Guide pratique des certificats SSL

Version française du *SSL Certificates HOWTO*

Franck Martin

28 février 2003

Historique des versions

Version v0.5.fr.1.0	2003−02−28	Revu par : FR
Traduction française de François Romieu. Relecture de Pierre Machard. Traduction de la licence et quelques retouches de Jean−Philippe Guérard <jean−philippe.guerard@laposte.net>.		
Version v0.5	2002−10−20	Revu par : FM
Ajout des information de Nate Carlson <natecars@natecarlson.com> sur IPsec Ajout des informations de Bill Shirley <webnut@telocity.com> sur IMAPS et POPS Ajout des informations de Colin McKinnon <colin@wew.co.uk> sur WinCrypt.		
Version v0.4	2002−06−22	Revu par : FM
Corrections diverses ajout d'illustrations en ASCII		
Version v0.3	2002−05−09	Revu par : FM
Ajout d'informations sur les extensions x509v3 Corrections de fautes		
Version v0.2	2001−12−06	Revu par : FM
Ajout du fichier openssl.cnf Ajout d'information sur CRL de la part d'Averroes <a.averroes@libertysurf.fr> Corrections de fautes		
Version v0.1	2001−11−18	Revu par : FM
Création du guide pratique		

Une expérience de gestion d'une autorité de certification. La création et la signature de certificats destinés à la sécurisation des transactions web, au courrier, à la signature du code et à d'autres usages.

Table des matières

Chapitre 1. Principes généraux

1.1. Introduction

Comme moi, vous avez sûrement lu avec attention les pages de manuel des applications fournies par le projet OpenSSL et, comme moi autrefois, vous ne voyez pas trop par où commencer ni comment travailler de façon sécurisée avec des certificats. Ce document répond à l'essentiel de vos questions.

Ce guide pratique traite aussi d'applications hors du monde Linux. Il ne sert à rien d'émettre des certificats qu'on ne peut pas utiliser & Toutes les applications ne figurent pas ici. Envoyez moi donc s'il vous plaît vos ajouts et corrections. On peut me contacter en anglais à l'adresse suivante : `<franck@sopac.org>`.

La version originale de ce guide pratique est publié via le Projet de documentation Linux. Vous y trouverez la dernière version originale de ce document.

La version française de ce document est publiée par le projet de traduction traduc.org. Vous y trouverez notamment la dernière version française de ce document.

1.1.1. Droits d'utilisation et limitation de responsabilité

Ce document est distribué dans l'espoir qu'il se révélera utile, mais SANS AUCUNE GARANTIE sans même la garantie implicite qu'il soit PROPRE À LA VENTE ou ADAPTÉ À UN BESOIN PARTICULIER.

En résumé, si les conseils qui vous sont prodigués ici annihilent la sécurité de votre application de commerce électronique, eh bien, tant pis pour vous   ce n'est jamais de notre faute. Désolé.

Vous pouvez librement copier et distribuer (commercialement ou gratuitement) ce document dans n'importe quel format. Il vous est demandé de faire parvenir vos corrections et commentaires au responsable actuel de ce document. Vous pouvez créer des travaux dérivés de celui−ci et les distribuer si vous respectez les conditions suivantes :

1. Faire parvenir vos travaux dérivés (dans le format le plus approprié tel que SGML) au Projet de documentation Linux (LDP), ou à un projet du même type pour qu'il soit diffusé sur internet. Si vous n'envoyez pas votre document au LDP, informez le LDP de l'endroit où votre document est disponible.
2. Diffuser vos travaux dérivés sous la même licence, ou sous la licence publique GNU (GPL). Inclure la notice précisant les droits d'utilisation, et au moins un pointeur sur la licence utilisée.
3. Reconnaître la contribution des précédents auteurs et contributeurs principaux. Si vous envisagez de réaliser un travail dérivé autre qu'une traduction, il vous est demandé d'en discuter au préalable avec le responsable actuel de ce document.

Il vous est également demandé que si vous publiez ce guide pratique dans un format papier, vous envoyiez à ses auteurs quelques exemplaires à des fins d'*évaluation* :−). Vous voudrez peut−être également m'envoyer quelque chose pour assaisonner mes épinards ;−)

1.1.2. Disclaimer and Licence

La licence de ce document nous oblige à inclure une copie en anglais de ce texte. La version française de ce texte se trouve à la section précédente.

