Côté Labo : le service Web

Description du thème

Propriétés	Description
Intitulé long	Mise à disposition d'une application Web sécurisée
Formation concernée	BTS Services Informatiques aux Organisations
Matière	SI5
Présentation	L'objectif de ce coté labo (mis en œuvre en module) est de se familiariser avec la configuration d'un service Web. Il se déroule en 4 parties : • les principaux paramètres d'exécution du serveur ; • la gestion des accès aux pages distribuées ; • la mise à disposition de l'application Web ; • l'accès sécurisé à l'application Web. **Le contexte est celui du laboratoire pharmaceutique Galaxy-Swiss Bourdin (GSB)** *L'objectif est de mettre à disposition des utilisateurs l'application de gestion de frais avec un accès sécurisé.*
Savoirs	Les savoirs suivants peuvent faire l'objet d'un cours à partir d'applications existantes installées et configurées par le professeur : • principes d'architecture d'un service (architecture du service Web) ; • rôle et protocole associé au service WEB : HTTP ; • chiffrement et certificat ; • le protocole HTTPS et service Web sécurisé ; • typologies des risques et des dispositifs de sécurité liés à un service et à un serveur
Savoir-faire	Caractériser un service et le serveur associé Exploiter les fonctions de base d'un langage de commandes. Installer, configurer et administrer un service Gérer les habilitations d'accès aux ressources d'un serveur et d'un service Mettre en œuvre un protocole sécurisé associé à un service
Activités	D3.2 - Installation d'une solution d'infrastructure • A3.2.1 Installation et configuration d'éléments d'infrastructure D3.3 - Administration et supervision d'une infrastructure • A3.3.1 Administration sur site ou à distance des éléments d'un réseau, de serveurs, de services et d'équipements terminaux • A3.3.3 Gestion des identités et des habilitations D5.1 - Gestion des configurations • A5.1.2 Recueil d'informations sur une configuration et ses éléments
Transversalité	SI6
Outils	**SE** : serveur Linux debian Wheezy (stable). **Serveurs/sevices** : apache2, mysql-server **Clients** : navigateur web sur STA Linux, Windows ou autre système. Outils d'analyse et de tests de bon fonctionnement ainsi que phpmyadmin. **Contexte** : organisation/GSB-Organisation.doc. **Documentation** : organisation/outils/GSB-DocumentTechnique.doc.

Pré-requis	Commande de base d'un système Linux (fiche 1 de la documentation) avec accès à distance en SSH sur un serveur Linux Utilisateurs et gestion des droits d'accès Architecture d'un service sur Linux Le service de bases de données (fiche 4 de la documentation) Les notions sur le chiffrement (fiche 6 de la documentation) Rôle du service DNS
Mots-clés	Service, dns, url, web, http, https, chiffrement, certificat
Durée	10 heures
Auteur(es)	Apollonie Raffalli avec la relecture précieuse de Freddy Didier et Marie-Pascale Delamare
Version	v 2.0
Date de publication	15/05/12
Dernière modification	30/07/13

Contexte

Le laboratoire pharmaceutique Galaxy-Swiss Bourdin (GSB) désire mettre à disposition des visiteurs médicaux une application Web de gestion des frais de remboursement.
Il souhaite disposer d'une application en ligne, sécurisée, accessible depuis un navigateur par le nom pleinement qualifié **gestionfrais.gsb.coop**.

Cette application nécessite :

- un serveur Web sécurisé (HTTPS, SSL/TLS) développé en PHP ;
- l'accès à une base de données relationnelle, éventuellement administrable par interface Web ;

avec la possibilité pour les deux serveurs d'être sur la même machine physique.

L'authentification des visiteurs pour l'accès au contenu est gérée par l'application à travers la base de données.

L'entreprise a choisi d'héberger en interne les serveurs exécutant l'application sur un serveur Linux.

L'objectif de ce coté labo est de simuler dans l'environnement du laboratoire la mise à disposition de cette application.
Afin de mener à bien cette mission, les deux premières parties du TP sont consacrées à la compréhension et à la prise en main du serveur Web Apache2.

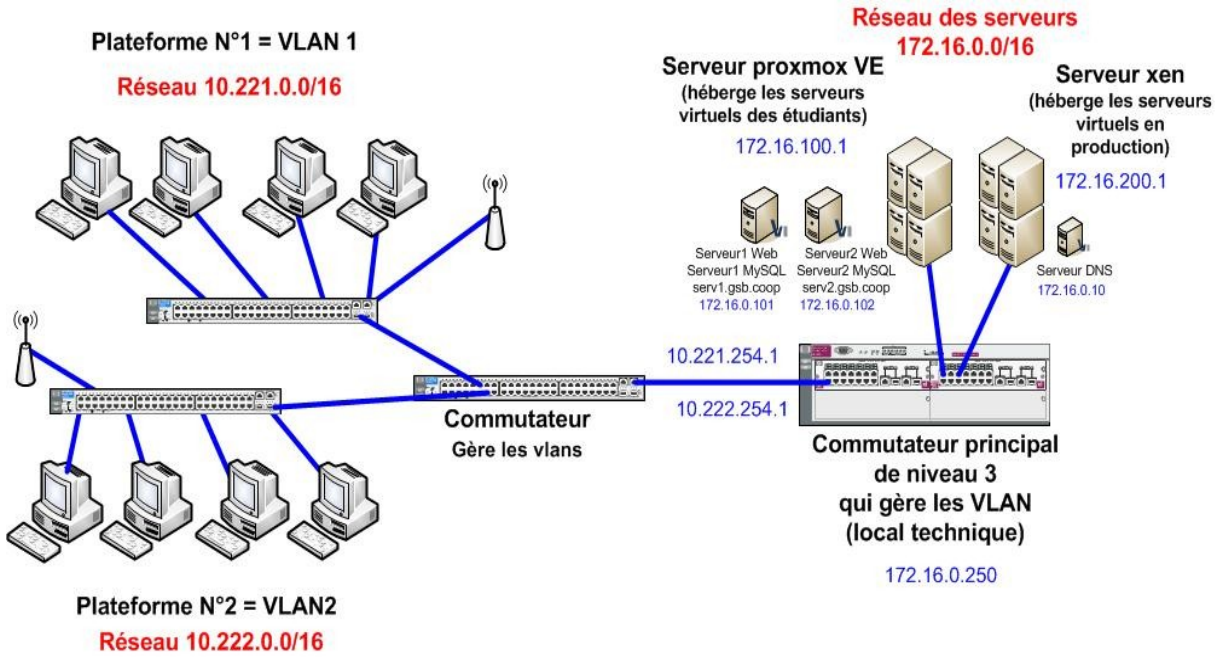
Vous disposez :

- d'une machine virtuelle sur laquelle sont installés le serveur SSH, le serveur de base de données MySQL et le serveur Web Apache2 avec la configuration par défaut ;
- d'applications Web déjà installées dont l'interface graphique « phpMyAdmin ».

La résolution des noms est prise en charge par un **serveur DNS déjà configuré** qui a (en interne) autorité sur la zone **gsb.coop**.

Votre serveur fait partie de la zone **gsb.coop**, il est accessible par son nom pleinement qualifié (par exemple **serv1.gsb.coop**).

Schéma réseau simplifié et réduit à deux plate-formes étudiantes et deux serveurs Web virtualisés



Préalable et rappels

Avant de commencer à modifier la configuration, faîtes une sauvegarde du répertoire (par exemple : `cp -r /etc/apache2 /etc/apache2.ori`).

