

- **Pour la création du certificat https auto-signé avec redirection automatique de http vers https nous avons écrit ces commandes sur notre serveur :**

**Définition :**

Un **certificat HTTPS** auto-signé avec redirection automatique de HTTP vers HTTPS est un certificat de sécurité généré localement qui permet d'établir une connexion sécurisée entre un serveur et un navigateur, tandis que la redirection automatique assure que toutes les requêtes HTTP sont redirigées vers une connexion HTTPS sécurisée.

**Étape 1 - Création du certificat SSL**

- Pour sécuriser les communications via TLS/SSL, on a créé un certificat SSL comprenant une clé privée pour chiffrer le contenu et un certificat public partagé pour déchiffrer le contenu, ce qui a été réalisé en une seule commande avec OpenSSL.

**openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/apache-selfsigned.key -out /etc/ssl/certs/apache-selfsigned.crt**

Cette commande crée un certificat auto-signé pour un serveur Apache en générant une clé privée et un certificat public avec une durée de validité de 365 jours, et les enregistre respectivement dans les répertoires /etc/ssl/private/ et /etc/ssl/certs/.

**Étape 2 - Configuration d'Apache pour utiliser SSL**

- Nous avons créé un extrait de configuration Apache avec des paramètres de chiffrement fort :

**nano /etc/apache2/conf-available/ssl-params.conf**

Cette commande ouvre le fichier "ssl-params.conf" situé dans le répertoire "/etc/apache2/conf-available/" en utilisant l'éditeur de texte "nano" avec les privilèges de superutilisateur (sudo).

**Nous y avons collés la configuration suivante :**

```
SSLCipherSuite ECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH
SSLProtocol All -SSLv2 -SSLv3 -TLSv1 -TLSv1.1
SSLHonorCipherOrder On
# Disable preloading HSTS for now. You can use the commented out header line
that includes
# the "preload" directive if you understand the implications.
# Header always set Strict-Transport-Security "max-age=63072000;
includeSubDomains; preload"
Header always set X-Frame-Options DENY
Header always set X-Content-Type-Options nosniff
# Requires Apache >= 2.4
SSLCompression off
SSLUseStapling on
SSLStaplingCache "shmcb:logs/stapling-cache(150000)"
# Requires Apache >= 2.4.11
SSLSessionTickets Off
```

- **Nous avons faits quelques modifications du fichier d'hôte virtuel Apache SSL par défaut :**

**cp /etc/apache2/sites-available/default-ssl.conf /etc/apache2/sites-available/default-ssl.conf.bak**

Cette commande effectue une copie du fichier de configuration "default-ssl.conf" situé dans le répertoire "/etc/apache2/sites-available/" vers un fichier de sauvegarde "default-ssl.conf.bak" dans le même répertoire.

**nano /etc/apache2/sites-available/default-ssl.conf**

Cette commande ouvre le fichier de configuration "**default-ssl.conf**" du serveur Apache en utilisant l'éditeur de texte "**nano**".

- Nous y avons modifiés quelques lignes et voici à quoi il ressemble :

```
IfModule mod_ssl.c>
    <VirtualHost _default_:443>
        ServerAdmin massonclément3@gmail.com
        ServerName ap23

        DocumentRoot /var/www/html

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        SSLEngine on

        SSLCertificateFile      /etc/ssl/certs/apache-selfsigned.crt
        SSLCertificateKeyFile   /etc/ssl/private/apache-selfsigned.key

        <FilesMatch \"\.(cgi|shtml|phtml|php)$\">
            SSLOptions +StdEnvVars
        </FilesMatch>
        <Directory /usr/lib/cgi-bin>
            SSLOptions +StdEnvVars
        </Directory>

    </VirtualHost>
</IfModule>
```

- Par recommandation, nous avons fait quelques modification du fichier hôte HTTP pour rediriger vers HTTPS :

**nano /etc/apache2/sites-available/000-default.conf**

Cette commande ouvre le fichier de configuration "**000-default.conf**" du répertoire "**/etc/apache2/sites-available**" dans l'éditeur de texte "**nano**" pour permettre la modification des paramètres de configuration du site par défaut d'Apache.

**On a ajouté cette ligne :**

```
<VirtualHost *:80>
    . . .

    Redirect "/" "https://ap23/"

    . . .
</VirtualHost>
```

### **Étape 3 - Réglage du pare-feu**

Cette étape nous a permis de mettre en place une couche de filtrage des ports (avec ufw, iptables, ferm ou autre ) et de faire un diagnostic des ports ouverts, et notamment de mettre en place un pare-feu.

- Nous installons d'abord le paquet ufw : **apt install ufw**

**Définition :**

**UFW** (Uncomplicated Firewall) est un outil simple de configuration du pare-feu sur les systèmes Linux, permettant de gérer les règles de sécurité réseau de manière conviviale.

- On lance la commande suivante : **ufw app list**

Elle affiche la liste des applications et des profils de pare-feu disponibles gérés par UFW (Uncomplicated Firewall) sur le système.

