|  |  |
| --- | --- |
| Une image contenant texte, clipart  Description générée automatiquement | **Sites Web sécurisés (HTTPS)** |

## Présentation

De nos jours, la plupart des sites Web utilisent le protocole SSL ou TLS.  
Ceci permet des communications client/serveur chiffrées et l’assurance que l’on a contacté le bon serveur … et non un site alternatif.

Les protocoles TLS (*Transport Layer Security*), et son prédécesseur SSL (*Secure Sockets Layer*), sont les protocoles Web utilisés pour envelopper le trafic normal dans un conteneur protégé et chiffré. Bien que SSL ait été remplacé par le protocole TLS il y a quelque temps, SSL demeure toujours un terme couramment utilisé pour cette technologie.

Grâce à ces protocoles, les serveurs peuvent gérer le trafic entre un serveur et les clients en toute sécurité, sans que les messages ne soient interceptés par des tiers.

Le système de certificats aide également les utilisateurs à vérifier l'identité des sites auxquels ils se connectent.

### Utilisation de SSL/TLS

Le principal cas d'utilisation de SSL/TLS est la sécurisation des communications entre un client et un serveur, mais il peut également sécuriser les courriels, la VoIP et d'autres communications sur des réseaux non sécurisés.

Voici les principes essentiels à saisir pour comprendre le fonctionnement de SSL/TLS …

* La communication sécurisée commence par une poignée de main TLS, dans laquelle les deux parties communicantes ouvrent une connexion sécurisée et échangent une clé publique ;
* Pendant la négociation TLS, les deux parties génèrent des clés de session et les clés de session chiffrent et déchiffrent toutes les communications après la négociation TLS ;
* Différentes clés de session sont utilisées pour chiffrer les communications dans chaque nouvelle session ;
* TLS garantit que la partie côté serveur, ou le site Web avec lequel l'utilisateur interagit, est bien celui qu'il prétend être ;
* TLS garantit également que les données n'ont pas été modifiées, car un code d'authentification de message (MAC) est inclus avec les transmissions ;
* Avec TLS, les données HTTP que les utilisateurs envoient à un site Web (en cliquant, en remplissant des formulaires, …) et les données HTTP que les sites Web envoient aux utilisateurs sont chiffrées.   
  Les données chiffrées doivent être déchiffrées par le destinataire à l'aide d'une clé.

### Poignée de main TLS

Les sessions de communication TLS commencent par une négociation TLS.

Une négociation TLS utilise le chiffrement asymétrique, ce qui signifie que deux clés différentes sont utilisées aux deux extrémités de la conversation.   
Ceci est possible grâce à une technique appelée cryptographie à clé publique.

Avec la cryptographie à clé publique, deux clés sont utilisées …

* une clé publique, que le serveur met à disposition publiquement  
  et
* une clé privée, qui est gardée secrète et utilisée uniquement côté serveur.

Les données chiffrées avec la clé publique ne peuvent être déchiffrées qu'avec la clé privée.

Au cours de la négociation TLS, le client et le serveur utilisent les clés publiques et privées pour échanger des données générées aléatoirement, et ces données aléatoires sont utilisées pour créer de nouvelles clés de chiffrement, appelées clés de session.

Voici une poignée de main SSL standard lorsque l'algorithme d'échange de clés RSA est utilisé …

1. **Client Bonjour** (*Client Hello)*  
   Informations dont le serveur a besoin pour communiquer avec le client à l’aide du protocole SSL.   
   Cela inclut le numéro de version SSL, les paramètres de chiffrement, les données spécifiques à la session l
2. **Bonjour serveur** (*Server Hello*)  
   Informations dont le serveur a besoin pour communiquer avec le client à l’aide du protocole SSL.  
   Cela inclut le numéro de version SSL, les paramètres de chiffrement, les données spécifiques à la session ;
3. **Authentification et secret pré-maître** (*Authentiquassions and Pre-Master Secret*)  
   Le client authentifie le certificat du serveur. (Nom commun/Date de validité/Émetteur).   
   Le client (selon le chiffrement) crée le secret pré-maître pour la session, chiffre avec la clé publique du serveur et envoie le secret pré-maître crypté au serveur ;
4. **Déchiffrement et clé maitre** (*Decryption and Master Secret*)  
   Le serveur utilise sa clé privée pour déchiffrer le secret pré-maître.   
   Le serveur et le client effectuent des étapes pour générer le secret principal avec le chiffrement convenu ;
5. **Chiffrement avec clé de session** (*Encryption with Session Key*)  
   Le client et le serveur échangent des messages pour informer que les futurs messages seront chiffrés.

