# Table des matières

[Table des matières 1](#_Toc59620973)

[Protection des sites web 3](#_Toc59620974)

[Objectifs 3](#_Toc59620975)

[Contrôle sur l'origine des clients 3](#_Toc59620976)

[Require 3](#_Toc59620977)

[Fournisseur d'autorisation "all" 3](#_Toc59620978)

[Fournisseur d'autorisation "ip" 3](#_Toc59620979)

[Fournisseur d'autorisation "host" 3](#_Toc59620980)

[L'intervenant DNS 4](#_Toc59620981)

[Les conteneurs d'autorisations 4](#_Toc59620982)

[Exemple de Require 5](#_Toc59620983)

[Exemple de conteneurs Require 5](#_Toc59620984)

[Le conteneur <Directory> 6](#_Toc59620985)

[Contrôle par authentification 6](#_Toc59620986)

[L'authentification http de base 6](#_Toc59620987)

[Création du fichier htpasswd 6](#_Toc59620988)

[Activation de l'autorisation 6](#_Toc59620989)

[Synoptique 7](#_Toc59620990)

[Fournisseur d'autorisation "user" 7](#_Toc59620991)

[Fournisseur d'autorisation "group" 7](#_Toc59620992)

[Autres types d'authentification 8](#_Toc59620993)

[HTTPS et les certificats 8](#_Toc59620994)

[Rappel 8](#_Toc59620995)

[Les intervenants 8](#_Toc59620996)

[Obtention d’un certificat 8](#_Toc59620997)

[Connexion SSL 9](#_Toc59620998)

[Génération d’une clé de session 9](#_Toc59620999)

[Communication SSL 9](#_Toc59621000)

[Question de confiance 10](#_Toc59621001)

[Signature 10](#_Toc59621002)

[Implémentation d'un certificat auto-signé 10](#_Toc59621003)

[Etape 1 : installer openSSL et son module Apache 10](#_Toc59621004)

[Etape 2 : création de la clé privée du serveur 10](#_Toc59621005)

[Etape 3 : création du CSR (Certificate Signing Request) 10](#_Toc59621006)

[Etape 4 : création du certificat 10](#_Toc59621007)

[Etape 5 : copie des clés et des certificats au bon endroit 10](#_Toc59621008)

[Etape 6 : connexion SSL 11](#_Toc59621009)

[Configuration du virtual.conf 11](#_Toc59621010)

[Configuration du ssl.conf 11](#_Toc59621011)

[Configuratoin du host 11](#_Toc59621012)

[Création d'une interface 11](#_Toc59621013)

[Test avec lynx 12](#_Toc59621014)

[La redirection SSL 12](#_Toc59621015)

[Fournisseur d'autorisation 13](#_Toc59621016)

# Protection des sites web

## Objectifs

Bloquer l'accès a une partie d'un site en limitant sa consultation a quelques utilisateurs privilégies ou à quelques machines.

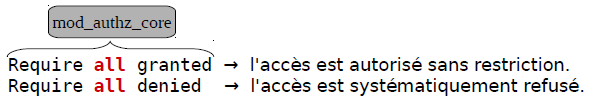
* Accéder à une porte d'administration du site (login membres …)
* Verrouiller une partie du site en construction.
* Permettre des accès seulement a un ou plusieurs réseaux de machines.

## Contrôle sur l'origine des clients

### Require

La directive Require [not] est utilisée au cours de la phase d'autorisationpour accorder ou refuser l'accès a une ressource a un client grâce à un fournisseur d'autorisation.

### Fournisseur d'autorisation "all"

Le fournisseur d'autorisation générique **all** est pris en charge par le module mod\_authz\_core  
Qui a le droit d'accéder à la ressource ? All = tout le monde, granted = autoriser, denied = refuser

1. Authentification   
   Si KO -> Stop

Si OK -> 2.

