Guide pratique de mise en œuvre d’un serveur WebDAV sous Apache avec LDAP et SSL

Version française du Apache based WebDAV Server with LDAP and SSL

Saqib Ali

Développement XML/XHTML distant

`<saqib CHEZ seagate POINT com>`

Adaptation française: Denis Berhaut

Relecture de la version française: Vincent Loupien

Préparation de la publication de la v.f.: Jean-Philippe Guérard

Version : 4.1.2.fr.1.1

30 janvier 2006

Historique des versions		
Version 4.1.2.fr.1.1	2006-01-30	JPG
Correction de mise en forme sur une suggestion de Bernard Siaud.		
Version 4.1.2.fr.1.0	2004-12-20	DB, VL, JPG
Première traduction française		
Version 4.1.2	2003-10-17	SA
Ajout de la section d'optimisation SSL		
Version 4.1.1	2003-09-29	SA
Mise à jour de la section SSL suite à des commentaires de lecteurs		
Version 4.1.0	2003-09-02	SA
Mise à jour de la section SSL suite à des commentaires de lecteurs		
Version 4.0.2	2003-08-01	SA
Mises à jour mineures de la ligne de commande de configuration d'Apache /dev/random référencée dans la section SSL.		
Version 4.0.1	2003-07-27	SA
Ajout d'informations dans la section SSL		
Version 4.0	2003-06-29	SA
Mise à jour du guide pratique pour Apache 2.0. De plus, conversion du source en XML.		

Résumé

Ce document constitue le guide pratique de mise en œuvre d'un serveur WebDAV Apache utilisant LDAP pour l'authentification et SSL pour le chiffrement.

Table des matières

1. Introduction

L’objectif de ce document est de configurer un serveur d’applications Apache avec mySQL, PHP et WebDAV, qui utilise LDAP pour l’authentification. La documentation fournira aussi des détails sur le chiffrement des transactions LDAP.

N.B. :

Si vous rencontrez des problèmes en installant Apache ou un quelconque de ses modules n’hésitez pas à contacter l’auteur en anglais à <`saqib CHEZ seagate POINT com`>

N’hésitez pas à faire parvenir tout commentaire relatif à la version française de ce document à <`commentaires CHEZ traduc POINT org`> en précisant le titre et la version du document.

1.1. À propos de ce document

J'ai commencé à écrire ce document en 2001. Un grand nombre de mises à jour et de rajouts ont été faits depuis. Je remercie tous ceux qui m'ont soumis des mises à jour et des corrections.

Le code source XML DocBook de la plus récente version française de ce document à l'adresse : ftp://ftp.traduc.org/pub/traduc.org/doc-vf/HOWTO/telechargement/sgml/Apache-WebDAV-LDAP-HOWTO.xml.

Vous trouverez la plus récente version française de ce document à l'adresse : http://www.traduc.org/docs/howto/lecture/Apache-WebDAV-LDAP-HOWTO.html.

La code source au format XML de la version originale ce document est disponible à http://www.xml-dev.com/xml/Apache-WebDAV-LDAP-HOWTO.xml.

La dernière version originale de ce document est disponible à http://www.xml-dev.com:8080/tldp/http://cvsview.tldp.org/index.cgi/*checkout*/LDP/howto/docbook/Apache-WebDAV-LDAP-HOWTO.xml.

1.2. Contributions au document

Si vous désirez contribuer à la version originale de ce guide pratique, vous pouvez télécharger le code source XML de http://www.xml-dev.com/xml/Apache-WebDAV-LDAP-HOWTO.xml, et envoyer le fichier source modifié à <`saqib CHEZ seagate POINT com`> AVEC VOTRE NOM DANS LA LISTE D'AUTEURS ET DANS L'HISTORIQUE DES VERSIONS :) Cela sera plus facile pour moi de contacter la personne en cas de mises à jour ou de corrections. Je vous remercie.

1.3. Qu'est-ce qu'Apache ?

Le serveur HTTP Apache est un serveur HTTP open-source pour systèmes d'exploitation modernes comme UNIX et Windows NT. Il fournit des services HTTP conformes aux standards HTTP actuels.

Le serveur Web Apache peut être téléchargé librement de http://httpd.apache.org/

1.4. Qu'est-ce que WebDAV ?

WebDAV signifie Web enabled Distributed Authoring and Versioning, c'est-à-dire gestion de publication et de configuration sur Internet. Il fournit un environnement partagé aux utilisateurs pour éditer/gérer leurs fichiers sur les serveurs Web. Techniquement, DAV est une extension du protocole http.

Voici une brève description des extensions fournies par DAV :

Protection contre l'écrasement : mécanisme de verrouillage et de déverrouillage pour éviter les problèmes de synchronisation de mises à jour. Le protocole DAV supporte les accès exclusifs et partagés.

Propriétés : méta-données (titre, sujet, créateur, et cætera)

Gestion des attributs de fichiers : copier, renommer, déplacer et supprimer des fichiers

Contrôle d’accès : limitation d’accès à des ressources diverses. Généralement, DAV considère qu’un contrôle d’accès est déjà en place, et ne fournit pas de mécanisme d’authentification robuste.

Gestion des versions : contrôle de versions des documents. Le contrôle des versions n’est pas encore mis en œuvre.

1.5. Qu’est-ce que PHP ?

PHP (acronyme récursif pour *Processeur Hypertexte PHP*) : c’est un langage de scripts open source à usage général qui est particulièrement adapté au développement Web et qui peut être associé à du HTML.

On peut se procurer PHP de http://www.php.net

1.6. Qu’est-ce que mySQL ?

MySQL, la base de données SQL open source la plus populaire, est développée, distribuée, et maintenue par MySQL AB

On peut télécharger le moteur de base de données de MySQL de http://www.mysql-fr.com/

1.7. Que nous faut-il ?

Les outils nécessaires sont :

i) un compilateur C, c-à-d GCC

ii) un serveur Web Apache 2

iii) le module LDAP pour Apache

iv) les fichiers de la bibliothèque iPlanet LDAP lib

v) le moteur SSL

vi) PHP

vii) Le moteur de base de données mySQL

> **N.B. :**
>
> tous ces paquets sont libres, téléchargeables sur Internet.

1.8. Considérations

Nous considérons que vous avez déjà installé les éléments suivants dans votre système.

i) gzip or gunzip — disponibles à http://www.gnu.org/home.fr.html

ii) gcc et GNU make — disponibles à http://www.gnu.org/home.fr.html

2. Pré-requis

Il est nécessaire de télécharger et de compiler différent paquets. Ce document expliquera le processus de compilation, mais vous êtes sensés savoir installer à partir du code source.

