

SSL avec Apache2 sous GNU/Linux

Tout d'abord, pour ce tutoriel j'ai utilisé la documentation de Ubuntu-server accessible ici : <https://help.ubuntu.com/8.04/serverguide/C/>. Ensuite, un peu d'imagination, de test, d'erreurs et ça roule !

Voici donc un tutoriel pour mettre en place un Apache2, avec SSL actif, et comment le forcer sur certaines pages et pas d'autres.

J'ai utilisé Ubuntu-Server 8.04, ça marchera donc normalement avec d'autres versions plus récentes d'Ubuntu et Debian. Les seules modifications probables seront liées aux noms des packages.

Tout d'abord, il faut installer apache2 et ce qui va servir plus tard :

```
apt-get install apache2 libapache2-mod-auth-mysql openssl
```

Pour ceux qui utilisent Ubuntu-server, vous pouvez aussi lancer la commande **tasksel**, et cocher "LAMP". Ce choix installera apache2, Mysql et PHP avec pas mal de packages qui pourraient servir.

En ce qui concerne les sites que vous installerez, je vous conseille fortement de les mettre dans un répertoire uniquement prévu à cet effet, et sur une partition différente du système, ça peut aider en cas de besoin de tout réinstaller, ça vous évitera de formater la partition de données.

Dans ce tutoriel, on utilisera par exemple le répertoire **/sites** comme répertoire de base. Je ne détaillerai pas le fonctionnement des VirtualHosts sur Apache, ni les alias, mais si vous suivez bien, vous comprendrez comment ça marche, c'est assez simple ;).

Tout ce passe dans le fichier **default**, que vous trouverez dans le répertoire de votre apache2, soit **/etc/apache2/site-available**. Ce fichier contient les déclarations des sites, les chemins d'accès, et les liens accessibles par le web.

Le but est de déclarer deux « **VirtualHost** », un pour le http sur port 80, un pour le https sur port 443.

A la base un Virtualhost permet de contrôler un site pour un domaine donné, mais on peut aussi en créer un qui les contrôle tous. Chaque virtualhost donne le site par défaut du serveur, les alias, et les autorisations de chacun.

Au passage, avant d'oublier, ouvrez les ports 80 et 443 sur votre machine par le biais de votre pare-feu, ainsi que dans votre box si votre serveur est derrière ;)

Pour illustrer ce tutoriel, voici les sites exemples que l'on utilisera, le but est de les paramétrer :

- Dotclear en page racine du domaine, accessible sur <http://monserveur>
- L'administration de dotclear sur <https://monserveur/admin>
- Un blog piwigo sur <http://monserveur/blog>
- Et un forum phpbb2 sur <http://monserveur/forum>

Le but est de faire ça pour chacun :

- Dotclear accessible en http et https (pas obligatoire pour ce dernier, mais ça coute rien donc je le laisse)
- Administration de dotclear autorisée en https, refusée en http (pour ne pas diffuser le login/mdp en clair sur le net).
- Le blog en http et https (l'administration du blog se fait par la même page que l'accueil, on peut donc imaginer que l'on passe manuellement en https quand on en a besoin, sinon l'accès au public se fait par http).
- Le forum accessible en http et https, à configurer dans son administration pour autoriser l'https à la connexion d'un utilisateur.

Ouvrez le fichier **default** avec votre éditeur préféré, nano, MC, ou VI pour les courageux :

```
nano /etc/apache2/site-available/default
```