1. Autorisation  
   Si KO -> Stop  
   Si OK -> Accès à la ressource

### Fournisseur d'autorisation "ip"

Il est utilisé pour autoriser ou non une ip ou une plage ip.

Le fournisseur d'autorisation générique **ip** est pris en charge par le module mod\_authz\_host

Autoriser un accès : Require ip @IP  
Refuser un accès : Require not ip @IP

En @IP on peut avoir :

* 10/255.0.0.0 = 10.0.0.0/8
* 10 = 10.0.0.0/8
* 172.20/16 = 172.20.0.0/16
* 172.20 = 172.20.0.0/16
* 192.168.2.4

### Fournisseur d'autorisation "host"

Il est utilisé pour autoriser ou non les hôtes dont les noms correspondent ou se terminent par la chaine spécifiée.

Le fournisseur d'autorisation générique **host** est pris en charge par le module mod\_authz\_host

Autoriser un accès : Require host www.example.org  
Refuser un accès : Require not host .net example.edu  
 Si la machine fait partie du domaine .net ou example.edu, l'accès à leur ressource sera refusé

### L'intervenant DNS

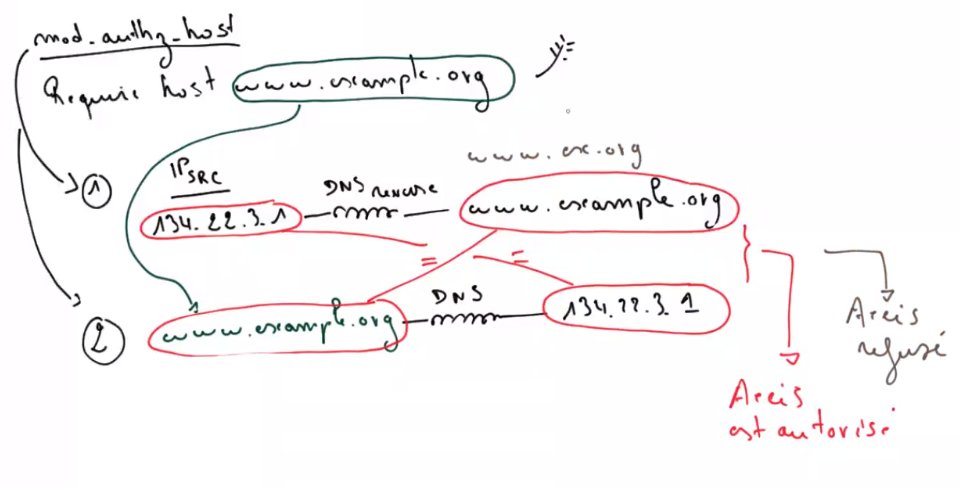
Require host www.example.org

Require not host .net example.edu

Le fournisseur d'autorisation **host** va effectuer une double recherche DNS sur l'adresse IP du client :

1. Une recherche DNS inverse sur l'adresse IP source pour trouver le nom associe
2. Puis une recherche DNS directe sur le nom renseigne dans la directive pour trouver l'adresse IP originale.

L'accès ne sera accordé que si les recherches DNS inverse et directe sont cohérentes.



### Les conteneurs d'autorisations

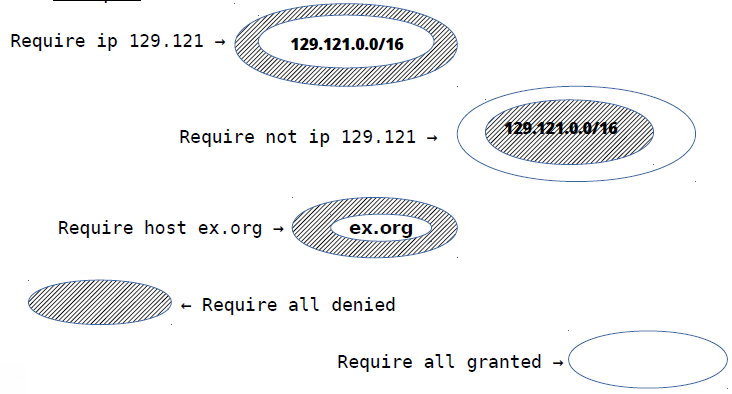
Les conteneurs d'autorisations Require font partie du module mod\_authz\_core