2.1. Éléments essentiels

Il vous faudra une machine sous Solaris ou Linux et un compilateur GCC. Vous aurez aussi besoin de GNU gzip et de GNU tar.

2.2. Apache 2.0.46

Apache est le serveur HTTP, et on l'utilisera pour faire tourner le serveur Web applicatif. Téléchargez les sources d'Apache 2.0.46 depuis http://www.apache.org/dist/httpd/.

2.3. OpenSSL

Il vous faudra télécharger OpenSSL de http://www.openssl.org/source/. Téléchargez la dernière version. L'installation d'OpenSSL sera utilisée pour compiler mod_ssl avec Apache à l'aide des bibliothèques SSL, et pour gérer les certificats SSL sur le serveur Web. Téléchargez les sources d'OpenSSL compressées par gzip dans `/tmp/downloads`

2.4. La bibliothèque iPlanet LDAP

Téléchargez le SDK de iPlanet LDAP de http://wwws.sun.com/software/download/products/3ec28dbd.html. Nous utiliserons le SDK d'iPlanet LDAP, parce qu'il comprend les bibliothèques pour ldaps:// (LDAP over SSL) :

2.5. mod_auth_ldap

Nous utiliserons mod_auth_ldap pour compiler le support LDAP avec Apache. Téléchargez mod_auth_ldap de http://www.muquit.com/muquit/software/mod_auth_ldap/mod_auth_ldap_apache2.html

2.6. Le moteur de base de données mySQL

Téléchargez les exécutables mySQL pour votre plate-forme de http://www-fr.mysql.com/downloads/index.html

2.7. PHP

Téléchargez les sources de PHP de http://www.php.net/downloads.php

3. Installation

Nous nous occuperons d'abord des quelques pré-requis, puis nous procéderons à l'installation principale.

3.1. Pré-requis

Pour installer le serveur d'application, nous avons besoin des bibliothèques SSL et LDAP. Le moteur SSL est lui aussi nécessaire pour gérer les certificats SSL dans Apache 2.x

3.1.1. Le SDK iPlanet LDAP

Devenez root à l'aide de la commande su :

```
$ su -
```

Créez le répertoire `/usr/local/iplanet-ldap-sdk.5`. Copiez le répertoire `ldapcsdk5.08-Linux2.2_x86_glibc_PTH_OPT.OBJ.tar.gz` de `/tmp/downloads` vers `/usr/local/iplanet-ldap-sdk.5`.

```
# mkdir /usr/local/iplanet-ldap-sdk.5
# cp /tmp/downloads/ldapcsdk5.08-Linux2.2_x86_glibc_PTH_OPT.OBJ.tar /usr/local/iplanet-ldap-sdk.5
# cd /usr/local/iplanet-ldap-sdk.5
# tar -xvf ldapcsdk5.08-Linux2.2_x86_glibc_PTH_OPT.OBJ.tar
```

À présent, tous les fichiers de la bibliothèque iPlanet LDAP devraient se trouver dans le bon répertoire.

3.1.2. Le moteur OpenSSL

Ensuite, il nous faut installer le moteur OpenSSL

OpenSSL est une mise en œuvre open source du protocole SSL/TLS. Il est indispensable pour créer et gérer les certificats SSL sur le serveur Web. Cette installation est aussi indispensable pour les fichiers et les bibliothèques qui seront utilisés par le module SSL d'Apache.

Allez dans le répertoire où vous avez placé les fichiers du code source openSSL

```
# cd /tmp/download
# gzip -d openssl.x.x.tar.gz
# tar -xvf openssl.x.x.tar
# cd openssl.x.x
# make
# make test
# make install
```

Après exécution complète de la commande **`make install`** les exécutables openssl devraient se trouver dans le répertoire `/usr/local/ssl`

3.2. mySQL

L'installation de mySQL est très simple. Les binaires téléchargés doivent être placés dans le répertoire approprié.

Nous commençons par créer un utilisateur:groupe pour le démon mysql, et copions les fichiers dans les répertoires appropriés.

```
# groupadd mysql
# useradd -g mysql mysql
# cd /usr/local
# gunzip < /path/to/mysql-VERSION-OS.tar.gz | tar xvf -
# ln -s full-path-to-mysql-VERSION-OS mysql
```

Puis nous lançons le script install_db et changeons les permissions des fichiers

```
# cd mysql
# scripts/mysql_install_db
# chown -R mysql .
```

3.2.1. Démarrer mySQL

Nous lançons maintenant le serveur mySQL pour vérifier l'installation

```
# bin/mysqld_safe --user=mysql &
```

Vérifiez que le démon mySQL est lancé en utilisant la commande **`ps -ef`**. Vous devriez voir s'afficher :

```
# ps -ef | grep mysql
root  3237    1 0 May29 ? 00:00:00 /bin/sh bin/safe_mysqld
mysql 3256 3237 0 May29 ? 00:06:58 /usr/local/mysql/bin/mysqld --defaults-extra-file=/usr/local/mysql/data/my.cnf --basedir=/usr/local/mysql --datadir=/usr/local/mysql/data --user=mysql --pid-file=/us...
```

3.2.2. Arrêter mySQL

Pour arrêter le serveur mySQL, suivez les instructions suivantes

```
# cd /usr/local/mysql
# ./bin/mysqladmin -u root -p shutdown
```

3.2.3. Localiser le répertoire de données

Le démon mySQL place toutes les informations dans un répertoire appelé « répertoire de données ». Si vous avez suivi les instructions d'installation ci-dessus, votre répertoire de données devrait être situé sous `/use/local/mysql/data`.