Nous devons voir ceci : (ce qui est le cas).

```
Output
Available applications:
```

```
. . .
WWW
WWW Cache
WWW Full
WWW Secure
. . .
```

Nous avons regardé que ufw est bien activé avec : **ufw status**

```
Status: active
```

To	Action	From
--	-----	----
22/tcp	ALLOW	Anywhere
80/tcp	ALLOW	Anywhere
25/tcp	ALLOW	Anywhere
WWW Full	ALLOW	Anywhere
22	ALLOW	Anywhere
2222	ALLOW	Anywhere
80	ALLOW	Anywhere
443	ALLOW	Anywhere
6000:6007/tcp	ALLOW	Anywhere
6000:6007/udp	ALLOW	Anywhere
3306	DENY	Anywhere
22/tcp (v6)	ALLOW	Anywhere (v6)
80/tcp (v6)	ALLOW	Anywhere (v6)
25/tcp (v6)	ALLOW	Anywhere (v6)
WWW Full (v6)	ALLOW	Anywhere (v6)
22 (v6)	ALLOW	Anywhere (v6)
2222 (v6)	ALLOW	Anywhere (v6)
80 (v6)	ALLOW	Anywhere (v6)
443 (v6)	ALLOW	Anywhere (v6)
6000:6007/tcp (v6)	ALLOW	Anywhere (v6)
6000:6007/udp (v6)	ALLOW	Anywhere (v6)
3306 (v6)	DENY	Anywhere (v6)

**Étape 4 - Activation des modifications dans Apache**

**a2enmod ssl**

**a2enmod headers**

La commande "**a2enmod ssl**" active le module SSL dans Apache, tandis que la commande "**a2enmod headers**" active le module Headers dans Apache.

**a2ensite default-ssl**  
**a2enconf ssl-params**

La commande "**a2ensite default-ssl**" active le site par défaut en utilisant la configuration SSL dans Apache.

La commande "**a2enconf ssl-params**" active la configuration du module SSL dans Apache pour utiliser les paramètres de sécurité spécifiés dans le fichier "**ssl-params.conf**".

**apache2ctl configtest**

La commande "**apache2ctl configtest**" permet de vérifier la validité et la syntaxe de la configuration du serveur Apache.

Nous avons obtenu : « **Output Syntax OK** », cela signifie qu'il n'y a pas d'erreurs de syntaxe dans nos fichiers.

Nous avons redémarré apache2 pour être sûr que les paramètres et les nouvelles configurations sont appliqués.

**systemctl restart apache2****Étape 5 - Test de chiffrement**

Dans cette étape, nous avons vérifiés que l'on peut se connecter à notre serveur depuis un navigateur web et qu'il y'a bien la redirection HTTPS :

On s'est connecté ici : <https://ap23>

Il est tout a fait normal que le navigateur nous dise que notre connexion n'est pas privé étant donné que le certificat n'est qu'auto-signé.

Lorsque l'on se connecte ici : <http://ap23> on est automatiquement rediriger vers : <https://ap23>

- **Pour permettre l'accès SSH uniquement par clé publique+ : [//////////a voir plus tard//////////](#)**
- **Pour améliorer la sécurité, on souhaite exporter les journaux d'évènements vers une machine distante.**
- **Proposer une stratégie de mise en œuvre et la réaliser :**

Nous allons exporter les journaux d'évènements de serveur ap23 (adresse IP 10.121.38.78) sur une machine distante dans un cluster **pxlab** (**nous allons créer une VM sur Proxmox qui sera hébergé sur pxlab3**), composé de :

- [pxlab1 - 172.16.0.10/24 - Dell T110-II - 32 Go - 1 To DD](#)
- [pxlab2 - 172.16.0.11/24 - Dell T110-II - 16 Go - 1 To DD](#)
- [pxlab3 - 172.16.0.12/24 - Dell T110-II - 16 Go - 1 To DD](#)

Ce qui représente actuellement 64 Go de mémoire et 3 To de stockage.

Entre le serveur **ap23** et **pxlab3**, il y'a déjà une première passerelle **gwsio5** (debian 11 routage nat avec d'un coté 192.168.0.1/24 et de l'autre 10.121.38.35) mais notre serveur ap23 est dans le réseau 10.121.38.0 et une deuxième passerelle **gwlab** (debian 11 routage avec d'un coté 10.121.38.253 et de l'autre 172.16.0.254) qui va jusqu'au pxlab3.

Pour que le serveur **ap23** puisse atteindre la machine distante pxlab3, nous allons configurer une route sur le serveur **ap23**. Cette route doit être configurée pour passer par les passerelles intermédiaires **gwsio5** et **gwlab**, afin de pouvoir acheminer le trafic vers l'adresse IP 172.16.0.12 de **pxlab3**.

1. Pour cela nous ouvrirons un terminal sur le serveur ap23.
2. On utilisera la commande suivante pour ajouter une route :

**sudo ip route add 172.16.0.12 via 10.121.38.253**

3. Cette commande indique au serveur **ap23** d'acheminer le trafic destiné à l'adresse IP 172.16.0.12 en passant par la passerelle **gwlab** dont l'adresse IP est 10.121.38.253.
4. On vérifiera que la route a été ajoutée correctement en utilisant la commande :  
**ip route show**
5. On s'assurera qu'on voit une ligne indiquant que le trafic destiné à 172.16.0.12 sera acheminé via 10.121.38.253.
6. On essayera de pinguer la machine distante **pxlab3** en utilisant la commande : **ping 172.16.0.12**