### Chiffrement symétrique et clés de session

Contrairement au chiffrement asymétrique, avec le chiffrement symétrique les deux parties d'une conversation utilisent la même clé.   
Après la négociation TLS, les deux parties utilisent les mêmes clés de session pour le chiffrement.

Une fois les clés de session utilisées, les clés publiques et privées ne sont plus utilisées.

Les clés de session sont des clés temporaires qui ne sont plus utilisées une fois la session terminée.

Un nouvel ensemble aléatoire de clés de session sera créé pour la prochaine session.

### Authentification du serveur d'origine

Les communications TLS du serveur incluent un code d'authentification de message, ou MAC, qui est une signature numérique confirmant que la communication provient du site Web réel. C  
ela authentifie le serveur, empêchant les attaques de l’homme du milieu (*Man in the Middle*) et l'usurpation de domaine.

Il garantit également que les données n'ont pas été modifiées pendant le transport.

### Certificat SSL

Un certificat SSL est un fichier installé sur le serveur d'origine d'un site Web.   
Il s'agit simplement d'un fichier de données contenant la clé publique et l'identité du propriétaire du site Web, ainsi que d'autres informations.   
Sans certificat SSL, le trafic d'un site Web ne peut pas être chiffré avec TLS.

Techniquement, tout propriétaire de site Web peut créer son propre certificat SSL, et ces certificats sont appelés certificats auto-signés.   
Cependant, les navigateurs ne considèrent pas les certificats auto-signés comme aussi fiables que les certificats SSL émis par une autorité de certification.

Les responsables de sites Web doivent obtenir un certificat SSL auprès d'une autorité de certification, puis l'installer sur leur serveur Web (souvent un hébergeur peut gérer ce processus). Une autorité de certification est une partie externe qui peut confirmer que le propriétaire du site Web est bien celui qu'il prétend être.   
Ils conservent une copie des certificats qu'ils délivrent.

### Différence entre HTTP et HTTPS

Le S dans HTTPS signifie sécurisé. HTTPS est juste HTTP avec SSL/TLS.   
Un site Web avec une adresse HTTPS possède un certificat SSL légitime émis par une autorité de certification, et le trafic vers et depuis ce site Web est authentifié et crypté avec le protocole SSL/TLS.

Pour encourager Internet dans son ensemble à passer au HTTPS plus sécurisé, la plupart des fureteurs (*browsers*) Web ont commencé à marquer les sites Web HTTP comme non sécurisés ou dangereux.   
Ainsi, non seulement HTTPS est essentiel pour assurer la sécurité des utilisateurs et des données des utilisateurs, mais il est également devenu essentiel pour établir la confiance avec les utilisateurs.

## Certificat et clés

Un certificat logé sur le serveur Web 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 dont l’accès est restreint.  
Toutefois, le processus maître nginx doit être capable de lire ce fichier.

TLS/SSL fonctionne en utilisant une combinaison d'un certificat public et d'une clé privée.   
La clé SSL est gardée secrète sur le serveur. Ellelest utilisé pour chiffrer le contenu envoyé aux clients.   
Le certificat SSL est partagé publiquement avec toute personne demandant le contenu.   
Il peut être utilisé pour déchiffrer le contenu signé par la clé SSL associée.