Pour trouver l'emplacement de votre répertoire de données, utilisez la commande **mysqladmin** comme suit :

```
# /usr/local/mysql/bin/mysqladmin variables -u root --password={votre_mot_de_passe} | grep datadir
```

3.3. Apache 2.0

Commençons par rajouter quelques options de compilation

```
# export LDFLAGS="-L/usr/local/iplanet-ldap-sdk.5/lib/ -R/usr/local/iplanet-ldap-sdk.5/lib/:/usr/local/lib"
# export CPPFLAGS="-I/usr/local/iplanet-ldap-sdk.5/include"
```

Puis décompressez les sources d'Apache 2.0 avec UNTAR, et exécutez le script de `configuration`.

```
# cd /tmp/download
# gzip -d httpd-2.0.46.tar.gz
# tar -xvf httpd-2.0.46.tar
# cd httpd-2.0.46
#./configure --enable-so --with-ssl --enable-ssl --enable-rewrite --enable-dav
```

Exécutez ensuite la commande **make**

```
# make
# make install
```

3.3.1. Démarrer Apache

```
# /usr/local/apache2/bin/apachectl start
```

3.3.2. Arrêter Apache

```
# /usr/local/apache2/bin/apachectl stop
```

3.4. mod_auth_ldap

Décompressez modauthldap_apache2.tar.gz avec Untar

```
cd /tmp/download
# gzip -d modauthldap_apache2.tar.gz
# tar -xvf modauthldap_apache2.tar
# cd modauthldap_apache2
```

À présent, configurez et installez mod_auth_ldap

```
# ./configure --with-apxs=/usr/local/apache2/bin/apxs --with-ldap-dir=/usr/local/iplanet-ldap-sdk.5/
# make
# make install
```

3.5. CERT DB for LDAPS://

Vous devrez aussi télécharger cert7.db and key7.db des sites http://www.xml-dev.com/xml/key3.db et http://www.xml-dev.com/xml/cert7.db et les placer dans le répertoire `/usr/local/ssl`.

3.6. PHP

Décompressez les fichiers source de PHP avec Unzip

```
gzip -d php-xxx.tar.gz
tar -xvf php-xxx.tar
```

Exécutez les commandes configure puis make

```
cd php-xxx
./configure --with-mysql --with-apxs=/usr/local/apache2/bin/apxs
```

Compilez le code source

```
# make
# make install
```

Copiez le fichier php.ini dans le répertoire approprié

```
cp php.ini-dist /usr/local/lib/php.ini
```

4. Configurer et installer les services WebDAV

Et maintenant la partie la plus facile. Dans cette section, nous rendrons un répertoire situé à la racine d'Apache disponible à WebDAV.

4.1. Modifications au fichier `/usr/local/apache/conf/httpd.conf`

Vérifiez que la directive Apache suivante apparaît dans le fichier `/usr/local/apache/conf/httpd.conf` :

```
Addmodule mod_dav.c
```

Si elle n'y est pas, ajoutez la. Cette directive informe Apache du support des fonctionnalités de DAV. La directive doit être placée à l'extérieur des conteneurs.

Ensuite nous devons déterminer où Apache stockera le fichier DAVLockDB. DAVLockDB est une base de données de verrouillage pour WebDAV. le processus httpd doit avoir les droits en écriture dans ce répertoire.

J'enregistre le fichier DAVLock sous `/usr/local/apache/var`. J'utilise aussi ce répertoire pour d'autres besoins. Ajoutez la ligne suivante dans votre fichier `/usr/local/apache/conf/httpd.conf` pour préciser que le fichier DAVLockDB est situé dans le répertoire `/usr/local/apache/var` :

```
DAVLockDB /usr/local/apache/var/DAVLock
```

La directive doit être placée à l'extérieur des conteneurs.

4.2. Créer un répertoire pour DAVLockDB

Comme il est mentionné plus haut, il faut créer un répertoire pour DAVLockDB auquel le processus du serveur Web doit pouvoir accéder en écriture. D'habitude, le processus du serveur Web s'exécute sous l'utilisateur « `nobody` ». Vérifiez-le sur votre système en utilisant la commande :

```
ps -ef | grep httpd
```

à partir de `/usr/local/apache`. Créez le répertoire et définissez ces permissions en utilisant les commandes suivantes :

```
# cd /usr/local/apache
# mkdir var
# chmod -R 755 var/
# chown -R nobody var/
# chgrp -R nobody var/
```

4.3. Donner l’accès à DAV

Donner l’accès à DAV est une tâche insignifiante. Pour autoriser DAV à accéder à un répertoire situé sous la racine d’Apache, ajoutez simplement la directive suivante dans le conteneur de cette directive particulière :

```
DAV On
```

Cette directive autorisera DAV à accéder au répertoire et à ses sous-répertoires.

Ce qui suit est un exemple de configuration activant WebDAV et le service d’authentification LDAP dans `/usr/local/apache/htdocs/DAVtest`. Placez ceci dans le fichier `/usr/local/apache/conf/httpd.conf`.

```
DavLockDB /tmp/DavLock
<Directory "/usr/local/apache2/htdocs/DAVtest">
  Options Indexes FollowSymLinks
  AllowOverride None
  order allow,deny
  allow from all
  AuthName "SMA Development server"
  AuthType Basic
  LDAP_Debug On
  #LDAP_Protocol_Version 3
  #LDAP_Deref NEVER
  #LDAP_StartTLS On
  LDAP_Server you.ldap.server.com
  #LDAP_Port 389
  # If SSL is on, must specify the LDAP SSL port, usually 636
  LDAP_Port 636
  LDAP_CertDbDir /usr/local/apache2/sslcert
  Base_DN "o=SDS"
  UID_Attr uid
  DAV On
  #require valid-user
  require valid-user
  #require roomnumber "123 Center Building"
  #require filter "(&(telephonenumber=1234)(roomnumber=123))"
  #require group cn=rcs,ou=Groups
</Directory>
```

4.4. Créer un répertoire nommé DAVtest

Comme il est mentionné dans une section précédente, le processus du serveur Web doit avoir les droits en écriture dans tous les répertoires DAV. Dans cet exemple nous considérons que le serveur Web s’exécute sous l’utilisateur « `nobody` ». La plupart du temps c’est le cas. Pour vérifier sous quel utilisateur httpd s’exécute, saisissez :

```
# ps -ef | grep httpd
```

Créez un répertoire nommé « `DAVtest` » dans le répertoire `/usr/local/apache/htdocs` :

```
# mkdir /usr/local/apache/htdocs/DAVtest
```

Changez les permissions du répertoire pour le rendre accessible au processus httpd en lecture et écriture. Considérant que httpd s'exécute sous l'utilisateur « `nobody` », utilisez les commandes suivantes :

```
# cd /usr/local/apache/htdocs
# chmod -R 755 DAVtest/
# chown -R nobody DAVtest/
# chgrp -R nobody DAVtest/
```

4.5. Redémarrer Apache

Pour finir, vous devez exécuter la routine du test de configuration fournie avec Apache pour vérifier la syntaxe du fichier `httpd.conf` :

```
# /usr/local/apache/bin/apachectl configtest
```

S'il y a des messages d'erreur, vérifiez que vous avez suivi correctement toutes les étapes mentionnées ci-dessus. Si vous n'arrivez pas à comprendre ce que signifie le message d'erreur, n'hésitez pas à m'envoyer un courrier électronique (en anglais) avec le message d'erreur (<`saqib CHEZ seagate POINT com`>).