Dans ces conteneurs, on peut retrouver des Require accompagner de fournisseurs d'autorisation (all, ip , host…)

|  |  |
| --- | --- |
| <RequireAll>  Require …  Require …  </RequireAll> | Si toutes les directives d'autorisation sont vraies alors l'accès est autorisé. |
| <RequireAny>  Require …  Require …  </RequireAny> | Si au moins une directive d'autorisation est vraie alors l'accès est autorisé.  /!\ Require not … interdit |
| <RequireNone>  Require …  Require …  </RequireNone> | Si au moins une directive d'autorisation est vraie alors l'accès est refusé.  /!\ Require not … interdit |

Les directives d'autorisation inversées (not) sont interdites dans les directives <RequireAny> et <RequireNone>.

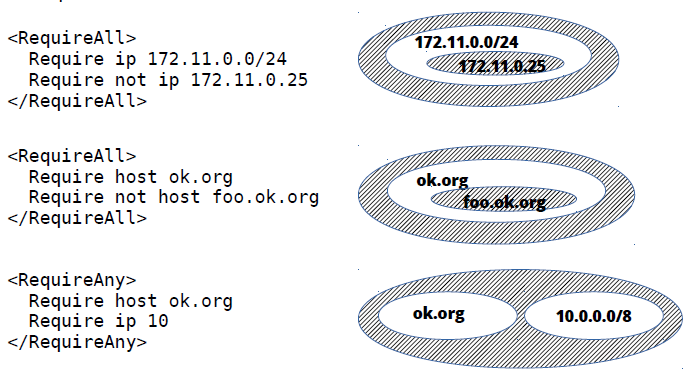
### Exemple de Require

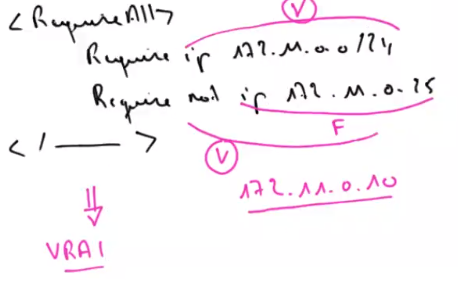


Achuré : pas accès  
Non achuré : accès

Require ip 129.121 : seulement ceux qui commence par 129.121 ont accès à la ressource  
Require not p 129.121 : seulement ceux qui commence par 129.121 n'ont pas accès à la ressource  
Require host ex.org : toutes les machines faisant partie du domaine ex.org ont accès.

### Exemple de conteneurs Require





## Le conteneur <Directory>

Il contient des directives qui ne s'appliquent qu'à celui-ci.

## Contrôle par authentification

### L'authentification http de base

Les *"users/passwords"* sont stockes dans un fichier texte.

L'utilitaire htpasswd permet de créer et mettre à jour ce fichier.

Les mots de passe de ce fichier sont cryptes à l'aide de la fonction crypt.

Cette méthode est la plus simple mais la moins sécurisée car les identifiants sont transmis chiffres en 'Base64'.

Le Base64 est très facilement déchiffrable. Tshark et Wireshark le décrypte à la volée.