Alternativement, la clé privée peut être stockée dans le même fichier que le certificat :

* Dans ce cas, il est important de restreindre l'accès au fichier.   
  Bien que le certificat et la clé soient stockés dans un seul fichier, seul le certificat est envoyé aux clients ;
* Les directives ssl\_protocols et ssl\_ciphers peuvent être utilisées afin d’exiger que les clients n'utilisent que les versions et les chiffrements forts de SSL/TLS lors de l'établissement de connexions.

### Certificat auto-signé

Un certificat auto-signé chiffrera la communication entre un serveur et tous les clients.   
Cependant, puisque ce certificat n'est signé par aucune des autorités de certification de confiance incluses avec les fureteurs (*browsers*) Web, les utilisateurs ne peuvent pas utiliser le certificat afin de valider automatiquement l'identité du serveur distant.  
Un certificat auto-signé peut être approprié si on ne possède pas de nom de domaine associé à un serveur et pour les cas où l'interface Web chiffré n'est pas orientée utilisateur. Dans la plupart des cas, il est préférable d'utiliser un certificat signé par une autorité de certification (CA).

## Installation

Pour installer Apache seulement, on installe simplement le paquet nginx …  
**>> sudo apt update && sudo apt upgrade -y && sudo apt install nginx**

À la suite de cette installation le serveur doit fonctionner et être accessible à l'adresse …  
**http://localhost** (à partir du même hôte)  
ou  
**http://<Adresse IP>**

La page par défaut s'afficher dans le fureteur (*browser*).   
Il s'agit du **contenu du fichier /var/www/html/index.html** qui est affiché par défaut.

## Lancement

nginx se lance automatiquement dès son installation, et se relance automatiquement à chaque démarrage.   
C'est l'idéal pour un serveur qui doit continuellement fournir du contenu en ligne, mais pour un serveur de test (on dit de développement) on peut éventuellement désirer un comportement différent.

Pour empêcher nginx de démarrer automatiquement …  
**>> sudo systemctl disable nginx.service**

Pour réactiver le démarrage automatique …  
**>> sudo systemctl enable nginx.service**

### Commande nginx

nginx propose une série de commande afin de gérer son propre démon (*daemon*).  
Voici les commandes de base …

Pour afficher la version de nginx utilisée …  
**>> sudo nginx -v  
nginx version: nginx/1.18.0**

Pour tester l'ensemble de la configuration de nginx …  
**>> sudo nginx -t  
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  
nginx: configuration file /etc/nginx/nginx.conf test is successful**

Pour le démarrage de nginx, on peut utiliser la commande systemctl comme la plupart des services.

Une fois nginx démarré, il peut être contrôlé en appelant l'exécutable nginx avec le paramètre -s …  
La syntaxe de la commande est la suivante …  
**>> sudo nginx -s <signal>**

Où le signal peut être l'un des suivants …

* **stop**  
  Arrêt rapide ;
* **quit**  
  Arrêt en douceur ;
* **reload**  
  Rechargement du fichier de configuration ;
* **reopen**  
  Réouvrir les fichiers journaux

Par exemple, pour arrêter les processus nginx en attendant que les processus de travail aient fini de traiter les demandes en cours …  
**>> sudo nginx -s quit**

## Mode de fonctionnement sommaire

Lorsqu'il démarre, nginx charge les fichiers de configuration et se met en attente de requêtes sur les interfaces réseaux. On dit qu'il **écoute** (*listen* en anglais) **certains ports**.

Lorsqu'on utilise un fureteur web, que l'on clique sur un lien ou qu'on rentre directement une URL dans la barre d'adresse, on effectue une requête …

* Le fureteur résout le nom de domaine (il obtient l'adresse IP du serveur) ;
* Il envoie une requête HTTP avec la méthode GET à l'adresse IP du serveur sur le port 80   
  (ou HTTPS sur le port 443) pour lui demander de retourner un contenu particulier ;
* Le serveur HTTP reçoit la requête, et en fonction de divers paramètres   
  (URL appelée, configuration du serveur, ...), va chercher un contenu dans un fichier ou lance un script qui va générer un contenu ;
* Le serveur renvoie ce contenu à l'adresse IP du fureteur ;
* Le fureteur traite le contenu et le rend accessible à l'internaute.