Pour retrouver rapidement le fichier et le numéro de la ligne dans lequel se trouve une directive ou une expression particulière, vous pouvez utiliser le filtre « grep » avec les options « ***nir*** » par exemple :

```
grep -ni "listen" /etc/apache2/* renvoie :
/etc/apache2/ports.conf:9: Listen 80
/etc/apache2/ports.conf:17: Listen 443
/etc/apache2/ports.conf:21 Listen 443
```

L'option « ni » permet de chercher sans tenir compte de la casse et de renvoyer aussi le numéro de la ligne. L'astérisque peut être remplacé par un nom de fichier.

```
grep -ni "documentroot" /etc/apache2/* ne renvoie rien car la directive est
présente dans des fichiers de sous-répertoires.
```

L'option « r » permet une recherche récursive :

```
grep -nir "documentroot" /etc/apache2/* renvoie :
/etc/apache2/sites-available/default:4: DocumentRoot /var/www
/etc/apache2/sites-available/default-ssl:5: DocumentRoot /var/www
/etc/apache2/sites-enabled/000-default:4: DocumentRoot /var/www
```

Pour accéder directement à une ligne « n » d'un fichier : `vi nom_fichier +n`
Quand vous éditez un fichier avec l'éditeur « vi », **pour rechercher un mot** il suffit de saisir « /mot_recherche ».

Il est rappelé que la **configuration d'Apache** (comme celle de la plupart des services sur Linux) est **modulaire** c'est à dire que le fichier principal de configuration « inclut » des fichiers de configuration « externalisés » ayant chacun un rôle bien précis ; de plus, lors d'une mise à jour ce fichier pourrait être modifié. Il est donc fortement **déconseillé de le modifier** mais plutôt d'intervenir sur les fichiers inclus dont « httpd.conf » et/ou de créer dans les sous-répertoires des fichiers personnalisés.

Pour que Apache **prenne en compte les modifications** sur les fichiers de configuration, il ne faut pas oublier de les relire : `/etc/init.d/apache2 reload` (voir selon certaines modifications de redémarrer le démon : `/etc/init.d/apache2 restart`)

A chaque modification ou ajout de fichier, vérifier la syntaxe en passant la commande :
`apache2ctl configtest`

De même le statut d'Apache est détaillé par la commande : `apache2ctl status`

Prenez aussi l'habitude, notamment pour dépanner, de **consulter les fichiers d'activité** (log) ; s'il y a un problème, il est utile de consulter le fichier d'activité d'erreurs au fur et à mesure où il se construit :
`tail -f /var/log/apache2/error.log`

Des explications complémentaires sur les principales directives figurent en annexe 1 .

Il existe une excellente documentation en ligne (en français) : http://httpd.apache.org/docs/2.4/fr/ (documentation que l'on peut aussi installer en local).

Déroulement de la séquence

1. Les principaux paramètres d'exécution du serveur

- Vérifiez que les serveurs de base de données et Web sont installés et opérationnels.
- Selon le schéma d'articulation des applications, expliquez quel est le type de client serveur.

Processus et variables

À partir de la **liste des processus actifs d'Apache** (commande `ps -ef | grep apache`) :

- Repérez le numéro du processus père lancé par l'utilisateur « root ».
- Le nombre de processus fils lancés par l'utilisateur système « www-data ».

Dans le fichier de configuration principale d'Apache2, quelles sont les valeurs des directives :

- User ?
- PidFile ?
- StartsServers (du module « mpm_perfork_module ») ?

Dans le fichier « envvars » :

- Quelle est la valeur de la variable d'environnement « APACHE_RUN_USER » ?
- Quelle est la valeur de la variable d'environnement « APACHE_PID_FILE » ?
- Consultez le fichier correspondant à la variable d'environnement « APACHE_PID_FILE » ; à quoi correspond le nombre inscrit ?

La directive ServerName

La directive « ServerName » définit le nom d'hôte du serveur ; ce nom doit correspondre à une adresse IP et être renseigné dans un serveur DNS (ou fichier hosts). S'il n'est pas défini, alors le serveur tentera de le résoudre à partir de sa propre adresse IP. La configuration de la zone inverse du serveur DNS n'étant pas opérationnelle, cette résolution n'aboutit pas.

- Lancez la commande qui permet de vérifier la syntaxe `apache2ctl configtest`. Que renvoie-t-elle ?
- Il est possible que vous ayez un « warning » concernant le nom du serveur Web (n'empêchant pas le serveur de se lancer) ; le mieux est de créer la directive « ServerName » dans le fichier */etc/apache2/httpd.conf* et de la renseigner avec le nom de votre serveur pleinement qualifié.
- Rechargez le fichier de configuration et testez à nouveau la configuration.
- Y-a-t-il une autre possibilité qu'un serveur DNS pour obtenir une résolution de nom ?

Le port d'écoute

- Quelle est la directive principale qui définit le port d'écoute ?
- Modifiez ponctuellement ce port d'écoute (dans toutes les directives nécessaires) en le mettant à 888 et relancez le serveur Web.
- Quelle est l'URL que vous devez saisir pour accéder à votre serveur Web ?
- Revenez à un port d'écoute à 80.

Divers

- Citez un module disponible mais non chargé.

2. Page d'accueil et page par défaut

- Après avoir rappelé ci-après quel est le fichier représentant la page d'accueil du serveur et dans quel répertoire il se trouve, modifiez-le rapidement pour afficher aussi le nom et l'adresse IP de votre serveur. Vérifiez vos modifications avec l'URL : **http://nom_dns_du_serveur.**
- Renommez le fichier « index.html » par « accueil.html » et relancer l'URL. Que se passe-t-il ? Expliquez pourquoi en consultant la directive « DirectoryIndex »
- Comment peut-on procéder pour obtenir quand même la page demandée ?

Le fichier */etc/apache2/sites-available/default* contient la définition du site par défaut. En voici un extrait :

```
<VirtualHost *:80>
      ServerAdmin webmaster@localhost
      DocumentRoot /var/www
      ...
      <Directory /var/www/>
           Options Indexes FollowSymLinks MultiViews
           ...
      </Directory>
      ...
</VirtualHost>
```

- Quelle est l'option qui permet de rendre visible aux utilisateurs le contenu d'un répertoire ? Annihilez l'action de cette option sur le dossier /var/www/.
- Vérifiez vos modifications avec l'URL **http://nom_dns_du_serveur.** Quel est le message d'erreur ?

3. Les alias

Les alias permettent que des sites soient accessibles sans passer par le répertoire /var/www/ et/ou avec une URL qui cache le chemin complet des fichiers.

L'URL qui permet d'obtenir la page de connexion de « phpMyAdmin » est : **http://nom_dns_du_serveur/phpmyadmin/**.

- Les pages web de « phpMyAdmin » se trouvent-elles dans le répertoire */var/www/phpmyadmin/* ?

L'accès aux pages Web de « phpMyAdmin » se réalise grâce au fichier */etc/phpmyadmin/apache.conf* chargé par la configuration d'Apache. En effet, l'installation de « phpMyAdmin » a eu pour effet d'ajouter dans le dossier */etc/apache2/conf.d/* un lien appelé *phpmyadmin.conf* vers */etc/phpmyadmin/apache.conf*.

- Consultez ce dernier fichier et déterminez quelle est la directive permettant d'accéder à l'application « phpMyAdmin » via l'URL : **http://nom_dns_du_serveur/phpmyadmin/**

Installer la documentation d'Apache2 en local et vérifiez que vous pouvez y accéder à partir de l'URL http://nom_dns_du_serveur/manual.

- Quel est l'Alias permettant d'accéder à cette documentation ? Dans quel fichier se trouve-t-il ?