### Création du fichier htpasswd

Le fichier doit se trouver en dehors du site pour plus de sécurité

Création du fichier :  
 htpasswd -c /var/www/securite/userfile toto

Ajout d'utilisateur (avec mot de passe)  
 htpasswd -b /var/www/securite/userfile albert secret

-m → pour chiffrer le mdp en MD5

-p → pas de chiffrement  
-s → pour chiffrer le mdp en SHA

### Activation de l'autorisation

<Directory rep\_a\_proteger>

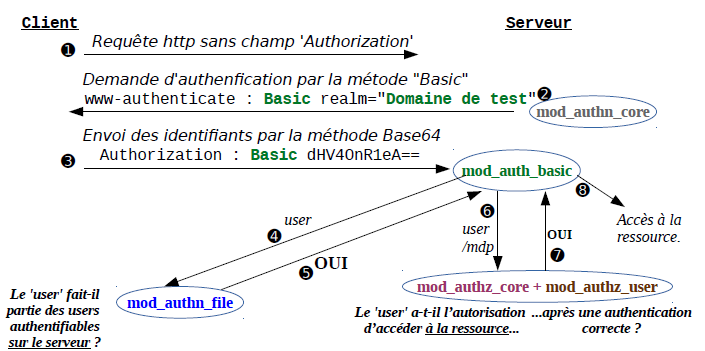
AuthName "Domaine de test" → identificateur d'autorisation  
AuthType Basic → type d'authentification (Basic ou Digest)  
AuthUserFile /var/www/securite/userfile → (1) fournisseur d'authentification  
Require valid-user → (2) fournisseur d'autorisation

</Directory>

(1) Fichier contenant la liste des utilisateurs qui ont le droit de s'authentifier.  
(2) Liste des utilisateurs qui peuvent accéder aux ressources après s'être authentifier correctement.  
Avec Require valid-user: tous les utilisateurs repris dans le fournisseur d'authentification.  
 C'est l'ensemble des login et mdp qui sont codé dans le fichier userfile

|  |  |
| --- | --- |
| Directive | Module |
| AuthName AuthType | mod\_authn\_core |
| Basic | mod\_auth\_basic |
| AuthUserFile | mod\_authn\_file |
| Require | mod\_authz\_core |
| valid-user | mod\_authz\_user |

### Synoptique



### Fournisseur d'autorisation "user"

Le fournisseur d'autorisation **user** est pris en charge par le module mod\_authz\_user

<Directory rep\_a\_proteger>

AuthName "Domaine de test"  
AuthType Basic  
AuthUserFile /var/www/securite/userfile  
**Require user toto albert**

</Directory>

Ici, seuls 'toto' et 'albert' repris dans 'userfile' ont le droit d'accéder aux ressources du répertoire moyennant une authentification valide.

### Fournisseur d'autorisation "group"

Le fournisseur d'autorisation **group** est pris en charge par le module mod\_authz\_groupfile

Soit un fichier texte /var/www/securite/groupfile contenant les groupes avec leurs utilisateurs

|  |  |
| --- | --- |
| admins:toto albert friends:toto linda associates:bruno | <Directory rep\_a\_proteger>  AuthName "Domaine de test"  AuthType Basic  **AuthUserFile /var/www/securite/userfile**  **AuthGroupFile /var/www/securite/groupfile**µ  **Require group admins friends**  </Directory> |

Ici, seuls les membres des groupes 'admins' et 'friends' ont le droit d'accéder aux ressources du répertoire moyennant une authentification valide.

La directive AuthUserFilereste obligatoire car elle contient la liste des **users qui peuvent s'authentifier.**

La directive Requiredétermine **qui a l'autorisation** d'accéder aux ressources **après authentification** réussie.

## Autres types d'authentification

Par digest : le mot de passe ne circule plus en clair mais tous les navigateurs ne gèrent pas cette méthode.

Par db : Unix DBM, MYSQL, Microsoft SQL Server, Oracle, Postgres ...