Voilà comment fonctionne grossièrement un serveur web.

## Mise en place d’un site web sécurisé

Les **fichiers qui seront livré par le serveur** se situent dans le répertoire **sous /var/www/html**.

Par défaut, **nginx s'attend à ce que les fichiers d’un site web se retrouvent dans un répertoire spécifique** (répertoire qui peut varier).   
On détermine cet emplacement dans le fichier de configuration du serveur.

### Création du certificat et des clés

Le répertoire /etc/ssl/certs, qui peut être utilisé pour contenir le certificat public.   
Ce dernier doit déjà exister sur le serveur.

On crée le répertoire /etc/ssl/private, répertoire qui va contenir le fichier de clé privée.   
Étant donné que le secret de cette clé est essentiel pour la sécurité, de modifier les autorisations pour empêcher tout accès non autorisé …  
**>> sudo mkdir --verbose /etc/nginx/certs  
mkdir: création du répertoire '/etc/nginx/certs'  
>> sudo chmod --verbose 700 /etc/nginx/certs  
Le mode de '/etc/nginx/certs' a été modifié de 0755 (rwxr-xr-x) en 0700 (rwx------)**

La prochaine est la création d’une paire de clé et d’un certificat auto-signé avec OpenSSL en une seule commande …  
**>> sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/nginx/certs/nginx.key   
 -out /etc/nginx/certs/nginx.crt**

openssl propose une série de questions afin de personnaliser le certificat.

Voici en détail les options de cette commande …

* **openssl**  
  Outil permettant de créer et gérer des certificats, des clés et d'autres fichiers OpenSSL ;
* **req**  
  Cette sous-commande spécifie la demande de signature de certificat (CSR) X.509.   
  La norme X.509 est une norme d'infrastructure à clé publique à laquelle SSL et TLS adhèrent pour la gestion de ses clés et de ses certificats ;
* **-x509**  
  Cette option indique à OpenSSL une demande de création de certificat auto-signé au lieu de générer une demande de signature de certificat (l’option par défaut) ;
* **-nodes**  
  Cette option demande à OpenSSL d’ignorer la demande d’une phrase secrète.  
  Nginx doit pouvoir lire le fichier, sans intervention de l'utilisateur, au démarrage du serveur.   
  L’absence de l’option -nodes oblige la saisie une phrase secrète après chaque redémarrage ;
* -**days 365**  
  Cette option définit la durée pendant laquelle le certificat sera considéré comme valide ;
* **-newkey rsa:2048**  
  Cette option spécifie de générer un nouveau certificat et une nouvelle clé en même temps.   
  rsa:2048 lui dit de créer une clé RSA d'une longueur de 2048 bits ;
* **-keyout**  
  Cette option indique à OpenSSL l’emplacement où sauvegarder la clé privée ;
* **-out**  
  Cette option indique à OpenSSL l’emplacement où sauvegarder le certificat.

Ces options créeront à la fois un fichier de clé et un certificat.   
Les réponses aux questions vont permettre d'intégrer correctement les informations dans le certificat.

On répond à ces questions de manière appropriée.   
La ligne la plus importante est celle qui demande le nom commun (par exemple, le FQDN du serveur ou son nom).

**Country Name (2 letter code) [AU]:CA  
State or Province Name (full name) [Some-State]:Quebec  
Locality Name (eg, city) []:Montreal  
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Web-Maisonneuve  
Organizational Unit Name (eg, section) []:Formation  
Common Name (e.g. server FQDN or YOUR name) []:www.monsite.info  
Email Address []:webmaster@monsite.info**

Les deux fichiers qui ont été créés seront placés dans le répertoire /etc/ngginx/certs.