4. Configuration d'un hôte virtuel et mise à disposition de l'application Web

Il est très courant d'héberger et de gérer plusieurs sites, dits "virtuels", par un seul et même serveur (hébergement mutualisé).

Ceux-ci sont appelés par les clients sous différents noms DNS ; le serveur écoute une seule adresse IP à laquelle sont associés plusieurs noms de sites qui seront utilisés dans les URL clientes. Les sociétés se partageant un serveur Web peuvent ainsi avoir leurs propres domaines accessibles.

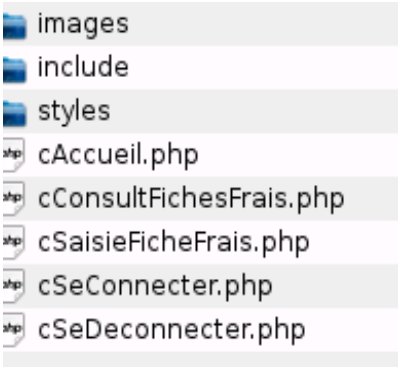
L'objectif est ici de mettre à disposition, sur chaque serveur, l'application Web de gestion de frais de GSB accessible via l'URL de type : gestionfraisServ{numServeur}.gsb.coop

Il est rappelé ici que :

- l'application accède à une base de données hébergée par le SGBD MySQL ;
- le serveur WEB et le serveur de base de données sont sur la même machine virtuelle ;
- l'authentification préalable pour l'accès au contenu est pris en charge par l'application via la table « Visiteurs »

L'accès sécurisé au service via le protocole HTTPS sera mis en œuvre dans la partie suivante.

Organisation du stockage des fichiers de l'application



Le code PHP a été organisé de façon à respecter certaines contraintes dont :

- la gestion des données est isolée dans un fichier à inclure (dossier *Include*) *bdGestionDonnees.lib.php* qui comporte les fonctions de manipulation des données et les fonctions de connexion ;
- la gestion des événements et de l'affichage sont prises en charge dans les mêmes fichiers (ce sont ceux qui commencent par un « c » .

Le fichier *cAccueil.php* est le fichier de démarrage de l'application.

Le principe de mise à disposition d'une application Web est le suivant :

- installer les fichiers web sur l'espace de stockage avec les droits nécessaires ;
- installer la base de données et en permettre l'accès à l'application ;
- configurer le(s) site(s) virtuel(s) dans un fichier de configuration spécifique situé dans */etc/apache2/sites-available/* ;
- activer ce site ;
- configurer le serveur DNS de manière à ce que n'importe quel hôte puisse accéder aux différents sites (cette étape sera réalisée par le professeur à partir des données que vous lui communiquerez).

Démarche

L'archive contenant les fichiers de l'application est *appliFrais.zip* et celle contenant les fichiers permettant de créer et d'initialiser la base de données est *bddAppliFrais.zip*

- Récupérez directement l'archive contenant les fichiers de l'application avec la commande « wget » à partir du serveur.
- Après décompression avec l'utilitaire « unzip », copiez les fichiers de l'application (dossier *appliFrais* compris) dans un dossier */var/www/appliGSB* (ce dernier doit être créé préalablement).

- Quelle est la commande permettant d'attribuer les droits corrects sur le répertoire « appliFrais » ?
- À partir de l'archive contenant les fichiers permettant de créer et d'initialiser la base de données, importez les fichiers SQL de création de la base et d'insertion des données.
- Selon la valeur des paramètres de la fonction de connexion à la base de données de l'application, créez l'utilisateur mysql requis et accordez-lui les droits nécessaires et suffisants.
- Vérifiez que vous pouvez accéder à l'application via l'URL « classique ».
- Vérifiez que vous pouvez vous authentifier pour accéder au contenu de l'application à partir d'un login et d'un mot de passe contenu dans la table « visiteurs ».
- Configurez le site virtuel de manière à y accéder à partir de l'URL : http://*gestionfraisServ{numServeur}.gsb.coop/cAccueil.php ;* cette configuration se fera dans le fichier *siteAppliFrais* que vous créerez dans */etc/apache2/sites-available/*
- Fournissez les paramètres nécessaires à votre professeur pour qu'il configure le serveur DNS.
- Activez le site et vérifiez l'accès à l'application.

5. La gestion des accès aux ressources

Dans un premier temps l'application ne pourra être accessible qu'à partir du serveur lui-même et du réseau dans lequel se trouvent vos machines clientes

- Modifiez le paramétrage de manière à tenir compte de cette contrainte.
- Testez votre nouveau paramétrage. Pour tester à partir du serveur Web, on utilisera le client WEB en mode texte « lynx ».

6. L'accès sécurisé à l'application Web

Les applications de GSB contiennent des données sensibles et sont accessibles de l'extérieur du réseau ; il est donc indispensable d'utiliser des protocoles d'échange sécurisés.

L'objectif est que les visiteurs puissent accéder à l'application « AppliFrais » avec le protocole HTTPS. Il est possible d'acheter, voir même d'obtenir gratuitement un certificat pour notre serveur Web mais la technique utilisée ici consiste à devenir notre propre autorité de certification qui délivrera ensuite le certificat signé pour notre serveur Web. Il est considéré ici que les notions de chiffrement et notamment de chiffrement asymétrique sont acquises. Vous trouverez en **annexe 2** la procédure pour mettre en œuvre une infrastructure de clé à l'échelle « réseau local » et pour configurer un serveur web avec le protocole SSL/TLS.

- Vérifiez que les paquetages « openssl » et « libssl » sont installés.

La suite du TP doit être réalisée par groupe de 2 étudiants, chacun disposant de son serveur, l'un jouant le rôle de l'autorité de certification et l'autre le rôle du représentant de la société GSB demandant un certificat (chaque étudiant doit bien évidemment participer au travail de l'autre).

6.1 - Le rôle de l'autorité de certification

Dans un système simplifié, l'Autorité de certification (AC ou CA) a pour rôle, après vérification de l'identité du demandeur du certificat, de signer, émettre et maintenir des certificats ainsi qu'une liste de révocation desdits certificats.
Elle doit, pour cela, posséder son propre certificat (auto-émis/auto-signé) qui permettra de valider le(s) certificat(s) émis.

Préparation du système

Il est nécessaire d'adapter le système et le fichier de configuration d'openssl aux besoins. La clé de l'autorité de certification, les demandes de certificats et certificats (émis et révoqués) seront créés dans le dossier /etc/ssl.

- Changez la valeur de la variable « dir » dans le fichier /etc/ssl/openssl.conf.
- Créez les dossiers strictement nécessaires dans le répertoire */etc* (qui accueillera les demandes et les nouveaux certificats créés).
- Créez un fichier vide *index.txt* et un fichier *serial* avec "01" écrit dedans.

Génération du certificat de l'autorité de certification

- Créez la clé privée de l'autorité de certification (attention de ne pas oublier votre pass phrase).
- Consultez (uniquement en lecture) avec un éditeur de texte le fichier produit (le format « PEM » permet d'obtenir un fichier constitué de caractères lisibles) ; relevez la première et la dernière ligne du fichier.
- Protégez la clé privée de l'autorité de certification en accordant uniquement des droits de lecture à l'utilisateur « root ».
- Créez le certificat de l'autorité de certification auto-signé (valide pour 5 ans et nommé *cacert.pem* et consultez-le.
- Rédigez une note à destination de GSB définissant la procédure de demande de certificat.

Avant de passer à l'étape suivante, l'étudiant concerné est dans l'attente d'une demande de certificat (voir 6.2).