Si le test de configuration a réussi, démarrez le serveur Web Apache :

```
# /usr/local/apache/bin/apachectl restart
```

À présent, vous avez un serveur WebDAV Apache utilisant LDAP pour l'authentification et SSL pour le chiffrement.

4.6. Test de conformité au protocole du serveur WebDAV

Il est très important que le serveur WebDAV que nous venons juste d'installer soit totalement compatible avec le protocole WebDAV-2. S'il n'est pas totalement compatible, les applications WebDAV côté client pourront ne pas fonctionner correctement.

Pour tester la compatibilité nous utiliserons un outil nommé Litmus. Litmus est une suite de tests de compatibilité avec le protocole d'un serveur WebDAV, qui est destinée à tester si un serveur est compatible avec le protocole WebDAV selon les spécifications de la norme RFC2518.

Téléchargez les sources de Litmus du site http://www.webdav.org/neon/litmus/ et placez les dans le répertoire /tmp/downloads

Puis utilisez gzip et tar pour extraire les fichiers :

```
# cd /tmp/downloads
# gzip -d litmus-0.6.x.tar.gz
# tar -xvf litmus-0.6.x.tar
# cd litmus-0.6.x
```

Il est facile de compiler et d'installer Litmus :

```
# ./configure
# make
# make install
```

make install installera les fichiers binaires de Litmus dans les répertoires `/usr/local/bin` et les fichiers d'aide dans `/usr/local/man`

Pour tester la compatibilité du serveur WebDAV que vous venez d'installer, recourez à la commande suivante

```
# /usr/local/bin/litmus http://you.dav.server/DAVtest userid passwd
```

5. Administration du serveur WebDAV

Dans cette section, nous aborderons les différentes tâches d'administration — par exemple l'utilisation de LDAP pour le contrôle d'accès, et comment on travaille dans Apache avec DAV

La plupart des changements de configuration pour DAV devront être faits dans le fichier `httpd.conf`. L'emplacement de ce fichier est `/usr/local/apache/conf/httpd.conf`.

`httpd.conf` est un fichier texte qui est utilisé pour la configuration d'Apache. Il peut être édité à l'aide de n'importe quel éditeur de texte — je préfère vi. Faites une copie de sauvegarde de ce fichier avant de le modifier.

Après avoir effectué des modifications au fichier `httpd.conf` le serveur Apache doit être redémarré avec la commande **`/usr/local/apache/bin/apachectl restart`**. Cependant avant de le redémarrer, vous testerez la validité du fichier `httpd.conf` en utilisant la commande **`/usr/local/apache/bin/apachectl configtest`**.

5.1. Limiter les accès aux partages de DAV

Dans la section précédente, quand nous avons créé le partage DAVtest, nous avons utilisé LDAP pour l'authentification. Cependant, n'importe qui pouvant s'authentifier en utilisant son compte_utilisateur/mot_de_passe pourra accéder à ce dossier.

En utilisant la directive `require` dans le fichier httpd.conf, vous pouvez limiter l'accès à certains individus ou groupes d'individus.

Si nous regardons la configuration de DAVtest de la précédente section :

```
<Directory /usr/local/apache/htdocs/DAVtest>

Dav On
#Options Indexes FollowSymLinks
AllowOverride None
order allow,deny
allow from all
AuthName "LDAP_userid_password_required"
AuthType Basic

<Limit GET PUT POST DELETE PROPFIND PROPPATCH MKCOL COPY MOVE LOCK UNLOCK>

Require valid-user
```

```
</Limit>

LDAP_Server ldap.server.com
LDAP_Port 389
Base_DN "o=ROOT"
UID_Attr uid

</Directory>
```

nous voyons que la commande `require` a pour paramètre `valid-user`. Ce qui signifie que n'importe quel utilisateur authentifié peut accéder à ce dossier.

5.1.1. Limitations d'accès basées sur les UID individuels

Les UID de LDAP peuvent être utilisés pour limiter les accès au dossier DAV.

La directive require `valid-user` peut être remplacée par `require user 334455 445566`

Ceci limitera l'accès aux individus ayant pour UID 334455 et 445566. Personne d'autre ne pourra accéder à ce dossier.

5.1.2. Limitations d'accès basées sur des groupes d'individus

La directive `require` peut aussi être utilisée pour limiter les accès à des groupes d'individus. On peut le faire en utilisant soit les groupes de LDAP, soit les filtres de LDAP. Le filtre doit avoir une syntaxe de filtre LDAP valide.

5.2. Limiter l'accès en écriture à des partages DAV

On peut avoir besoin de limiter l'accès en écriture aux ressources des partages DAV à une certaine personne, en laissant toutefois n'importe qui voir les ressources. On peut le faire facilement en utilisant les balises `<Limit>` dans le fichier httpd.conf

```
<Directory /usr/local/apache/htdocs/DAVtest>
Dav On
#Options Indexes FollowSymLinks
AllowOverride None
order allow,deny
allow from all
AuthName "LDAP_userid_password_required"
AuthType Basic

<Limit GET PUT POST DELETE PROPFIND PROPPATCH MKCOL COPY MOVE LOCK UNLOCK>

Require valid-user

</Limit>

LDAP_Server ldap.server.com
LDAP_Port 389
Base_DN "o=ROOT"
UID_Attr uid
</Directory>
```

Vous limiterez l’accès en écriture à certains utilisateurs en changeant la balise `<limit>` en

```
<Limit PUT POST DELETE PROPPATCH MKCOL COPY MOVE LOCK UNLOCK>

Require 334455

</Limit>
```

En fait, nous limitons les méthodes PUT POST DELETE PROPPATH MKCOL COPY MOVE LOCK et UNLOCK à l’utilisateur qui a pour UID 334455. N’importe qui d’autre pourra employer les méthodes GET et PROPFIND pour les ressources, mais aucune autre méthode.