**Remarque** …  
Pour vérifier le contenu du certificat généré …  
**>> sudo** **openssl x509 -noout -text -in /etc/nginx/certs/nginx.crt  
Certificate:  
 Data:  
 Version: 3 (0x2)  
 Serial Number:  
 4d:64:de:b8:f4:f1:2a:9c:7d:60:c5:c7:1a:06:e9:d4:d7:24:f4:1b  
 Signature Algorithm: sha256WithRSAEncryption  
 Issuer: C = CA, ST = Quebec, L = Montreal, O = Web-Maisonneuve, OU = Formation, CN =** [**www.monsite.info**](http://www.monsite.info)**, emailAddress =** [**webmaster@monsite.info**](mailto:webmaster@monsite.info) **Validity  
 Not Before: Nov 13 18:07:02 2021 GMT  
 Not After : Nov 13 18:07:02 2022 GMT  
Subject: C = CA, ST = Quebec, L = Montreal, O = Web-Maisonneuve, OU = Formation, CN = www.monsite.info, emailAddress =** [**webmaster@monsite.info**](mailto:webmaster@monsite.info) **Subject Public Key Info:  
 Public Key Algorithm: rsaEncryption  
 RSA Public-Key: (2048 bit)  
 Modulus:  
 00:d0:4f:d5:9b:13:f4:f9:db:ed:82:9f:c3:59:c1:  
 bd:58:44:39:dc:56:0c:73:41:89:61:8f:d5:cd:83:  
 5a:cd:8a:c4:42:38:c3:63:79:64:f6:6a:47:37:48:  
 e7:6b:c5:23:d7:ff:62:fb:3f:4a:c1:0c:a1:97:38:  
 a5:61:b1:60:59:a0:2e:85:33:17:89:b9:91:fe:6d:  
 79:93:d8:db:b1:d4:e4:84:d8:9c:57:48:75:91:c6:  
 64:49:eb:c9:4e:9a:5f:4c:4c:52:e7:6d:10:44:fe:  
 40:a7:5a:d7:6d:e9:31:11:7d:3c:86:f8:7c:ad:a3:  
 11:c0:d5:6b:f3:7a:64:25:bb:e1:ae:16:1f:81:37:  
 bf:58:27:b9:24:79:67:14:51:dc:f6:65:ec:ea:5b:  
 8d:23:95:64:9c:c9:7f:7b:2a:f2:a0:26:f4:7f:3d:  
 91:71:c2:4f:ec:3e:f9:9e:77:82:70:66:58:a3:b1:  
 0a:63:6f:19:fb:10:39:30:99:90:a6:e8:b7:d9:b4:  
 3d:f2:04:da:ea:8b:09:8a:0a:0c:ac:75:43:44:3e:  
 25:7c:5b:53:32:23:2f:6a:77:16:b6:75:7e:d0:ea:  
 02:3c:fc:5d:2e:ae:70:01:59:ce:b8:8d:db:aa:07:  
 4d:03:7e:4f:74:bf:24:85:bc:f7:a8:b5:f0:42:0e:  
 89:79  
…**

### Secret de transmission parfait

Lors de l’utilisation d’OpenSSL, il est intéressant de créer également un groupe Diffie-Hellman fort.

Ce groupe est utilisé pour négocier le secret de transmission parfait (*Perfect Forward Secrecy*) avec les clients.  
Le secret de transmission parfait signifie qu'un élément d'un système de chiffrement modifie automatiquement et fréquemment les clés qu'il utilise afin de chiffrer et déchiffrer les informations. Ainsi si la dernière clé est compromise, elle n'expose qu'une petite partie des données sensibles de l'utilisateur.

On peut créer ce secret de transmission parfait …  
**>> sudo openssl dhparam -out /etc/nginx/certs/dhparam.pem 2048**

L’exécution de cette commande peut prendre quelques minutes, mais une fois terminé, on aura un groupe DH puissant dans /etc/nginx/certs/dhparam.pem qu’il sera possible d’utiliser dans la configuration du site Web …  
**>> ls -l /etc/nginx/certs  
total 12  
-rw-r--r-- 1 root root 424 13 nov 09:18 dhparam.pem  
-rw------- 1 root root 1704 12 nov 16:06 monsite.info.key  
-rw-r--r-- 1 root root 1566 12 nov 16:08 monsite.info.pem**

### Modification du fichier index.html

Afin de créer un site personnalisé, on remplace le fichier /var/www/html/index.html.