Génération du certificat du serveur Web

Après avoir reçu, selon la procédure définie, la demande de certificat :

- Générez le certificat du serveur Web que vous nommerez *servgsbcert.pem*.

La commande de génération du certificat du serveur Web produit la sortie « écran » suivante :

```
Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for /etc/ssl/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 1 (0x1)
        Validity
            Not Before: Apr  2 21:32:43 2012 GMT
            Not After : Apr  2 21:32:43 2013 GMT
        Subject:
            countryName               = FR
            stateOrProvinceName       = Europe
            localityName              = Paris
            organizationName          = GSB
```

```
            organizationalUnitName    = Laboratoire France
            commonName                = gestionfraisServ1.gsb.coop
            emailAddress              = gsbparis@swiss-galaxy.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                7B:B2:96:A9:FC:05:74:60:C0:35:E3:48:D0:BF:83:0B:C3:6D:F4:5A
            X509v3 Authority Key Identifier:

keyid:AF:34:68:0A:D4:FA:8F:09:5E:1F:B2:2D:31:38:4C:44:1C:0D:C8:A6

Certificate is to be certified until Apr  2 21:32:43 2013 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Vous pouvez aussi consulter le contenu du certificat du serveur web généré.

En fonction de la sortie écran et/ou du contenu du certificat du serveur, répondez aux questions suivantes :

- Quelle est la clé privée utilisée pour signer le certificat ?
- Comment la commande `openssl` « sait » laquelle utiliser ?
- Quelle est la valeur du numéro de série du certificat ?
- Quelle est la durée de validité du certificat ?
- Relevez et expliquez le contenu des fichiers *index.txt* et *serial.*
- Quelle a été l'action précise de la dernière phrase affichée « Data Base Updated » ?

6.2 - Le rôle de la société GSB

Le rôle de la société est de générer la demande de certificat et de configurer le serveur WEB.

Génération de la demande de certificat du serveur Web

- Changez la valeur de la variable « dir » dans le fichier /etc/ssl/openssl.conf.
- Créez le(s) dossier(s) strictement nécessaire(s) dans le répertoire */etc* (qui accueillera les demandes et les certificats envoyés par l'autorité de certification).
- Générez la clé privée du serveur Web d'une longueur de 4096 bits dans un fichier *servgsbkey.pem*.
- Protégez cette clé privée en accordant uniquement des droits de lecture à l'utilisateur « root ».
- Générez la demande de certificat (dans un fichier *servgsbcsr.pem*) et relevez la valeur de votre « Common Name ».
- Consultez le contenu du fichier généré et relevez la première et la dernière ligne.

Configuration d'APACHE2 avec mod_ssl

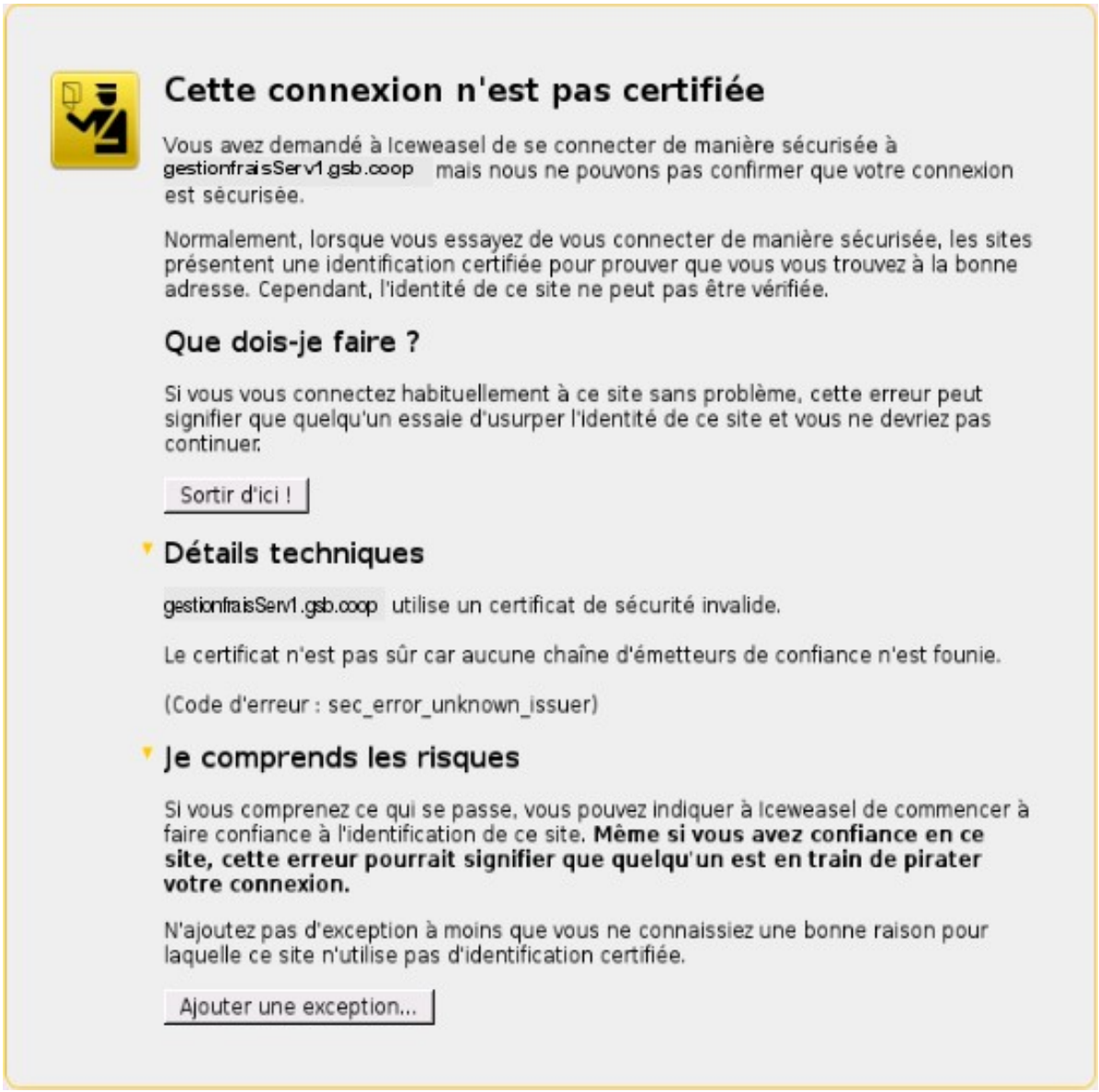
- Activez le module « ssl » et redémarrez le serveur WEB.
- Vérifiez l'écoute sur le port 443.

L'objectif est que l'application soit accessible non plus en HTTP mais en HTTPS.

- Copiez le fichier de configuration */etc/apache2/site-available/siteAppliFrais* en le renommant */etc/apache2/sites-available/siteAppliFraisSSL*.
- Désactivez le site accessible en HTTP et vérifiez qu'il ne soit plus accessible.
- Configurez (dans le fichier */etc/apache2/sites-available/siteAppliFraisSSL*) l'hôte virtuel pour qu'il soit accessible en HTTPS.
- Activez le nouveau site, redémarrez le serveur et s'il n'y a pas d'erreurs testez une connexion en HTTPS ; l'alerte de sécurité (warning) au niveau du certificat est « normal » (voir ci-après).

Test du serveur Web sécurisé

Lorsque vous tentez d'accéder à votre site Web sécurisé (par exemple, avec l'URL **https://gestionfraisServ1.gsb.coop**), le navigateur renvoie une alerte de sécurité qui se présente ainsi sur le navigateur « firefox » :



- Rédigez une note en direction des visiteurs médicaux précisant les raisons de cette alerte de sécurité et la procédure pour y remédier (la solution « Ajouter une exception » n'est pas recevable !).

Annexes

Annexe 1 : architecture du serveur Web

1.1 Architecture Client-Serveur

Le rôle du service Web est de diffuser de l'information sur Internet ou sur un Intranet ainsi que de mettre à disposition des utilisateurs des applications Web.

Ce service rendu aux utilisateurs s'appuie sur des services réseaux comme le service DNS de résolution de nom et nécessite des outils complémentaires comme un serveur de bases de données. Il concerne donc tous les acteurs de l'organisation (aussi bien les administrateurs réseau et système que les développeurs).

La notion de serveur représente la machine (physique ou virtuelle) et aussi l'application installée et configurée qui permet de rendre le service.
Selon le journal du net (http://www.journaldunet.com/developpeur/outils/part-de-marche-des-serveurs-web/) les parts du marché des serveurs WEB en janvier 2012 s'élèvent à 65% pour Apache et 15 % pour IIS sur Windows Server

Le protocole mis en œuvre dans le cadre d'un serveur Web est le protocole HTTP qui implique deux éléments :

- **la requête HTTP** qui est le message envoyé par le client vers le serveur qui contient :
  - l'URL de la ressource demandée ;
  - les informations relatives à la plate-forme du client ;
  - les informations relatives au client.
- **la réponse HTTP** qui est la réponse à la demande d'un client :
  - les entêtes : nom du serveur, versions du protocole supportées, autres fonctions accessibles... ;
  - des pages HTML et les éléments qui leurs sont liés (CSS, scripts côté client, images appelées par <img>, modules incorporés par <embed> comme du flash ou du streaming, etc) ;
  - des pages HTML résultant de l'exécution de pages de Script côté serveur (PHP, ASP, JSP, ...).

Un serveur Web contient donc des pages Web (liées entre elles par des liens hypertexte) et beaucoup d'objets incorporés. Chacun de ces objets (images, sons, …) est un objet indépendant récupéré séparément.
Le client web récupère le texte (HTML) suivi des objets liés.

Chaque page du site est identifiée par son URL (Uniform Resource Locator).

Il existe plusieurs possibilités d'hébergement Web :

- hébergement local : l'organisation dispose et gère son (ses) propre(s) serveur(s) (réel(s) ou virtuel(s)) ;
- hébergement Internet (par exemple, OVH.com (http://www.ovh.com/fr/index.xml) rend ce type de service) :
  - l'organisation loue une machine (voire des baies) hébergée dans un datacenter (en général, l'organisation dispose d'un contrôle complet sur la machine) ;
  - l'hébergement peut être mutualisé : l'organisation partage une machine avec d'autres utilisateurs (dans ce cas, pas de contrôle sur la machine mais un accès FTP pour gérer le(s) site(s) Web).

Dans tous les cas, on rencontrera l'une ou l'autre des configurations suivantes :

- plusieurs petits sites hébergés par machine ;
- plusieurs machines pour un gros site.

1.2 Installation et démarrage du serveur

1.2.1 Installation

Si Apache2 n'est pas installé, la commande `apt-get install apache2 libapache2-mod-php5` installera le serveur Web avec ses dépendances ainsi que le module php.

Les paquets installés peuvent être listés avec la commande `dpkg -l | grep apache2`