6. Mettre en œuvre et utiliser SSL pour protéger le trafic HTTP

De nos jours, la sécurité des données stockées sur un serveur de fichiers est très importante. Des données compromises peuvent coûter des milliers de dollars à une entreprise. Dans la dernière section, nous avons compilé le module d’authentification LDAP dans Apache pour fournir un mécanisme d’authentification. Cependant, le trafic http est très peu sur, et toutes les données sont transférées en clair — ce qui signifie que l’authentification LDAP (utilisateur/mot_de_passe) sera transmise elle aussi en clair. Ceci pose un problème. N’importe qui peut intercepter cet utilisateur/mot_de_passe et accéder aux dossiers de DAV. Pour éviter ceci nous devrons chiffrer le trafic http, essentiellement par HTTP + SSL ou HTTPS. Tout ce qui est transféré en HTTPS est chiffré, ce qui fait que le couple utilisateur/mot_de_passe LDAP ne peut pas être aisément déchiffré. HTTPS tourne sur le port 443. Les binaires résultants étant compilés selon la dernière section, Apache pourra écouter à la fois sur les ports 80 (HTTP normal) et 443 (HTTPS). Si vous désirez utiliser ce serveur uniquement pour DAV, alors je vous suggère fortement de fermer le port 80. Dans cette section du guide pratique, je fournirai des informations sur SSL et comment l’administrer dans un serveur http Apache.

6.1. Introduction à SSL

SSL (Secure Socket Layer) est une couche protocolaire qui se situe entre la couche Réseau et la couche Application. Comme son nom le suggère, SSL fournit un mécanisme de déchiffrement pour toutes sortes de trafic : LDAP, POP, IMAP et plus important, HTTP.

Ce qui suit est une structure ultra simplifiée des couches impliquées par SSL.

```
+-------------------------------------------+
|   LDAP    |   HTTP    |    POP    |   IMAP    |
+-------------------------------------------+
|                    SSL                    |
+-------------------------------------------+
|               Couche réseau               |
+-------------------------------------------+
```

6.1.1. Algorithmes de cryptographie utilisés par SSL

SSL utilise trois sortes de techniques de cryptographie : les systèmes de clés publiques-privées, de clés symétriques et de signatures numériques [p 25] .

Chiffrement par clés publiques-privées — Initialisation d'une connexion SSL : dans cet algorithme, le chiffrement et le déchiffrement sont effectués en utilisant une paire de clés publiques et privées. Le serveur Web détient la clé privée, et envoie la clé publique au client dans le certificat.

1. Le client demande un contenu au serveur Web en utilisant HTTPS.
2. Le serveur Web répond en envoyant un certificat numérique qui comprend la clé publique du serveur.
3. Le client vérifie si le certificat est expiré.
4. Puis le client vérifie si l'autorité de certification qui a signé le certificat est une autorité de confiance figurant dans la liste du navigateur. Ceci explique pourquoi il est nécessaire d'obtenir un certificat d'une autorité de certification de confiance.
5. Puis, le client vérifie si le nom de domaine pleinement qualifié (FQDN) du serveur Web coïncide avec le Nom Commun (Common Name CN) du certificat.
6. Si tout est correct, la connexion SSL est initialisée.

N.B. :

On ne peut déchiffrer ce qui a été chiffré avec une clé privée qu'avec sa clé publique. De la même façon, on ne peut déchiffrer ce qui a été chiffré avec une clé publique qu'avec sa clé privée. C'est une erreur répandue de penser qu'une clé publique est utilisée pour le chiffrement et que la clé privée est utilisée pour le déchiffrement. Ce n'est pas le cas. On peut utiliser les deux clés pour chiffrer ou déchiffrer. Cependant, si on utilise une clé pour chiffrer, alors l'autre clé devra servir à déchiffrer. Par exemple On ne peut chiffrer un message puis le déchiffrer en utilisant uniquement une clé publique.

L'utilisation d'une clé privée pour chiffrer et d'une clé publique pour déchiffrer garantit l'identité de l'émetteur (qui est le propriétaire de la clé publique) à ses destinataires. L'utilisation d'une clé publique pour chiffrer et d'une clé privée pour déchiffrer garantit que seul le destinataire (qui est le propriétaire de la clé publique) accédera aux données. (c'est-à-dire que seul le détenteur de la clé privée pourra déchiffrer le message).

Chiffrement symétrique — Transmission effective des données : une fois la connexion SSL établie, on utilise le chiffrement symétrique, qui est moins consommateur en cycles de processeur. Avec le chiffrement symétrique, on peut chiffrer et déchiffrer les données en utilisant la même clé. La clé de chiffrement symétrique est échangée durant le processus d'initialisation, en utilisant la clé de chiffrement publique.

Sommation de messages Le serveur utilise des algorithmes de sommation de messages comme HMAC [p 25] , SHA-1 [p 26] , MD5 [p 26] pour vérifier l'intégrité des données transférées.

6.1.2. Garantie d'authenticité et d'intégrité

Processus de chiffrement

```
   Clef privée              Clef publique
  de l'émetteur            du destinataire
  ,-.                      ,-.
 (   ).........           (   ).........
```

```
         `_' ''''|'|'||          `_' ''''''''||
                 | |                    |
                 | |                    |
  .----------.   | |    .----------.    |     .----------.
  |  Texte   |   V |    |   Texte  |    V     |   Texte  |
  |   en     |--------->|  chiffré |--------->|  chiffré |
  |  clair   | Étape1   |     1    | Étape2   |     2    |\
  `----------'     |    `----------'          `----------' \    __
        |          |                                        \   [_'
        |          |                                  Étape5 \   |
        |Étape3    |                                       __   --|--
        |          |                                  _.--'      |
        V          |                            _..-''          / \
   .---------.     |    .---------.     _..-''            Destinataire
   |  SHA 1  |     V    |Signature| _..-''
   |SomMessag|--------->|numérique|'
   `---------' Étape4   `---------'
             _  ___ ___
       _    (_)/  _)  _)                                      _
  ____| |__  _ | |_| |_ ____ _____ _____  _____  _____ ____ _| |_
 / ___)  _ \| ||  _)  _) ___) ___ |  _  \|  _  \| ___ |  _ (_   _)
( (___  | | | || | | || |   | ____| || | | || | | ____| | | || |_
 \____)_| |_|_||_| |_||_|   |_____)_||_|_|_||_|_|_____)_| |_| \__)
```

- Étape 1 : le message original en clair est chiffré avec la clé privée de l'émetteur, ce qui produit le texte chiffré 1. L'authenticité de l'émetteur est garantie.
- Étape 2 : le « texte chiffré 1 » est chiffré à l'aide de la clé publique du destinataire, aboutissant au « texte crypté 2 ». Celui-ci garantira l'authenticité du destinataire, c'est-à-dire que seul le destinataire peut déchiffrer le message à l'aide de sa clé privée.
- Étape 3 : la somme SHA1 du « texte en clair » est créée.
- Étape 4 : la somme SHA1 du message est ensuite chiffrée avec la clé privée de l'émetteur, ce qui produit la signature numérique du « texte en clair ». Le destinataire peut utiliser la signature numérique pour s'assurer de l'intégrité du message et de l'authenticité de l'émetteur.
- Étape 5 : la « signature numérique » et le « texte chiffré 2 » sont ensuite envoyés au destinataire.