Voici ce que donnerai le fichier :

```
#On declare le virtualhost http, qui concerne l'ensemble du domaine
NameVirtualHost *:80

#de meme pour https
NameVirtualHost *:443

#Début du virtualHost http
<VirtualHost *:80>

#Déclarez ici l'admin du site avec son mail, qui sera accessible sur les pages
#d'erreurs.
    ServerAdmin admin@votredomaine.com

#Ci après le chemin de base vers le site web par défaut - /sites/www où est
#installé mon dotclear, qui doit être le site par défaut.
    DocumentRoot /sites/www

#La declaration d'après était sur apache par défaut, j'ai pas touché :D
    <Directory />
        Options FollowSymLinks
        AllowOverride None
    </Directory>

#Déclarez les droits et options de la racine du site:
    <Directory /sites/www>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride None
        Order allow,deny
        allow from all #on autorise tout le monde sinon c'est pas drôle
    </Directory>

#De meme que tout à l'heure, la suite j'ai pas touché :p
    ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
    <Directory "/usr/lib/cgi-bin">
        AllowOverride None
        Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
        Order allow,deny
        Allow from all
    </Directory>

    ErrorLog /var/log/apache2/error.log
```

```
    LogLevel warn

    CustomLog /var/log/apache2/access.log combined
    ServerSignature On

Alias /doc/ "/usr/share/doc/"
<Directory "/usr/share/doc/">
    Options Indexes MultiViews FollowSymLinks
    AllowOverride None
    Order deny,allow
    Deny from all
    Allow from 127.0.0.0/255.0.0.0 ::1/128
</Directory>

#Passons maintenant aux déclarations des alias
#Dans cet exemple, le dotclear installé dans /sites/www, a un panneau
#d'administration accessible dans /sites/www/admin, à la base on n'a pas besoin
#de le déclarer, mais pour bloquer son accès en http il faut le faire. On
#l'autorisera uniquement en https pour ne pas transférer les mots de passe en
#clair ;) Vous pouvez utiliser cette même méthode pour un autre site qui
#possède un panneau de configuration via le web, adaptez juste le chemin ;)

    Alias /admin "/sites/www/admin/"
    <Directory "/sites/www/admin">
        deny from all #on n'autorise pas en http
    </Directory>

#Je possède également un blog, je veux autoriser son accès en http.
#Déclarons-le comme tel. Il est installé dans le répertoire /sites/blog.

    Alias /blog "/sites/blog/"
    <Directory "/sites/blog">
        allow from all
    </Directory>

#J'ai également un forum dans /sites/forum, je veux qu'il soit autorisé en
http.

    Alias /forum "/sites/forum/"
    <Directory "/sites/forum">
        allow from all
    </Directory>

#J'aurais pu continuer avec les autres sites que je possède, mais c'est la même
#chose que précédemment. Terminons donc la configuration de ce virtualhost, en
#fermant les balises.

</VirtualHost>

#Créons le deuxième virtualhost pour https :
<VirtualHost *:443>

#Activons pour celui-ci le SSL, on configurera le module apache et les
#certificats nécessaires après.
SSLEngine On
SSLOptions +StrictRequire
#vous pourrez changer les chemins si nécessaire - voir fin tuto
SSLCertificateFile /etc/ssl/certs/server.crt
SSLCertificateKeyFile /etc/ssl/private/server.key
```

```
#Comme pour tout à l'heure, on configure le virtualhost :

    ServerAdmin admin@votredomaine.com
    DocumentRoot /sites/www/

#On donne les options à la racine du site
<Directory /sites/www/>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    allow from all #j'autorise l'accès au site racine en https
</Directory>

#Créons les mêmes alias que tout à l'heure, mais changeons les droits comme
#nécessaire.

#On avait bloqué l'admin du dotclear en http, autorisons le en https.

    Alias /admin "/sites/www/admin/"
    <Directory "/sites/www/admin">
        allow from all
    </Directory>

#J'autorise également le blog en https, celui-ci possède un compte admin pour
#le configurer, mais sur la même page que le blog, on pourra donc passer
#manuellement en https en cas de besoin de l'administrer pour que le mot de
#passe ne soit pas en clair sur le net.

    Alias /blog "/sites/blog/"
    <Directory "/sites/blog">
        allow from all
    </Directory>

#Pour le forum, activons le https, à vous après de configurer le forum pour
#qu'il puisse passer en https automatiquement à la connexion d'un utilisateur.

    Alias /forum "/sites/forum/"
    <Directory "/sites/forum">
        allow from all
    </Directory>

</VirtualHost>
```

Et voilà !

En fonction de vos sites, adaptez les alias, supprimez ceux inutiles, ceci n'est que pour illustrer les possibilités en fonction des sites.

Voici les commandes pour créer les certificats et clés pour SSL.

Tout d'abord, activons le module SSL pour apache2 (en root ou par sudo).

```
a2enmod ssl
```

Voici les étapes :

1. On crée la clé privée et publique pour l'encryptage.
2. On crée le certificat basé sur la clé publique.
3. On auto-signe notre certificat, ce n'est pas recommandé, il faudrait normalement l'envoyer au CA pour validation et signature, mais un CA reconnu officiellement coûte très cher ! Le fait qu'il soit auto-signé affichera juste un avertissement sur IE ou Firefox.

Créons les clés :

```
openssl genrsa -des3 -out server.key 1024
```

Donnez une passphrase ;)

Quand le service apache sera lancé, il aura besoin de la passphrase, puisqu'on ne peut pas se le permettre (car il faudrait la donner à chaque boot) on va faire une petite modification :

```
mv server.key server-old.key  
openssl rsa -in server-old.key -out server.key
```

Voilà. Maintenant on crée le certificat, grâce à la clé précédente :

```
openssl req -new -key server.key -out server.csr
```

Répondez à toutes les questions, ne mettez pas n'importe quoi.

On l'auto-signe (valable un an) :

```
openssl x509 -req -days 3650 -in server.csr -signkey server.key -out server.crt
```

On les mets dans les répertoires inscrits dans la configuration SSL d'apache :

```
sudo cp server.crt /etc/ssl/certs  
sudo cp server.key /etc/ssl/private
```

Relancez votre apache :

```
/etc/init.d/apache2 restart
```

Et hop ça devrait marcher !!

Essayez d'accéder à votre site web et aux pages précédemment définies !

Si vous avez des questions, un avis, des idées d'amélioration pour ce tutoriel, n'hésitez pas à me contacter via mon site <http://www.shivaserv.fr>