Par serveurs d'authentification : Radius, Ldap, Kerberos, Samba, Nis …

# HTTPS et les certificats

## Rappel

Pour être considère comme sur, un échange numérique doit satisfaire les objectifs suivants :

* Intégrité
* Confidentialité
* Authentification
* Non-répudiation

Pour atteindre ces objectifs, la cryptographie fait appel à différents algorithmes :

* Chiffrement symétrique
* Chiffrement asymétrique
* Hachage
* Signature
* Certificat

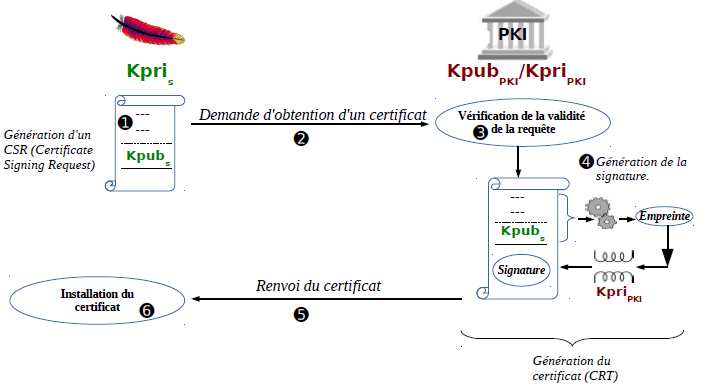
Un **certificat** permettra de s’assurer de l’authenticité de la clé publique que l’on utilise.

C’est un fichier associant la cle publique d’une personne (ou d’une machine) a son identité. Le tout étant signe par un tiers de confiance.

## Les intervenants

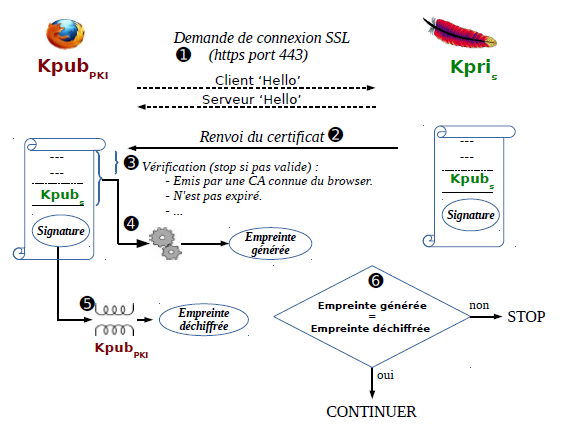
* Un client Web : firefox
* Un serveur Web : apache
* Une PKI

## Obtention d’un certificat

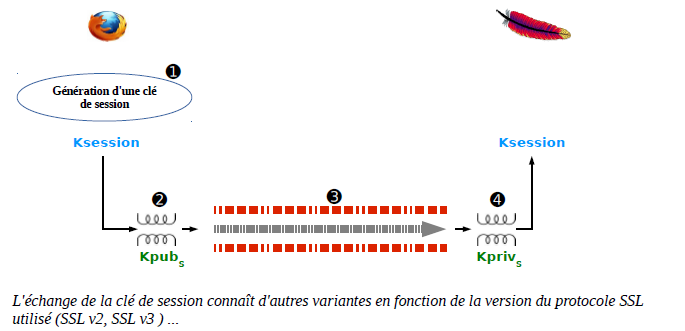
1. On génère un CSR avec la Kpub
2. On l'envoie à une PKI
3. La PKI vérifie l'identité du propriétaire
4. La PKI génère le certificat avec une signature  
   L'empreinte sera générée par la Kpub et les données
5. Le certificat est renvoyé au propriétaire
6. Le propriétaire installe le certificat

## Connexion SSL

Tout navigateur possède les Kpub des PKI

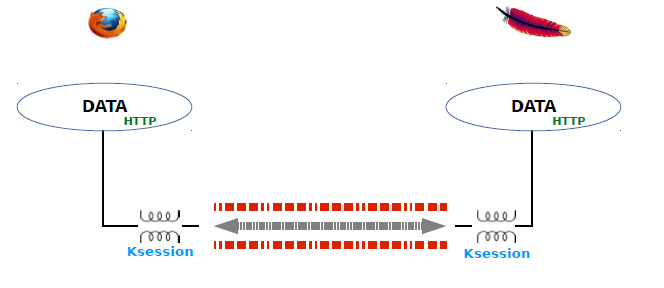
1. Le Firefox envoie au srv apache la chaine "Hello" en clair sur son port 443  
   Le server renvoie "Hello"  
   La demande de connexion SSL est initiée
2. Le srv renvoie le certificat au navigateur
3. Le navig va lire les données du certificat et vérifie si les données sont valides
4. Le navig va reprendre les 2 premiers blocs de données dans le certificat et va les hachés
5. Il va reprendre la signature et l'haché et déchiffré l'empreinte initiale
6. Il va comparer l'empreinte initiale et l'empreinte générée