Processus de déchiffrement

```
          Clef privée          Clef publique
        du destinataire        de l'émetteur
        ,-.                     ,-.
       (   )..........         (   )..........
        `_' ''''''''||          `_' '''''''|||
               |                      |    |
               |                      |    |
 .----------.  |       .----------.   |    | .----------.
 |  Texte   |  V       |  Texte   |   V    | |  Texte   |       .---No1---.
 | chiffré  |--------->| chiffré  |--------->|   en     |------>|  SHA 1  |
 |    2     | Étape1   |    1     | Étape2 | |  clair   |Étape3 |SomMessag|
 `----------'          `----------'        | `----------'       `---------'
                                           |                        ||
                                           |                        ||Étape5
                                           |                        ||
                                           |                        ||
                              .---------.  |                    .---------.
                              | Digital |  V                    |  SHA 1  |
```

```
                                 |Signature|----------------------->|SomMessag|
                                 `---------' Étape4                 `---No2---'
    _    _              _  ___ ___
   | |  //        _    (_)/  _)  _)                                          _
 __| |_____  ____| |__  _ | |_| |_ ____ _____ _____  _____  _____ ____ _| |_
/ _  | ___ |/ ___)  _ \| ||  _)  _) ___) ___ |  _  \|  _  \| ___ |  _ (_   _)
( (_| | ____( (___ | | | || | || |   | ____| || | || | | ____| | | || |_
 \____|_____)\____)_| |_|_||_| |_||_|   |_____)_||_|_|_||_|_|_____)_| |_| \__)
```

- Étape 1 : le « Texte chiffré 2 » est déchiffré avec la clé privée du destinataire, ce qui produit le texte chiffré 1.
- Étape 2 : le « texte chiffré 1 » est déchiffré à l'aide de la clé publique de l'émetteur, ce qui produit le « texte en clair ».
- Étape 3 : la somme SHA1 du « texte en clair » est créée.
- Étape 4 : la « signature numérique » est ensuite déchiffrée à l'aide de la clé publique de l'émetteur, ce qui produit la « somme SHA 1 du message ».
- Étape 5 : la « somme SHA 1 du message numéro 1 » est ensuite comparée à la « somme SHA 1 du message numéro 2 ». Si elles sont égales, cela signifie que les données n'ont pas été modifiées durant la transmission, et que l'intégrité de l'original « texte en clair » a été préservée.

6.2. Certificats de test

Lorsque nous compilons Apache, nous créons un certificat de test. Nous avons utilisé le makefile fourni par mod_ssl pour créer ce certificat sur mesure. Nous avons utilisé la commande :

```
# make certificate TYPE=custom
```

Nous pourrons utiliser ce certificat à des fins de test.

6.3. Certificats destinés à la production

Il est nécessaire d'obtenir un certificat d'une Autorité de Certification de confiance (nommée ci-après AC) pour une utilisation en production. Les autorités de certification sont des vendeurs de certificats, qui figurent dans la liste des AC de confiance de chaque navigateur. Comme on l'a précisé dans la section des algorithmes de cryptographie, si l'AC ne figure pas dans la liste des autorités de confiance, un message d'alerte s'affichera quand l'utilisateur essayera de se connecter à un site sécurisé.

Les certificats de test provoqueront eux aussi l'apparition d'un message d'alerte dans le navigateur de l'utilisateur.

6.4. Génération d'un CSR

Pour être signée, une CSR (Certificate Signature Request: Demande de Signature de Certificat) doit être envoyée à une AC de confiance. Cette section montre comment on crée une CSR, et comment on l'envoie à l'AC de son choix.

```
# openssl req
```

Pour créer une CSR, on peut recourir à cette commande comme suit :

```
# cd /usr/local/apache/conf/
# /usr/local/ssl/bin/openssl req -new -nodes -keyout private.key -out public.csr
Generating a 1024 bit RSA private key
...........++++++
....++++++
writing new private key to 'private.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:California
Locality Name (eg, city) []:San Jose
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Seagate
Organizational Unit Name (eg, section) []:Global Client Server
Common Name (eg, YOUR name) []:xml.seagate.com
Email Address []:saqib@seagate.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:badpassword
An optional company name []:
```

« PRNG not seeded »

Si le fichier `/dev/random` n'existe pas sur votre système, le message d'erreur « `PRNG not seeded` » s'affichera. Dans ce cas, vous pouvez utiliser la commande suivante :

```
# /usr/local/ssl/bin/openssl req -rand mon_fichier.ext -new -nodes -keyout private.key -out public.csr
```

Remplacez le fichier mon_fichier.ext par le nom d'un fichier existant dans votre système. Vous pouvez spécifier n'importe quel fichier. Openssl utilisera ce fichier pour générer le noyau.

Sur Solaris 9 on trouve le fichier `/dev/random` . Cependant, il est possible que vous ayez à installer le correctif 112438 pour accéder à /dev/random

Arrivé là, vous devrez répondre à plusieurs questions concernant votre serveur pour générer la CSR.

N.B. : Votre Common Name (CN) est le nom DNS pleinement qualifié (FQDN) de votre serveur web, c'est-à-dire dav.server.com . Si vous saisissez quelque chose d'autre, ça ne marchera PAS. Mettez de côté le mot de passe pour un usage ultérieur.

Une fois le processus achevé, un fichier `private.key` et un fichier `public.csr` seront présents dans votre arborescence. Il vous faudra envoyer le fichier `public.csr` à l'autorité de certification. À ce stade, le fichier public.key n'est pas chiffré. pour le chiffrer, saisissez :

```
# mv private.key private.key.unecrpyted
# /usr/local/ssl/bin/openssl rsa -in private.key.unecrpyted -des3 -out private.key
```

6.5. Installation de la clé privée et du certificat du serveur

une fois que l'autorité de certification aura traitée votre demande, elle vous renverra un certificat codé (certificat numérique). Le certificat numérique est au format défini par la norme X.509 v3. Les lignes qui suivent montre la structure d'un certificat numérique conforme à X509 v3 (version française entre parenthèses)

- Certificat
  - Version (Version)
  - Serial Number (Numéro de série)
  - Algorithm ID (Identification de l'algorithme)
  - Issuer (Émetteur)
  - Validity (Validité)
  -
    - Not Before (pas avant)
    - Not After (pas après)
  - Subject (sujet)
  - Subject Public Key Info (Info de sujet de clé publique)
  -
    - Public Key Algorithm (algorithme de clé publique)
    - RSA Public Key (clé publique RSA)
  - Extensions (Extensions)
- Certificate Signature Algorithm (algorithme de signature du certificat)
- Certificate Signature (signature du certificat)

6.5.1. Vérification d'un certificat numérique

Pour vérifier un certificat X.509, utilisez la commande suivante :

```
# openssl verify server.crt
server.crt: OK
```

où `server.crt` est le nom du fichier qui contient le certificat numérique.