```
ii apache2-mpm-prefork (moteur du serveur : façon dont le serveur intercepte les requêtes)
ii apache2-utils (contient les divers utilitaires facilitant la gestion du serveur)
ii apache2.2-bin (contient les exécutables du serveur)
ii apache2-common (contient le serveur et les modules standards Apache2)
ii libapache2-mod-php5 (module php5 pour Apache 2)
```

Pour installer la documentation en local : `apt-get install apache2-doc`
Elle est installée dans */usr/share/doc/apache2-doc*

Rappel : il existe aussi une excellente documentation en ligne (en français) :
http://httpd.apache.org/docs/2.4/fr/

1.2.2 Démon et processus

Pour que le serveur Web puisse répondre à une demande d'un client, il doit être démarré. Sous Debian : `/etc/init.d/apache2 start.`
Au moment de son démarrage, Apache2 charge ses fichiers de configuration et se met en attente de requêtes sur les interfaces réseaux. Le service écoute, par défaut, sur le port 80.

Le paquetage « apache2-mpm-prefork » installé par défaut permet au service WEB de fonctionner en mode « Prefork » (c'est le seul mode supporté par le module « mod-php5 ») ; ce qui signifie qu'un processus père lancé avec les droits (root) pré-exécute des processus enfants (lancés avec les droits limités de l'utilisateur système « www-data ») qui traiteront chacun une requête cliente.
Ces différents paramétrages et autres modules installables pour améliorer les performances seront étudiés l'année prochaine.

1.2.3 Accès aux pages et objets envoyés par le serveur

La racine par défaut du service WEB sur le serveur Apache2 est le répertoire */var/www/* et la page d'accueil par défaut est *index.html.*

Pour **pouvoir consulter le contenu d'un site Web**, les utilisateurs doivent avoir accès en lecture aux fichiers (un accès en écriture à certains répertoires n'est que très rarement permis).

Le processus qui répond aux requêtes a pour propriétaire l'utilisateur système « www-data » appartenant au groupe « www-data » ; les droits des utilisateurs sont donc en fait les droits de l'utilisateur « www-data ».

Ainsi, il existe 2 possibilités pour qu'un utilisateur accède en lecture aux fichiers demandés :

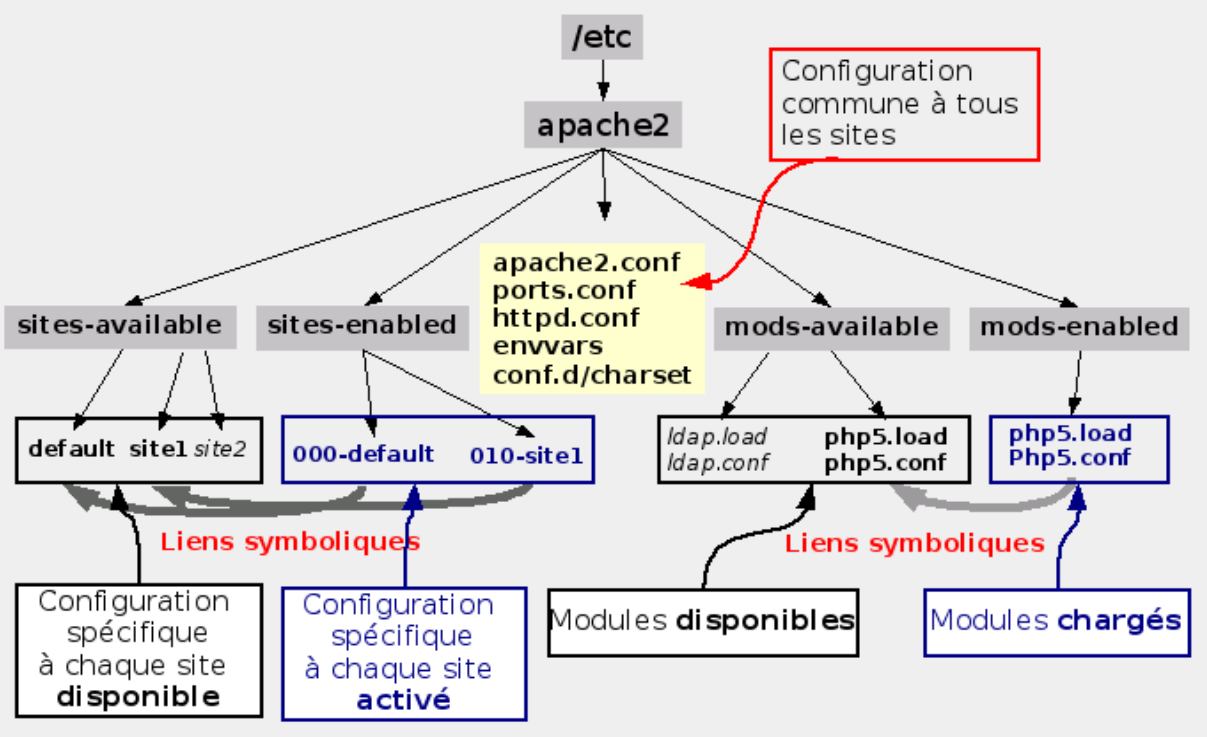
- donner le droit de lecture à tout le monde (donc aux « autres ») à ces fichiers ;
- attribuer la propriété des fichiers à l'utilisateur www-data et/ou au groupe www-data et attribuer les droits de lecture nécessaires

Des directives dans les fichiers de configuration d'Apache2 gèrent également les droits des fichiers.

1.3 Fichiers de configuration : une approche modulaire

1.3.1 - Vue d'ensemble des fichiers de configuration

Apache2 dispose de plusieurs fichiers de configuration dans */etc/apache2*.



apache2.conf : est le principal fichier de configuration d'Apache. Il renferme les paramètres généraux et communs à tous les serveurs. Ce fichier de configuration principal est modulaire : il inclut des directives provenant d'autres fichiers grâce aux clauses « Include ». Cette approche permet beaucoup de souplesse et renforce la sécurité.
ports.conf : contient la (les) directive(s) « listen » qui spécifie(nt) le(s) port(s) d'écoute (par défaut HTTP:80, HTTPS:443 si le module est chargé).Il permet la création d'hôtes virtuels (un seul serveur apache, plusieurs sites sous des noms différents)
httpd.conf : est l'ancien fichier de configuration d'Apache, il est actuellement vide. C'est le fichier idéal pour y insérer les paramétrages spécifiques communs à tous les serveurs (ce qui permet de ne pas modifier le fichier « apache2.conf »).
envvars : définit les variables d'environnement propres à Apache ;
conf.d : est un répertoire contenant des fichiers qui seront analysés par apache dont le fichier « charset » spécifiant l'encodage à utiliser par défaut. De nombreuses applications Web créent des fichiers de configuration spécifiques dans ce répertoire (ou des liens pointant vers des fichiers de configuration) ;
sites-available : contient les fichiers de configuration des sites web disponibles ;
sites-enabled : contient les fichiers de configuration des sites web ACTIVÉS : ce sont en fait des liens symboliques qui pointent vers les fichiers présents dans le dossier *sites-available*
mods-available : contient les modules et les fichiers de « conf » des modules dynamiques installés (mais non chargés) par apache2
mods-enabled : contient les modules et les fichiers de « conf » des modules dynamiques chargés par apache2. Ce sont en fait des liens symboliques qui pointent vers les fichiers présents dans le dossier *mods-available*.

On active un site (ce qui revient donc à créer le lien) avec la commande : `a2ensite site1`, *site1* étant un fichier de configuration présent dans */etc/apache2/sites-available/.*
Pour désactiver un site : `a2dissite site1`

On active un module (ce qui revient donc à créer le lien) avec la commande : `a2enmod nomModule`.
