

Pour créer ce bloc HTTP fourre-tout qui redirigera les visiteurs vers la version HTTPS du site …  
**>> sudo nano /etc/nginx/sites-available/default   
server {  
 listen 80;  
 server\_name \_;  
 return 301 https://192.168.1.100$request\_uri;  
}**

**server {  
 listen 443 ssl http2;  
 server\_name \_;  
 return 301 https://192.168.1.100$request\_uri;  
}**

La directive return 301 https://$host$request\_uri permet de rediriger le trafic vers le bloc de serveur HTTPS correspondant avec le code d'état 301 (Moved Permanently).   
La variable $host contient le nom de domaine de la requête.

Par exemple, si le visiteur s'ouvre <http://192.168.1.100/page2> dans le fureteur (*browser*), Nginx redirigera la demande vers https://192.168.1.100/page2.

Remarque …  
Si cela est possible, il est préférable de créer une redirection par domaine plutôt qu'une redirection globale HTTP vers HTTPS.