6.5.2. Vérification du contenu d'un certificat numérique

On peut voir le contenu d'un certificat numérique en utilisant la commande **`# openssl x509`** comme suit :

```
# openssl x509 -text -in server.crt
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 312312312 (0x0)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=US, O=GTE Corporation, CN=GTE CyberTrust Root
        Validity
            Not Before: Feb  8 03:25:50 2000 GMT
            Not After : Feb  8 03:25:50 2001 GMT
            Subject: C=US, ST=New York, L=Pelham, O=xml-dev, OU=web, CN=www.xml-dev.com/Email=saqib@xml-dev.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                ............
                ............
                Exponent: 65537 (0x10001)
    Signature Algorithm: md5WithRSAEncryption
        ............
        ............
```

6.5.3. Installation des certificats : modification du fichier httpd.conf

Vous devrez placer ce certificat dans le serveur, et indiquer à Apache où le trouver.

Dans cet exemple, la clé privée est située dans le répertoire `/usr/local/apache2/conf/ssl.key/` et le certificat du serveur est placé dans le répertoire `/usr/local/apache2/conf/ssl.crt/`.

Copiez en le renommant le fichier reçu de l'autorité de certification en `server.crt` dans le répertoire `/usr/local/apache2/conf/ssl.crt/`.

et placez le fichier private.key généré à l'étape précédente dans le répertoire `/usr/local/apache2/conf/ssl.key/`

Puis modifiez le fichier `/usr/local/apache2/conf/ssl.key/` pour qu'il pointe correctement vers la clé privée et le certificat du serveur :

```
#   Server Certificate:
#   Point SSLCertificateFile at a PEM encoded certificate.  If
#   the certificate is encrypted, then you will be prompted for a
#   pass phrase.  Note that a kill -HUP will prompt again.  Keep
#   in mind that if you have both an RSA and a DSA certificate you
#   can configure both in parallel (to also allow the use of DSA
#   ciphers, etc.)
SSLCertificateFile /usr/local/apache2/conf/ssl.crt/server.crt
#SSLCertificateFile /usr/local/apache2/conf/ssl.crt/server-dsa.crt

#   Server Private Key:
#   If the key is not combined with the certificate, use this
#   directive to point at the key file.  Keep in mind that if
#   you've both a RSA and a DSA private key you can configure
#   both in parallel (to also allow the use of DSA ciphers, etc.)
SSLCertificateKeyFile /usr/local/apache2/conf/ssl.key/private.key
#SSLCertificateKeyFile /usr/local/apache2/conf/ssl.key/server-dsa.key
```

6.6. Annulation de la phrase de passe pour la clef privée RSA

La clé privée RSA conservée sur le serveur Web est d’habitude chiffrée, et il vous faut une phrase de passe pour parcourir le fichier. Voilà pourquoi quand Apache est lancé avec modssl, une phrase de passe vous est demandée :

```
# apachectl startssl
Apache/1.3.23 mod_ssl/2.8.6 (Pass Phrase Dialog)
Some of your private key files are encrypted for security reasons.
In order to read them you have to provide us with the pass phrases.
Server your.server.dom:443 (RSA)
Enter pass phrase:
```

Il est très important de chiffrer une clé privée RSA. Si un pirate s’empare de votre clé privée RSA non chiffrée, il pourra facilement emprunter l’identité de votre serveur Web. Si la clé est chiffrée, la seule chose que pourra faire le pirate est de tenter une attaque en force brute sur votre phrase de passe. L’utilisation d’une phrase de passe robuste (c’est-à-dire longue) est encouragée.

Cependant, le fait de chiffrer la clé peut parfois être gênant, dans la mesure où vous devrez saisir la phrase de passe à chaque démarrage du serveur Web. En particulier si vous utilisez les scripts rc pour lancer le serveur Web au démarrage, le processus de démarrage sera stoppé sur l’invite de saisie d’une phrase de passe.

Vous pouvez facilement vous débarrasser de l’invite de saisie de la phrase de passe en déchiffrant la clé. Cependant, assurez-vous que personne ne pourra s’emparer de cette clé. Je ne saurais trop vous recommander d’appliquer les lignes de conduite de durcissement et de sécurisation du serveur avant de déchiffrer la clé du serveur Web.

Pour déchiffrer la clé :

tout d’abord, faites une copie de la clé chiffrée

```
# cp server.key server.key.cryp
```

Puis recréez la clé avec chiffrement. L’invite vous demandera la phrase de passe de la clé chiffrée d’origine

```
# /usr/local/ssl/bin/openssl rsa -in server.key.cryp -out server.key
read RSA key Enter PEM pass phrase: writing RSA key
```

Une façon de sécuriser la clé privée non chiffrée est de limiter l’accès en lecture à l’utilisateur root :

```
# chmod 400 server.key
```

6.7. Réglage des performances SSL

6.7.1. Cache de session SSL inter-processus

Le modèle de fonctionnement d’Apache est multi-processus ; toutes les requêtes ne seront PAS prises en charge par le même processus. L’information sur la session SSL se perd donc quand un client effectue de multiples requêtes. De multiples échanges de données SSL provoquent une surcharge du système sur le serveur Web et le client. Pour éviter cela, les informations de session SSL doivent être stockées dans un cache de session inter-processus, pour permettre à tous les processus d’accéder aux informations protocolaires. On peut spécifier l’emplacement du cache de session SSL dans la directive

SSLSessionCache dans le fichier `/usr/local/apache2/conf/ssl.key/` :

```
SSLSessionCache         shmht:logs/ssl_scache(512000)
#SSLSessionCache        shmcb:logs/ssl_scache(512000)
#SSLSessionCache        dbm:logs/ssl_scache
SSLSessionCacheTimeout  300
```

L'utilisation de dbm:logs/ssl_scache crée un cache de type fichier de hachage DBM sur le disque local.

L'utilisation de shmht:logs/ssl_scache(512000) crée un cache dans un segment de mémoire partagée

shmht contre shmcb

shmht : recoure à une table de hachage pour cacher les informations du protocole SSL dans la mémoire partagée.

shmcb : recoure à un tampon cyclique pour cacher les informations du protocole SSL dans la mémoire partagée.