Pour désactiver un module : `a2dismod nomModule`

1.3.2 Compléments sur quelques directives

ServerRoot /etc/apache2	La **directive ServerRoot** définit le répertoire dans lequel se situent les fichiers de configuration
DirectoryIndex DirectoryIndex index.html index.cgi index.pl index.php index.xhtml index.htm	La **directive DirectoryIndex** définit la liste des ressources à chercher lorsque le client requiert un index du répertoire par ajout du slash final à une URL pointant sur ce répertoire. Vous pouvez bien évidemment rajouter à la directive les fichiers que vous voulez.
ServerAdmin webmaster@localhost	La **directive ServerAdmin** définit l'adresse e-mail que le serveur inclut dans tout message d'erreur retourné au client.
ServerName nom_dns_du_serveur	La **directive ServerName** est le nom du serveur tel qu'il devra être tapé dans la barre d'adresse du navigateur
DocumentRoot /var/www	La **directive DocumentRoot** définit le répertoire racine à partir duquel Apache2 va distribuer les fichiers (chemin du répertoire source qui contient les documents accessibles aux clients) sauf si le répertoire est pointé par une directive telle que Alias. Le serveur ajoute le chemin mentionné dans l'URL pour établir le chemin complet jusqu'au document. **Exemple** : DocumentRoot /var/www Un accès à http://nom_dns_du_serveur/tp/index.html se réfère au document /var/www/tp/index.html.

Chaque répertoire potentiellement accessible par Apache peut être configuré via les directives placées entre les balises *<Directory nom_du_repertoire>* et *</Directory> ;* ces clauses s'appliquent aussi à tous les sous-répertoires sauf s'il est aussi prévu pour le sous répertoire des directives spécifiques inclues dans *<Directory nom_du_sous_repertoire>* et *</Directory>*.

	L'option *Indexes* contrôle la visibilité du contenu d'un répertoire : lorsqu'une URL requise pointe sur un répertoire et qu'aucun fichier défini par DirectoryIndex (ex. index.html) n'existe dans ce répertoire, alors le serveur retourne une liste du contenu du répertoire. Si l'indexation n'est pas activée, on obtient une page d'erreur. Pour désactiver une option, il suffit de mettre un « moins » (tiret) devant, par exemple : ***Options -Indexes***...
<Directory /sitesweb/btssio1/tp> **Options** Indexes **order** deny, allow **deny** from all **allow** from 192.168.1.0/24 **allow** from .sio.fr </Directory>	La **directive allow** configure les hôtes qui peuvent accéder au répertoire et sous-répertoires. La **directive deny** empêche certains hôtes d'accéder au répertoire et sous-répertoires. *On peut spécifier explicitement des adresses (unicast, réseau), des noms de machine ou de réseau ou all, allow from et deny from* La **directive order** contrôle l'ordre dans lesquelles les directives allow et deny sont évaluées. **deny,allow :** les directives deny sont évaluées avant les directives allow. **allow,deny :** les directives allow sont évaluées avant les directives deny. **Dans l'exemple :** on interdit tout accès et on gère ensuite les autorisations ; on désire supprimer l'accès du dossier */sitesweb/btssio1/eleve* à tout le monde sauf pour les machines du réseau d'adresse 192.168.1.0 et de nom de domaine sio.fr.

1.3.3 Configuration d'un hôte virtuel (ou « virtual host »)

Les directives concernant **un hôte virtuel** doivent être comprises dans les balises <*VirtualHost adresseIP_du_serveur:80*> et <*/VirtualHost*>. Pour chaque hôte virtuel il est préférable de créer un fichier dans */etc/apache2/sites-avalaible/* puis d'activer le nouveau site

Les directives minimales pour configurer un hôte virtuel sont les suivantes :

<VirtualHost adresseIP_du_serveur:80> **ServerName** nom_dns **DocumentRoot** chemin_racine_web_du_site … </VirtualHost>	**ServerName** nom_dns indique le nom pleinement qualifié à partir duquel le client peut accéder à la ressource désignée par la directive DocumentRoot. **On doit configurer le service DNS** de façon à ce que celui-ci fasse la résolution des noms : nom_dns <==> @IP du serveur **DocumentRoot** désigne donc le chemin absolu où se trouvent les documents du site.

Attention : le nom de serveur (*ServerName*) et la racine web (*DocumentRoot*) doivent exactement correspondre respectivement au nom sous lequel le serveur virtuel sera nommé dans les URL clientes et au chemin du répertoire d'accueil des documents du site avec les noms de dossiers sensibles à la casse.

Exemple :
ServerName serv1.sio.fr
DocumentRoot /var/www/TP
Un accès à http://serv1.sio.fr/tp1/index.php se réfère au document /var/www/TP/tp1/index.php.

On peut ensuite, si nécessaire, configurer l'accès au dossier (toujours entre les balises <*VirtualHost adresseIP_du_serveur:80*> *et* <*/VirtualHost*>) ; toutes les balises d'un serveur Web sont admises (DocumentRoot, Listen, Directory, etc.), par exemple :