1.1.3. Pré−requis

Comme indiqué dans l'introduction, ce document est un aide−mémoire et vous devez donc consulter les pages de manuel du paquet OpenSSL. La lecture de documents relatifs à la sécurité est vivement conseillée afin de comprendre comment votre sécurité peut être compromise. Les certificats sont destinés à améliorer la sécurité des transactions et il donc important que vous compreniez les implications de vos actes en terme de sécurité ainsi que les limitations d'OpenSSL.

1.2. Que sont SSL et les certificats ?

Le protocole SSL (*Secure Socket Layer*) a été créé par Netscape pour sécuriser les transactions entre les serveur web et les outils de navigation. Il a recours à un tiers, l'autorité de certification (*CA/Certificate Authority*) qui identifie n'importe laquelle des extrémités ou les deux. Il s'agit du fonctionnement en très résumé.

1. Un navigateur demande une page web sécurisée (en général https://).
2. Le serveur web émet sa clef publique accompagnée de son certificat.
3. Le navigateur vérifie que le certificat a été émis par un tiers fiable (en général une autorité de certification racine), qu'il est toujours valide et qu'il se rapporte bien au site en cours.
4. Le navigateur emploie la clef publique du serveur pour chiffrer une clef de chiffrement symétrique aléatoire et l'envoie au serveur web avec l'URL demandée et diverses données HTTP chiffrées.
5. Le serveur web déchiffre la clef de chiffrement symétrique grâce à sa sa clef privée et utilise la clef de chiffrement symétrique pour récupérer l'URL et les données HTTP.
6. Le serveur renvoie le document html et les données HTTP chiffrées avec la clef symétrique.

7. Le navigateur déchiffre l'ensemble avec la clef symétrique et affiche les informations.

Plusieurs concepts méritent d'être assimilés.

1.2.1. Clef publique / clef privée

Le chiffrement à base de clef publique garantit qu'une donnée chiffrée par une clef, publique ou privée, ne peut être déchiffrée que par la clef associée. Cela peut paraître surprenant mais croyez moi, ça fonctionne. Les clefs sont de même nature et leurs rôles peuvent être inversés : ce que l'une chiffre est déchiffrable par l'autre. La paire de clef repose sur des nombres premiers et leur longueur, exprimée en digits binaires, traduit la difficulté qu'il y a à déchiffrer un message sans connaissance à priori de la clef nécessaire. L'astuce consiste à conserver une clef secrète (la clef privée) et à distribuer l'autre clef (la clef publique) à tout le monde. N'importe qui peut vous envoyer un message chiffré mais personne d'autre que vous ne peut le déchiffrer tant que vous êtes le seul à conserver un des deux membres de la paire de clefs. On peut également prouver qu'un message provient de vous en le chiffrant avec votre clef privée : seule la clef publique correspondante le déchiffrera. Notez que le message n'est pas protégé parce que vous le signez. Tout le monde a accès à la clef de déchiffrement, ne l'oubliez pas.

Le problème consiste à obtenir la clef publique du correspondant. En général, vous lui demanderez de la transmettre via un message signé qui la contiendra accompagnée d'un certificat.

```
Message-->[Clef publique]-->Message chiffré-->[Clef privée]-->Message
```

1.2.2. Le certificat

Qu'est−ce qui vous garantit que vous dialoguez bien avec la bonne personne ou le bon site web ? En principe, quelqu'un aura fait l'effort (s'il est sérieux) de vérifier que les propriétaires du site sont bien ceux qu'ils affirment être. On fait naturellement confiance à ce quelqu'un : son certificat, un certificat racine, est incorporé à votre navigateur. Un certificat contient des informations relatives à son propriétaire comme une adresse électronique, un nom, l'usage prévu du certificat, une durée de validité, un identifiant de localisation ou un nom qualifié (*DN/Distinguished Name*) qui comprend le nom d'usage (*CN/Common Name*) et l'identifiant du signataire du certificat. Le certificat comprend également la clef publique et un haché pour garantir l'intégrité du message. Comme vous avez décidé de faire confiance au signataire du certificat, vous accordez du crédit à son contenu. On parle d'arbre de confiance de certification ou de chemin de certification. En général, le navigateur et les applications incluent d'office le certificat racine d'autorités de certification bien connues. L'autorité de certification tient à jour une liste des certificats signés ainsi qu'une liste des certificats révoqués. Un certificat n'est pas fiable tant qu'il n'est pas signé puisque seul un certificat signé ne peut pas être modifié. Un certificat peut se signer lui−même. On parle alors de certificat auto−signé. Tous les certificats de CA racines sont ce type.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=FJ, ST=Fiji, L=Suva, O=SOPAC, OU=ICT, CN=SOPAC Root CA/Email=administrator@sopac.org
        Validity
            Not Before: Nov 20 05:47:44 2001 GMT
            Not After : Nov 20 05:47:44 2002 GMT
        Subject: C=FJ, ST=Fiji, L=Suva, O=SOPAC, OU=ICT, CN=www.sopac.org/Email=administrator@sopac.org
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:ba:54:2c:ab:88:74:aa:6b:35:a5:a9:c1:d0:5a:
                    9b:fb:6b:b5:71:bc:ef:d3:ab:15:cc:5b:75:73:36:
                    b8:01:d1:59:3f:c1:88:c0:33:91:04:f1:bf:1a:b4:
```