N.B. :

tous les OS/plates-formes ne supportent pas de créer des tables de hachage dans la mémoire partagée. Donc, il faut utiliser dbm:logs/ssl_scache à la place.

6.7.2. Vérification du cache de session SSL

Pour vérifier si le cache de session SSL fonctionne correctement, vous devez utiliser la commande **openssl** avec l'option `-reconnect` comme suit :

```
# openssl s_client -connect your.server.dom:443 -state -reconnect
CONNECTED(00000003)
.......
.......
Reused, TLSv1/SSLv3, Cipher is EDH-RSA-DES-CBC3-SHA
SSL-Session:
.....
Reused, TLSv1/SSLv3, Cipher is EDH-RSA-DES-CBC3-SHA
SSL-Session:
.....
Reused, TLSv1/SSLv3, Cipher is EDH-RSA-DES-CBC3-SHA
SSL-Session:
.....
Reused, TLSv1/SSLv3, Cipher is EDH-RSA-DES-CBC3-SHA
SSL-Session:
.....
Reused, TLSv1/SSLv3, Cipher is EDH-RSA-DES-CBC3-SHA
SSL-Session:
.....
```

`-reconnect` oblige le s_client à se connecter au serveur 5 fois de suite en utilisant la même ID de session SSL. Comme vous le voyez plus haut, vous devriez voir cinq tentatives de réutilisation de la même ID de session.

A. Outils d'évaluation de performances HTTP/HTTPS

Vous trouverez ci-dessous une liste de quelques outils d'évaluation de performances OpenSource pour serveurs Web

i) SSLswamp — pour un audit de performances lors de la connexion à un serveur SSL autorisé

ii) HTTPERF — un outil pour mesurer les performances d'un serveur Web

iii) ab — outil d'évaluation d'un serveur HTTP Apache

B. Solutions matérielles basées sur le chiffrement SSL

Des solutions de chiffrement SSL matérielles sont présentées ci-dessous :

i) CHIL (Cryptographic Hardware Interface Library) fabriqué par nCipher

ii) ab — outil d'évaluation d'un serveur HTTP Apache

C. Autorités de certification

La liste qui suit présente des autorités de certification acceptées par les différents navigateurs :

i) Baltimore

ii) Entrust

iii) GeoTrust

iv) Thawte

v) TrustCenter

Glossaire de termes PKI

A

Asymmetric Cryptography : Chiffrement asymétrique

Une paire de clés publiques et privées est utilisée dans ce type de chiffrement. La clé privée est secrète et la clé publique est distribuée à volonté.

C

Certificat

Enregistrement contenant des informations conformes au format format X.509. [p 20] .

Certificate Authority (CA) : Autorité de Certification

Émetteur d’un certificat numérique. De plus, il valide l’identité de l’entité finale propriétaire du certificat numérique.

Certificate Signing Request (CSR) : demande de signature de certificat

Une demande de signature de certificat est ce que vous envoyez à une autorité de certification (CA) pour vous inscrire. Une CSR contient la clé publique de l’entité finale effectuant la demande de certificat numérique.

Common Name (CN) : Nom Commun

Le Common Name est le nom de l’entité finale c.-à-d. Saqib Ali. Si l’entité finale est un serveur Web, le Common Name (CN) est le nom DNS pleinement qualifié (FQDN) du serveur web.

D

Certificat numérique

Certificat qui relie une clé publique à un sujet (entité finale). Ce certificat contient également d’autres informations (définies dans le format X.509) sur le sujet. [p 20] Il est signé par l’autorité de certification, à l’aide de la clé privée de l’autorité de certification, c’est-à-dire à l’aide d’un certificat numérique [p 21]

Digital Signature : signature numérique

On crée une signature numérique en signant la somme du message (hachage du message) à l’aide de la clé privée. Elle garantit l’identité de l’émetteur et l’intégrité des données.

E

Entité finale

Entité protagoniste dans la PKI (infrastructure de clé publique). Il s’agit normalement d’un serveur, d’un service ou d’une personne. Une autorité de certification n’est pas une entité finale. Pour une autorité de certification, l’auteur de la demande est une entité finale.

H

Hash: hachage

Un hachage est un nombre hexadécimal généré à partir d’une chaîne de texte, de telle façon que deux chaînes différentes ne puissent produire le même hachage.

HMAC : (*Keyed Hashing for Message Authentication*) : clé de hachage pour l’authentification d’un message

HMAC est une mise en œuvre de l’algorithme « code d’authentification de messages » MAC.

M

Message Authentication Code : code d'authentification de messages

Analogue à une sommation de message (hachage/empreintes digitales), à ceci près qu'on utilise la clé secrète partagée pour calculer le hachage. Étant donné qu'une clé secrète partagée est utilisée, aucun attaquant ne peut changer la somme du message. Cependant, une clé secrète partagée doit tout d'abord être communiquée aux entités participantes, contrairement à une signature numérique pour laquelle une somme de messages est signée à l'aide de la clé privée. HMAC est un exemple d'un algorithme de code d'authentification de messages.

Message Digest 5 — MD5 : sommation de messages 5 — somme MD5

Message Digest 5 (MD5) est un ensemble de fonctions de hachage unidirectionnelles à 128 bits

P

Private Key : clé privée

En chiffrement asymétrique, la clé privée est celle qui est tenue secrète par le propriétaire (entité finale). Elle peut être utilisée pour le chiffrement ou le déchiffrement.

Public Key : clé publique

En chiffrement asymétrique, la clé publique est celle qui distribuée librement. Elle peut être utilisée pour le chiffrement ou le déchiffrement.

Public Key Infrastructure (PKI) : infrastructure de clé publique

Infrastructure de clé publique

S

SHA-1 (Secure Hash Algorithm) algorithme de hachage sécurisé

Secure Hash Algorithm (SHA-1) est une fonction de hachage unidirectionnelle à 160 bits. La taille maximum d'un message est de 2^64 bits.

Secure Socket Layer (SSL)

Secure Socket Layer (SSL) est un protocole de sécurité qui fournit des services d'authentification (certificats numériques), de confidentialité (chiffrement) et d'intégrité des données (sommation de messages — MD5, SHA, et cætera).

Chiffrement symétrique

Dans ce type de chiffrement, le message est chiffré et déchiffré par la même clé. (((n^2-n))/2) clés sont nécessaires pour n utilisateurs désirant utiliser ce système de chiffrement.