```
<Directory chemin_racine_web_du_site>
        Options Indexes
         AllowOverride None
        Order allow,deny
         allow from all
</Directory>
```

1.4 Les fichiers d'activité

Le serveur Apache2 enregistre son activité dans des journaux (logs), situés dans /var/log/apache2/ :

- **access.log** enregistre toutes les requêtes (qu'elles soient réussies ou non) ;
- **error.log** enregistre tout type d'erreurs dont les requêtes en erreur et les dysfonctionnements du serveur.

Exemple : /var/log/apache2/access.log (extrait)

```
66.249.72.103 - - [12/Mar/2012:01:22:20 +0100] "GET /spip/spip.php?
article14 HTTP/1.1" 200 10644 "-" "Mozilla/5.0 (compatible; Googlebot/2.1;
+http://www.google.com/bot.html)"
93.158.150.21 - - [12/Mar/2012:01:22:37 +0100] "GET
/~rafapo07/ftp/docserveurs.pdf HTTP/1.1" 200 3016165 "-" "Mozilla/5.0
(compatible; YandexBot/3.0; +http://yandex.com/bots)"
79.84.198.189 - - [12/Mar/2012:11:37:31 +0100] "GET
/spip/local/cache-texte/8cc5a43bd49043eee31d5d91e54b8cfc.png HTTP/1.1" 304
- "http://llb.ac-corse.fr/spip/" "Mozilla/5.0 (Windows NT 5.1; rv:10.0.2)
Gecko/20100101 Firefox/10.0.2"
79.84.198.189 - - [12/Mar/2012:11:37:32 +0100] "GET
/spip/local/couteau-suisse/header.js HTTP/1.1" 304 -
"http://llb.ac-corse.fr/spip/" "Mozilla/5.0 (Windows NT 5.1; rv:10.0.2)
Gecko/20100101 Firefox/10.0.2"
79.84.198.189 - - [12/Mar/2012:11:37:29 +0100] "GET /spip/spip.php?
page=porte_plume_start.js&lang=fr HTTP/1.1" 200 56111
"http://llb.ac-corse.fr/spip/" "Mozilla/5.0 (Windows NT 5.1; rv:10.0.2)
Gecko/20100101 Firefox/10.0.2"
10.36.96.143 - - [12/Mar/2012:11:38:15 +0100] "GET
/Squidguard/pageinterdite.html HTTP/1.1" 200 1172 "-" "Mozilla/4.0
(compatible; MSADAPI 2.5.4322.0; Windows NT 6.1; )"
90.14.112.119 - - [12/Mar/2012:11:47:52 +0100] "GET /article3.php
HTTP/1.1" 404 180 "-" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/535.11
(KHTML, like Gecko) Chrome/17.0.963.78 Safari/535.11"
```

Un enregistrement donné commence par l'adresse IP du client qui a accédé au serveur Web.

Les « nombres » en gras indiquent si la connexion s'est bien effectuée. Quelques exemples de code :
200 : requête acceptée et objet envoyé ;
304 : l'utilisateur revient sur une page qu'il a déjà dans le cache de son navigateur et cette version « en cache » est toujours d'actualité ;
404 : document non trouvé ;
401 : l'identification a échoué : il n'est pas permis à l'utilisateur d'obtenir ce document.

Lecture du fichier d'activité :

- les deux premières lignes correspondent à un robot d'indexation qui scanne les serveurs web pour en indexer les pages ;
- les trois lignes suivantes correspondent à une activité normale où, pour les 2 premières requêtes, le client avait « en cache » les pages demandées ;
- la ligne suivante correspond à une activité normale et fait suite à un refus du proxy d'accéder à leur demande : la requête a été détournée vers une page spécifique « pageinterdite.html » ;
- la dernière ligne correspond à une requête refusée car la page Web demandée n'existe pas (voir ci-après).

On doit forcément retrouver les lignes correspondantes aux erreurs (avec des explications complémentaires) dans le fichier « error.log ». Par exemple :

Pour la dernière ligne du fichier « access.log » :
90.14.112.119 - - [12/Mar/2012:11:47:52 +0100] "GET /article3.php HTTP/1.1" 404 180 "-" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/535.11 (KHTML, like Gecko) Chrome/17.0.963.78 Safari/535.11"

Voici la ligne correspondante dans le fichier « error.log » :
[Mon Mar 12 11:47:52 2012] [error] [client 90.14.112.119] script '/var/www/article3.php' not found or unable to stat

Annexe 2 : accès sécurisé à un serveur Web via le protocole HTTPS

2.1 Préparation du système

OpenSSL fournit une application pour créer des certificats client ou serveur. Il s'appuie sur un fichier de configuration */etc/ssl/openssl.conf*
Toutes les commandes d'openssl vont chercher divers paramètres dans ce fichier (à adapter aux besoins) sauf si la ligne de commande donne d'autres directives.

Voici l'extrait qui nous intéresse :