```
                    7a:c8:39:c2:89:1f:87:0f:91:19:81:09:46:0c:86:
                    08:d8:75:c4:6f:5a:98:4a:f9:f8:f7:38:24:fc:bd:
                    94:24:37:ab:f1:1c:d8:91:ee:fb:1b:9f:88:ba:25:
                    da:f6:21:7f:04:32:35:17:3d:36:1c:fb:b7:32:9e:
                    42:af:77:b6:25:1c:59:69:af:be:00:a1:f8:b0:1a:
                    6c:14:e2:ae:62:e7:6b:30:e9
                Exponent: 65537 (0x10001)
         X509v3 extensions:
             X509v3 Basic Constraints:
                 CA:FALSE
             Netscape Comment:
                 OpenSSL Generated Certificate
             X509v3 Subject Key Identifier:
                 FE:04:46:ED:A0:15:BE:C1:4B:59:03:F8:2D:0D:ED:2A:E0:ED:F9:2F
             X509v3 Authority Key Identifier:
                 keyid:E6:12:7C:3D:A1:02:E5:BA:1F:DA:9E:37:BE:E3:45:3E:9B:AE:E5:A6

                 DirName:/C=FJ/ST=Fiji/L=Suva/O=SOPAC/OU=ICT/CN=SOPAC Root CA/Email=administrator@sopac.org
                 serial:00