**Remarque** …  
Si ce fichier existe, on peut en créer une sauvegarde …  
**>> sudo mv /var/www/html/index.html /var/www/html/index.html.original**

On crée la page d’accueil du site …  
**>> sudo nano /var/www/html/index.html**

On y ajoute le contenu suivant …  
**<!DOCTYPE html>**

**<html>**

**<head>**

**<!-- En-tête de la page -->**

**<meta charset="utf-8" />**

**<title>En construction</title>**

**</head>**

**<body>**

**<!-- Corps de la page -->**

**Site en construction**

**</body>**

**</html>**

**Remarque** …  
Il est aussi possible de copier dans le répertoire /var/www/monsite.info un site Web existant.

## Configuration pour l’utilisation de SSL

La prochaine étape est la création d’un nouveau fichier dans le répertoire /etc/nginx/sites-available/ afin de configurer un bloc server qui répondra aux requêtes HTTPS à l'aide des fichiers de certificat préalablement cré.   
Il sera ensuite possible de configurer ce bloc de serveur par défaut afin de rediriger les requêtes HTTP vers HTTPS.

### Création du bloc server TLS/SSL

On doit informer nginx de l’existence du site Web, de l’utilisation de SSL/TLS et de la manière de le servir.  
**>> cd /etc/nginx/**

C'est dans ce répertoire que se trouvent les fichiers de configuration nginx.

Les deux répertoires qui intéressant à ce stade sont sites-available et sites-enabled …

* **sites-available**  
  contient des fichiers de configuration individuels pour tous les sites Web statiques possibles ;
* **sites-enabled**   
  contient des liens vers les fichiers de configuration que nginx lira et exécutera réellement.

La prochaine étape est de modifier le fichier de configuration existant nommé **default** ou de **créer un fichier de configuration dans le répertoire sites-available** et de **créer un lien symbolique (un pointeur) de ce fichier vers le répertoire sites-enabled** afin d’indiquer à nginx que ce site est activé.

On modifie le fichier /etc/nginx/sites-available/ssl.conf …

Le fichier débute par l’ouverture d’un bloc server.   
Par défaut, les connexions TLS/SSL utilisent le port 443, cela devrait donc être le port d'écoute.   
La directive server\_name doit être défini sur le nom de domaine ou l'adresse IP du serveur qui a été utilisé comme nom commun lors de la génération du certificat.

Ensuite, on utilise les directives ssl\_certificate, ssl\_certificate\_key et ssl\_dhparam afin de définir l'emplacement des fichiers SSL générés.

**server {**

**listen 80 default\_server;**

**listen [::]:80 default\_server;  
 server\_name \_;  
 return 301 https://$host$request\_uri;**

**}**

**server {  
 listen 443 http2 ssl;  
 listen [::]:443 http2 ssl;  
 root /var/www/html;  
 # Add index.php to the list if you are using PHP  
 index index.html index.htm index.nginx-debian.html;  
 server\_name \_;**

**ssl\_certificate /etc/nginx/certs/nginx.crt;  
 ssl\_certificate\_key /etc/nginx/certs/nginx.key;  
 ssl\_dhparam /etc/nginx/certs/dhparam.pem;**

**location / {  
 # First attempt to serve request as file, then  
 # as directory, then fall back to displaying a 404.  
 try\_files $uri $uri/ =404;  
 }  
}**

### Création d’une redirection de HTTP vers HTTPS (optionel)

Par défaut, nginx répond avec un contenu chiffré pour les requêtes sur le port 443 (https:// ...) et répond avec un contenu en clair pour les requêtes sur le port 80 (http:// ...).

Le site que l’on vient de configurer propose un chiffrement sans toutefois en imposer son utilisation.   
Cela peut convenir dans certains cas d'utilisation, mais il est généralement préférable d'exiger un le chiffrement des communications.   
Ceci est particulièrement important lorsque des données confidentielles, telles que des mots de passe, peuvent être transférées entre le fureteur et le serveur.