## Génération d’une clé de session



1. Le navig génère une clé de session
2. Il chiffre avec la Kpub du server
3. Il est envoyé sur server
4. Le srv est déchiffré avec la Kprivé

## Communication SSL

Les données seront échangées en chiffrement SYMETRIQUE

## Question de confiance

Lorsqu'un client désire communiquer en SSL avec un serveur, il lui suffit de se procurer le certificat de ce dernier.

Ce certificat valide l'identité du serveur.

Ici, la règle de non-répudiation repose donc sur la confiance envers une PKI.

## Signature

Les certificats auto-signes :

* Signes par un serveur local.
* Pour garantir la confidentialité des échanges au sein d'un intranet.

Les certificats signes par une PKI :

* Pour garantir la confidentialité des échanges au sein de l'internet.
* Le certificateur tiers permet d'assurer à l'utilisateur que le certificat appartient bien à l'organisation à laquelle il est déclaré appartenir.

## Implémentation d'un certificat auto-signé

Nous travaillerons ici avec un certificat auto- signe pour ne pas devoir l'enregistrer auprès d'une PKI extérieure... (quelques centaines d'euros/an).

### Etape 1 : installer openSSL et son module Apache

dnf install openssl mod\_ssl -y

### Etape 2 : création de la clé privée du serveur

cd /tmp  
openssl genrsa -out secure.mysite.be.key 2048  
cat secure.mysite.be.key

genrsa : clé de type rsa  
secure.mysite.be : clé unique pour chaque site web SSL. Convention : nom du site + extension .key  
2048 : 2048 bits

### Etape 3 : création du CSR (Certificate Signing Request)

openssl req -new -key secure.mysite.be.key -out secure.mysite.be.csr  
*Remplir le formulaire*cat secure.mysite.be.csr

### Etape 4 : création du certificat

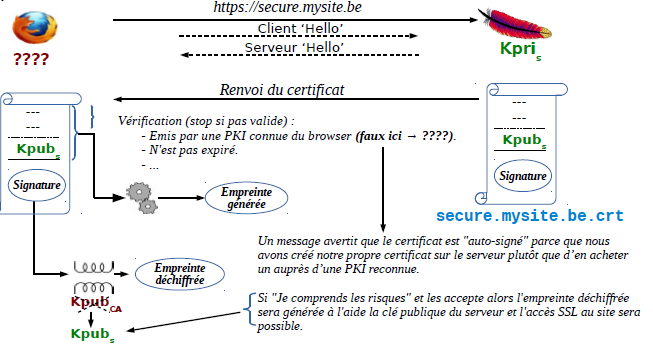
cd /tmp  
openssl x509 -in secure.mysite.be.csr -out secure.mysite.be.crt -req -signkey secure.mysite.be.key -days 3650

-signkey : Dans un certificat auto-signe, la signature est générée par la Kprivé du serveur et pas par la clé privée de la PKI

### Etape 5 : copie des clés et des certificats au bon endroit

cd /tmp  
cp secure.mysite.be.crt /etc/pki/tls/certs  
cp secure.mysite.be.csr /etc/pki/tls/private  
cp secure.mysite.be.key /etc/pki/tls/private

### Etape 6 : connexion SSL



### Configuration du virtual.conf

/etc/httpd/conf.d/virtual.conf

<VirtualHost 192.168.1.100:443>

ServerName secure.mysite.be  
DocumentRoot /var/www/html/secure  
**SSLEngine on (1)  
SSLCertificateFile /etc/pki/tls/certs/secure.mysite.be.crt (2)  
SSLCertificateKeyFile /etc/pki/tls/private/secure.mysite.be.key (3)**

</VirtualHost>

(1) Pour specifier que SSL doit etre utilise pour un hote virtuel et pas pour le serveur.  
(2) Nom complet du certificat pour ce site.  
(3) Nom complet de la cle privee pour ce site.