    Signature Algorithm: md5WithRSAEncryption
        34:8d:fb:65:0b:85:5b:e2:44:09:f0:55:31:3b:29:2b:f4:fd:
        aa:5f:db:b8:11:1a:c6:ab:33:67:59:c1:04:de:34:df:08:57:
        2e:c6:60:dc:f7:d4:e2:f1:73:97:57:23:50:02:63:fc:78:96:
        34:b3:ca:c4:1b:c5:4c:c8:16:69:bb:9c:4a:7e:00:19:48:62:
        e2:51:ab:3a:fa:fd:88:cd:e0:9d:ef:67:50:da:fe:4b:13:c5:
        0c:8c:fc:ad:6e:b5:ee:40:e3:fd:34:10:9f:ad:34:bd:db:06:
        ed:09:3d:f2:a6:81:22:63:16:dc:ae:33:0c:70:fd:0a:6c:af:
        bc:5a
-----BEGIN CERTIFICATE-----
MIIDoTCCAwqgAwIBAgIBATANBgkqhkiG9w0BAQQFADCBiTELMAkGA1UEBhMCRkox
DTALBgNVBAgTBEZpamkxDTALBgNVBAcTBFN1dmExDjAMBgNVBAoTBVNPUEFDMQww
CgYDVQQLEwNJQ1QxFjAUBgNVBAMTDVNPUEFDIFJvb3QgQ0ExJjAkBgkqhkiG9w0B
CQEWF2FkbWluaXN0cmF0b3JAc29wYWMub3JnMB4XDTAxMTEyMDA1NDc0NFoXDTAy
MTEyMDA1NDc0NFowgYkxCzAJBgNVBAYTAkZKMQ0wCwYDVQQIEwRGaWppMQ0wCwYD
VQQHEwRTdXZhMQ4wDAYDVQQKEwVTT1BBQzEMMAoGA1UECxMDSUNUMRYwFAYDVQQD
Ew13d3cuc29wYWMub3JnMSYwJAYJKoZIhvcNAQkBFhdhZG1pbmlzdHJhdG9yQHNv
cGFjLm9yZzCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAulQsq4h0qms1panB
0Fqb+2u1cbzv06sVzFt1cza4AdFZP8GIwDORBPG/GrR6yDnCiR+HD5EZgQlGDIYI
2HXEb1qYSvn49zgk/L2UJDer8RzYke77G5+IuiXa9iF/BDI1Fz02HPu3Mp5Cr3e2
JRxZaa++AKH4sBpsFOKuYudrMOkCAwEAAaOCARUwggERMAkGA1UdEwQCMAAwLAYJ
YIZIAYb4QgENBB8WHU9wZW5TU0wgR2VuZXJhdGVkIENlcnRpZmljYXRlMB0GA1Ud
DgQWBBT+BEbtoBW+wUtZA/gtDe0q4O35LzCBtgYDVR0jBIGuMIGrgBTmEnw9oQLl
uh/anje+40U+m67lpqGBj6SBjDCBiTELMAkGA1UEBhMCRkoxDTALBgNVBAgTBEZp
amkxDTALBgNVBAcTBFN1dmExDjAMBgNVBAoTBVNPUEFDMQwwCgYDVQQLEwNJQ1Qx
FjAUBgNVBAMTDVNPUEFDIFJvb3QgQ0ExJjAkBgkqhkiG9w0BCQEWF2FkbWluaXN0
cmF0b3JAc29wYWMub3JnggEAMA0GCSqGSIb3DQEBBAUAA4GBADSN+2ULhVviRAnw
VTE7KSv0/apf27gRGsarM2dZwQTeNN8IVy7GYNz31OLxc5dXI1ACY/x4ljSzysQb
xUzIFmm7nEp+ABlIYuJRqzr6/YjN4J3vZ1Da/ksTxQyM/K1ute5A4/00EJ+tNL3b
Bu0JPfKmgSJjFtyuMwxw/Qpsr7xa
-----END CERTIFICATE-----
```

Comme vous avez pu le remarquer, le certificat mentionne son émetteur, contient la clef publique de son propriétaire, la période de validité du certificat et une signature pour vérifier qu'il n'a pas été modifié. Le certificat ne contient PAS la clef privée qui ne doit jamais être révélée. Ce certificat contient tous les éléments nécessaires pour envoyer un message chiffré à son propriétaire ou pour vérifier un message dont la signature lui est attribuée.

1.2.3. La clef symétrique

Les algorithmes de chiffrement à paire de clefs publique/privée sont intéressants mais pas toujours pratiques. Ils sont asymétriques parce que des clefs différentes sont requises au chiffrement et au déchiffrement. La même clef ne peut remplir les deux rôles. Les algorithmes symétriques actuels sont nettement plus rapides que les algorithmes asymétriques. Une clef symétrique est toutefois potentiellement peu sûre. Il suffit qu'un individu nuisible se la procure et il n'y a plus d'information protégée. La clef symétrique doit donc être transmise au correspondant sans être interceptée. Rien n'est sûr dans l'Internet. La solution consiste à envelopper la clef symétrique dans un message chiffré au moyen d'un algorithme asymétrique. Comme personne d'autre que vous ne connaît la clef privée, le message chiffré avec la clef privée est sûr (enfin, relativement, rien n'est garanti en ce bas monde sinon la mort et les impôts). La clef de chiffrement symétrique est choisie aléatoirement de telle sorte qu'une compromission accidentelle n'ait pas d'impact sur les transactions futures.