Afin de s’assurer que nginx réponde en tout temps sur le port TCP 443 (même si la requête est adressée au port TCP 80), on ajoute au premier bloc server la ligne suivante …  
**return 301 https://$host$request\_uri/;**

Cette directive configure le bloc server HTTP sur le port 80 (par défaut) afin qu’il redirige les demandes entrantes vers le bloc de serveur HTTPS.

### Publication du site Web

Maintenant que le fichier de configuration du site est créé, on doit créer un lien symbolique vers le   
répertoire sites-enabled pour indiquer à nginx de l'activer.

La syntaxe est la suivante …  
**>> ln -s <Fichier source> <fichier destination>**

Dans cet exemple, la syntaxe ressemble à …  
**>> sudo ln -s /etc/nginx/sites-available/default /etc/nginx/sites-enabled/default**

### Activation des modifications

Après avoir créé ou modifier un ou plusieurs fichiers de configuration, on doit redémarrer le service nginx afin qu’il tienne compte des changements.

Toutefois, au préalable, il est possible de vérifier que les modifications apportées ne comportent pas des erreurs de syntaxe …  
**>> sudo nano nginx -t  
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  
nginx: configuration file /etc/nginx/nginx.conf test is successful**

Et on redémarre le service nginx …  
**>> sudo systemctl restart nginx.service**

Le processus Nginx est redémarré, mettant en œuvre les paramètres SSL nouvellement configurés.

**Remarque** …  
On peut également utiliser la commande suivante …  
**>> sudo nginx -s restart**

## Chiffrement fort

Afin de s’assurer que nginx n’utilise que des algorithmes identifiés commeforts, il est possible de le prpéciser dans le fichier de configuration /etc/nginx/sites-available/default.

Pour que le serveur Nginx n'accepte que les protocoles TLS 1.2 et TLS 1.3, protocoles identifiés comme forts et qu’il n’accepte que des algorithmes de chiffrement forts …  
**>> sudo nano /etc/nginx/sites-available/default**

On ajoute les lignes suivantes …  
**server {  
 listen 443 http2 ssl;  
 listen [::]:443 http2 ssl;  
 root /var/www/html;  
…  
 ssl\_certificate /etc/nginx/certs/nginx.crt;  
 ssl\_certificate\_key /etc/nginx/certs/nginx.key;  
 ssl\_dhparam /etc/nginx/certs/dhparam.pem;  
  
 ssl\_protocols TLSv1.2 TLSv1.3;  
 ssl\_prefer\_server\_ciphers off;  
 ssl\_ciphers "ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM- SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20- POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-SHA";  
  
 location / {  
…  
 }  
}**

On doit redémarrer le service nginx …  
**>> sudo systemctl restart nginx.service**

### Verification du bon fonctionnement

On peut vérifier si le site répond positivement ou négativement aux requêtes utilisant protocoles et algorithmes forts.

Depuis un hôte distant …  
**>> openssl s\_client -tls1\_1 -connect <Adresse ou nom DNS du serveur>:443 2> /dev/null | grep -i -E "cipher|protocol"  
>> openssl s\_client -tls1\_1 -connect 192.168.1.100:443 2> /dev/null | grep -i -E "cipher|protocol"  
New, (NONE), Cipher is (NONE)  
 Protocol : TLSv1.1  
 Cipher : 0000**La sortie de la commande indique que le serveur n’accepte pas le protocole TLSv1.1.

Depuis un hôte distant …  
**>> openssl s\_client -tls1\_2 -connect <Adresse ou nom DNS du serveur>:443 2> /dev/null | grep -i -E "cipher|protocol"  
>> openssl s\_client -tls1\_2 -connect 192.168.1.100:443 2> /dev/null | grep -i -E "cipher|protocol"  
New, TLSv1.2, Cipher is ECDHE-RSA-AES256-GCM-SHA384  
 Protocol : TLSv1.2  
 Cipher : ECDHE-RSA-AES256-GCM-SHA384**   
La sortie de la commande indique que le serveur n’accepte pas le protocole TLSv1.1.