```
[ CA_default ]
dir               = ./demoCA              # Where everything is kept
certs             = $dir/certs            # Where the issued certs are kept
crl_dir           = $dir/crl              # Where the issued crl are kept
database          = $dir/index.txt        # database index file
...
new_certs_dir     = $dir/newcerts         # default place for new certs.
certificate       = $dir/cacert.pem       # The CA certificate
serial            = $dir/serial           # The current serial number
...
crl               = $dir/crl.pem          # The current CRL
private_key       = $dir/private/cakey.pem      # The private key
```

L'arborescence à créer (ou à compléter) est donc la suivante :

- *certs* accueille les certificats nouvellement créés à envoyer aux demandeurs
- *crl* accueille les certificats révoqués
- *newcerts* accueille une copie des certificats que l'organisation garde
- *private* accueillera les clés privées

Les fichiers sont les suivants :

- *private/cakey.pem* est la clé privée de l'autorité de certification
- *index.txt* va contenir la liste des certificats que l'on générera
- *serial* contient le numéro de série du prochain certificat créé : il est donc nécessaire de l'initialiser à « 1 »

En ce qui concerne les clés privées, il est important de comprendre qu'une véritable infrastructure de clés (c'est à dire un organisme de tiers de confiance qui délivre des certificats) ne détient pas les clés privées correspondantes aux clés publiques contenues dans les certificats ; on est ici dans un cas spécifique où l'on a deux casquettes : le prestataire (le tiers de confiance qui délivre un certificat) et le client (qui demande un certificat).

2.2 Génération du certificat de l'autorité de certification

La génération du certificat de l'autorité de certification se réalise en deux étapes.

Première étape : création de la clé privée de l'autorité de certification

La commande `openssl genrsa -des3 -out chemin/cakey.pem 4096` crée un fichier *cakey.pem* contenant la clé privé de 4096 bits protégé par une « pass phrase » (option -des3)
Cette « pass phrase » (mot de passe long qui peut contenir des espaces) sera demandée à chaque utilisation de la clé.

Deuxième étape : génération du certificat auto-signé de l'autorité de certification

Il est possible de créer en une seule commande, à partir de cette clé, un certificat x509 auto-signé :
`openssl req -new -x509 -days durée_en nbre_jours -key chemin/cakey.pem -out chemin/nom_certificat.pem`
req -new : demande d'un nouveau certificat
-x509 : le certificat demandé est auto-signé
-days durée_en nbre_jours : la durée de validité du certificat sera du nombre de jours spécifié
-key chemin/cakey.pem : le chemin vers la clé privée à partir de laquelle sera généré le certificat contenant la clé publique correspondante.
-out chemin/nom_certificat.pem : création d'un fichier de nom *nom_certificat.pem* (qui sera le certificat auto-signé de l'autorité de certification) dans /etc/ssl (voir fichier /*etc/ssl/openssl.conf*)

Le mot de passe demandé est celui de la « pass phrase » saisie précédemment puisque vous utilisez la clé privée que vous avez protégée.

Il faut ensuite donner des renseignements sur l'autorité de certification :
*Country Name (2 letter code) [AU]:**FR***
State or Province Name (full name) [Some-State]:**Europe**
Locality Name (eg, city) : **Paris**
Organization Name (eg, company) [Internet Widgits Pty Ltd]: **GSB**
Organizational Unit Name (eg, section) []: **Laboratoire France**
Common Name (eg, YOUR name) []:**CA GSB France**
Email Address []: **cagsbparis@swiss-galaxy.com**

Remarque : la valeur de ces champs peut être saisie comme valeur par défaut dans *openssl.conf*

Ce certificat va permettre de vérifier l'authenticité et la validité du certificat du serveur Web.

2.3 Génération du certificat du serveur WEB

Cette génération s'effectue en trois étapes.

Première étape : création de la clé privée du serveur

On génère la clé privée de la même manière que précédemment mais sans la protéger par une « pass phrase » (sans l'option -des3) car sinon cette dernière sera demandée à chaque lancement d'Apache2 !

A partir de la clé privée générée, il s'agit de créer un fichier de **demande de certificat.**

Deuxième étape : création de la demande de certificat

Pour générer le CSR (Certificate Signing Request) c'est à dire **le formulaire de demande de certificat à partir de la clé privée préalablement créée,** la commande est la suivante : `openssl req -new -key chemin/nom_clé_privée.pem -out chemin/nom_fichier_dem.pem`

Le système va demander de **saisir des champs** ;
Country Name (2 letter code) [AU]: **FR**
State or Province Name (full name) [Some-State]: **Europe**
Locality Name (eg, city) []: **Paris**
Organization Name (eg, company) [Internet Widgits Pty Ltd]: **GSB**
Organizational Unit Name (eg, section) []: **Laboratoire France**
Common Name (eg, YOUR name) []: ***partie de l'URL correspondante au nom DNS***
Email Address []: **gsbparis@swiss-galaxy.com**

Ce n'est pas la peine de saisir d'autres "extra attributes"...

Soyez très vigilant sur le Common Name (CN) : celui-ci doit correspondre exactement au nom pleinement qualifié (nom DNS : celui que vous allez saisir dans l'URL de votre serveur qui correspond aussi à la directive ServerName dans le fichier de configuration d'Apache).

Remarque : il est possible d'utiliser un autre champ que CN, le champ « subjectAltName » pour spécifier le nom DNS du serveur. Ce dernier champ permet en outre de préciser un ou plusieurs noms DNS (ceux correspondant aux différents virtuals hosts). Ce qui implique qu'il suffit d'un seul certificat pour l'ensemble des hôtes virtuels d'un serveur web.
Mais il faut pour cela paramétrer plus finement le fichier openssl.conf.

La troisième et dernière étape consiste à signer ce certificat. Cette étape (comme celle concernant la génération d'un certificat d'autorité de certification) n'aurait pas lieu d'être si nous faisions appel à une autorité de certification externe à laquelle on aurait tout simplement envoyé (selon une procédure définie par l'autorité de certification) la demande de certificat.

Troisième étape : production du certificat signé par l'autorité de certification

Cette signature et la production du certificat se réalisent avec la commande suivante :
openssl ca -policy policy_anything -out chemin/nom_du_certificat.pem -infiles chemin/nom_fichier_dem.pem

ca : c'est l'autorité de certification qui agit.
-policy policy_anything : c'est la clause policy_anything du fichier de configuration *openssl.cnf* qui sera utilisée. Dans notre cas seul le « Common Name » sera testé.
-out chemin/nom_du_certificat.pem : le certificat généré et à envoyer
-infiles chemin/nom_fichier_dem.pem : le fichier de demande de certificat utilisé pour générer le certificat.

Le certificat du serveur (que l'on doit normalement envoyer à celui qui nous l'a demandé) a bien été créé mais il y a aussi une copie du certificat « **01.pem** » qui a été automatiquement créé dans **newcerts**. L'autorité de certification le garde car elle a besoin de cet original en cas de révocation...

Il est maintenant nécessaire de configurer le serveur Web Apache2.

2.4 Configuration d'APACHE2 avec mod_ssl

Le module ssl d'Apache2 implémente le protocole HTTPS. L'activation de ce module permet en outre à Apache2 d'écouter sur le port 443 qui est le port par défaut pour HTTPS.

Il s'agit maintenant de configurer l'hôte virtuel.

Nous allons faire fonctionner sur notre serveur **simultanément** des « virtualhost » fonctionnant sous HTTP (sur le port 80) et des « virtualhost » fonctionnant sous HTTPS (sur le port 443).
Il faut donc préciser systématiquement les ports dans la configuration des hôtes virtuels :

```
NameVirtualHost * : 80
+ tous les <VirtualHost @IP : 80>
```

Pour chaque hôte virtuel devant utiliser le protocole HTTPS, il est nécessaire d'ajouter un certain nombre de directives.
Vous avez un exemple de configuration d'un hôte virtuel dans le fichier */etc/apache2/sites-available/default-ssl*. Nous ne verrons ci-dessous que les directives obligatoires.

Après avoir changé le port 80 en 443, les directives à ajouter entre les balises <*VirtualHost @IP:port*> et </*VirtualHost*> sont les suivantes :

```
SSLEngine on
SSLCertificateFile chemin_vers_le_certificat_du_serveur_web
SSLCertificateKeyFile chemin_vers_la_clé_privé_du_serveur_web
```

Remarque :
Les erreurs de configuration sont fréquentes, il faut s'aider du fichier d'activité */var/log/apache2.log*. Il est même possible d'isoler les erreurs relatives à HTTPS (c'est beaucoup mieux pour « debugguer ») avec la directive :
ErrorLog /var/log/apache2/error_ssl.log (attention : il faut créer le fichier *error_ssl.log)*.

Et si ce n'est toujours pas suffisant, vous pouvez même mettre la directive « LogLevel » à « debug » (LogLevel debug) ou « info » (LogLevel info) à la place de la directive « LogLevel warn » puis dans une console : `tail -f /var/log/apache2/error_ssl.log` pour voir quels sont les problèmes qui surviennent quand vous rechargez la lecture de la configuration :

Une autre commande peut vous aiguiller sur une éventuelle cause d'erreur :
```
openssl s_client -connect nom_dns:443
```