### Configuration du ssl.conf

Listen 443 → Apache écoutera également sur le port 443 de toutes ses interfaces.

SSLPassPhraseDialog builtin → Une passphrase éventuelle sera demandée sur l'entrée standard.

SSLSessionCacheTimeout 300 → Timeout pour les sessions SSL (300 secondes).

### Configuratoin du host

/etc/hosts

…  
192.168.27.100 secure.mysite.be

### Création d'une interface

/etc/sysconfig/network-scripts/ifcfg-enp0s3:0

IPADDR=192.168.27.1  
NETMASK=255.255.255.0  
ONBOOT=yes  
NAME=enp0s3:0  
DEVICE=enp0s3:0

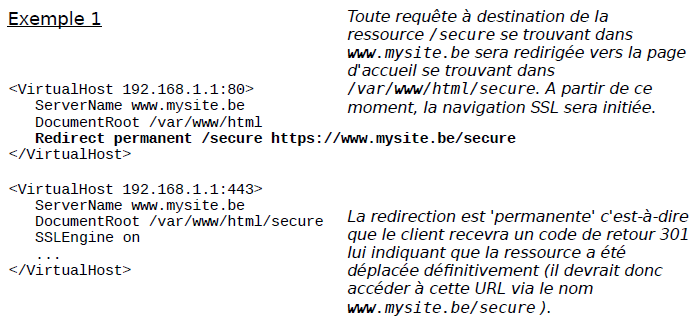
### Test avec lynx

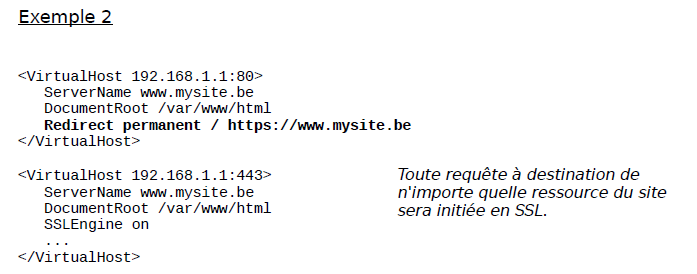
lynx [www.mysite.be](http://www.mysite.be) -> o (oui)  


## La redirection SSL

Pourquoi ?

* Obliger une navigation sécurisée.
* Un des facteurs de référencement de Google.





# Fournisseur d'autorisation

Fournisseur d'autorisation : all

* Ex : Require all granted
* Qui à le droit d'accéder à la ressource ? All = tout le monde, granted = autoriser, denied = refuser
* mod\_authz\_core

Fournisseur d'autorisation : ip

* Ex : Require ip 192.168.2/24
* mod\_authz\_host

Fournisseur d'autorisation : host

* Ex : Require host www.example.org .net example.edu
* mod\_authz\_host

Fournisseur d'autorisation : valid-user

* Require valid-user
* tous les utilisateurs repris dans le fournisseur d'authentification. : c'est l'ensemble des login et mdp qui sont codé dans le fichier userfile
* mod\_authz\_user

Fournisseur d'autorisation : user

* Require user toto albert
* les utilisateurs repris dans le fournisseur d'authentification (les utilisateurs voulu doivent être dans le fichier indiqué par la directive AuthUserFile
* mod\_authz\_user

Fournisseur d'autorisation : group

* Require group admins friends
* seuls les membres des groupes 'admins' et 'friends' ont le droit d'accéder aux ressources du répertoire moyennant une authentification valide.
* mod\_authz\_groupfile