Depuis un hôte distant …  
**>> openssl s\_client -tls1\_3 -connect <Adresse ou nom DNS du serveur>:443 2> /dev/null | grep -i -E "cipher|protocol"  
>> openssl s\_client -tls1\_3 -connect 192.168.1.100:443 2> /dev/null | grep -i -E "cipher|protocol"  
New, TLSv1.3, Cipher is TLS\_AES\_256\_GCM\_SHA384**

On peut constater l’activation del'utilisation de TLS 1.2 et TLS 1.3 sur le serveur Nginx.

## Autres fonctionnalités

### HTTP/2

Le protocole de transfert hypertexte, ou HTTP, est un protocole d'application qui est la norme de facto pour la communication sur le World Wide Web depuis son invention en 1989.

Depuis la sortie de HTTP/1.1 en 1997 jusqu'à récemment, il y a eu peu de révisions au protocole.

Mais en 2015, une version repensée appelée HTTP/2 a été utilisée, qui offrait plusieurs méthodes pour réduire la latence, en particulier lorsqu'il s'agissait de plates-formes mobiles et de graphiques et de vidéos gourmands en serveurs.

Pour utiliser le protocole http/2 sur un serveur nginx, dans la zone HTTPS, on ajoute HTTP2 à l'élément de configuration LISTEN …  
**>> sudo nano /etc/nginx/sites-available/default**

On modifie les lignes suivantes …  
**…  
listen 443 ssl http2;  
listen [::]:443 ssl http2;**

On doit redémarrer le service nginx …  
**>> sudo systemctl restart nginx.service**

**Remarque** …  
Il est important de s’assurer que l’on dispose d’une site HTTPS configuré sur le serveur nginx.   
HTTPS est une exigence pour HTTP2.

### Vérification de l’utilisation de HTTP/2

Il est possible de vérifier si le serveur peut travailler avec le protocole HTTP/2.

Depuis un hôte distant …  
**>> curl -I https://<Adresse IP ou Nom DNS>  
>> curl -I https://192.168.1.100  
HTTP/2 200  
server: nginx/1.18.0 (Debian)  
…**

La première ligne affichée confirme l'utilisation du protocole HTTP2.

### HTTP Strict Transport Security (HSTS)

Une politique HSTS est publiée en envoyant l'en-tête de réponse HTTP suivant à partir de sites Web sécurisés (HTTPS) …  
**Strict-Transport-Security: max-age=31536000**

Lorsqu'un fureteur voit cet en-tête à partir d'un site Web HTTPS, il apprend que ce domaine ne doit être accessible qu'à l'aide de HTTPS (SSL ou TLS).   
Il met alors en cache ces informations pour la période max-age (généralement 31 536 000 secondes, soit environ 1 an).

Le paramètre facultatif includeSubDomains indique au fureteur que la stratégie HSTS s'applique également à tous les sous-domaines du domaine actuel.

Pour activer HSTS sur un serveur ngonx…  
**>> sudo nano /etc/nginx/sites-available/default**

On ajoute les lignes suivantes …

**server {  
 listen 443 ssl default\_server;  
 listen [::]:443 ssl default\_server;  
...  
 add\_header Strict-Transport-Security "max-age=31536000; ncludeSubDomains";  
...**

On doit redémarrer le service nginx …  
**>> sudo systemctl restart nginx.service**

**Remarque** …  
Il est important de s’assurer que l’on dispose d’une site HTTPS configuré sur le serveur nginx.   
HTTPS est une exigence pour HTTP2.

### Vérification de l’utilisation de HSTS

Il est possible de vérifier si le serveur peut travailler avec le protocole HSTS.

Depuis un hôte distant …  
**>> curl -I https://<Adresse IP ou Nom DNS>  
>> curl -I https://192.168.1.100  
HTTP/2 200  
server: nginx/1.18.0 (Debian)  
…  
Strict-Transport-Security: max-age=31557600  
…**