```
Clef symétrique-->[Clef publique]-->Clef de session chiffrée-->[Clef privée]-->Clef symétrique
```

1.2.4. Les algorithmes de chiffrement

Différents algorithmes existent, symétriques ou non, avec diverses longueurs de clef. En principe les algorithmes ne peuvent pas être brevetés. Si Henri Poincaré l'avait fait, il aurait pu poursuivre Albert Einstein & Les algorithmes ne peuvent donc pas être brevetés, à l'exception toutefois des États−Unis. OpenSSL est mis au point dans des pays où les algorithmes ne sont pas brevetés et où l'usage de la cryptographie n'est pas réservé à des agences d'état pour des fins militaires ou d'espionnage. Lors de la négociation entre le navigateur et le serveur web, les applications fournissent l'une à l'autre une liste d'algorithmes qu'elles peuvent traiter, classés par ordre de préférence. L'algorithme préféré en commun est choisi. OpenSSL peut être compilé sans inclure la gestion de certains algorithmes de façon à rester utilisable dans des pays où l'usage de la cryptographie est réglementé.

1.2.5. Le hachage

Un hachage s'obtient par application d'une fonction, dite à sens unique, à un message. La fonction présente cette propriété qu'il n'est pas possible de former un message initial à partir du haché. En outre, le haché est complètement modifié en cas de changement, même mineur, du message d'origine. Il est donc pratiquement impossible de modifier un message à haché constant. Les fonctions de hachage s'emploient dans des mécanismes à base de mot de passe, des vérifications d'intégrité (MD5) et en général pour vérifier que des données n'ont pas été altérées. L'IETF (*Internet Engineering Task Force*) préfère apparemment SHA1 à MD5 pour diverses raisons techniques (cf RFC2459 7.1.2 and 7.1.3).

1.2.6. La signature

La signature d'un message consiste à associer une identité à son contenu. Le plus souvent il s'agit de prouver qui en est l'auteur mais ce n'est pas toujours le cas. Le message peut être textuel ou être le certificat de quelqu'un d'autre. Pour signer un message, son haché est d'abord créé puis ce dernier est chiffré avec la clef privée. Le résultat et votre certificat accompagnent ensuite le message d'origine. Le destinataire de l'ensemble recalcule le haché et le compare au résultat du déchiffrement de la signature avec votre clef publique supposée. Il vérifie ensuite la validité de votre certificat.

Ce type de signature permet de transmettre la clef publique et le certificat à chaque correspondant.

Il y a deux façons de signer, soit en incluant le message à la signature, soit en chiffrant le message et sa signature. Cette dernière forme n'est qu'un chiffrement très pauvre car n'importe quelle application qui dispose de la clef publique peut l'inverser. La première forme permet de passer un contenu textuel à l'utilisateur en toute circonstance tandis que la deuxième l'interdit dès que le message a été altéré.

1.2.7. Le mot de passe

Le mot de passe correspond à un mot de passe originel Unix à ceci près qu'il est normalement plus long que 8 caractères et ne se limite pas forcément à un seul mot. De nos jours les systèmes Unix utilisent des hachés MD5 ou autres qui n'imposent plus de limitation de longueur de mot de passe.

1.2.8. L'infrastructure de clefs publiques

L'infrastructure de clefs publiques (PKI/Public Key Infrastructure) désigne le système de gestion et la base de données qui permettent de signer les certificats, de tenir à jour une liste de certificats révoquer, de distribuer les clefs publiques, etc. Il est le plus souvent accessible via un site web ou un serveur LDAP. Certaines entités sont également chargées de vérifier que vous êtes bien qui vous prétendez être. On peut avoir recours à une PKI commerciale connue dont le certificat racine se trouve déjà dans votre application. Un problème apparaît au niveau du courrier électronique pour lequel vous avez le choix entre l'emploi d'un certificat générique ou une facturation d'environ 100E par an pour chaque adresse électronique. De plus, il n'y a aucun moyen d'obtenir la clef publique de quelqu'un sans avoir reçu auparavant de ce correspondant son certificat (accompagné de sa clef publique).

1.3. Qu'en est−il de S/MIME et des autres protocoles ?

Bien que SSL ait été mis au point pour les services web, il peut s'appliquer au chiffrement de n'importe quel protocole. C'est le cas d'IMAPS, de POPS, de SMTPS & Ces protocoles dupliquent sur un port alternatif leur service originel non−sécurisé. SSL peut également chiffrer n'importe quelle transaction sans qu'elle ait besoin d'être en ligne. S/MIME est construit de cette façon en insérant un message chiffré dans un courrier électronique usuel. Le message est chiffré avec la clef publique du destinataire. Si vous n'êtes pas en contact avec votre correspondant, vous devez connaître sa clef publique, soit que vous l'ayez récupérée depuis son site web, depuis un annuaire ou qu'il vous l'ait envoyée au préalable par courrier électronique accompagnée de son certificat (pour s'assurer qu'il s'agit bien de la personne souhaitée).

Dans l'autre sens, un navigateur peut transmettre son propre certificat au serveur web afin de se faire identifier.

Chapitre 2. Gestion des certificats

2.1. Installation

De nos jours l'installation d'OpenSSL ne soulève guère de difficultés. Les distributions incluent des gestionnaires de paquets. Reportez−vous à la documentation de votre distribution ou aux fichiers README et INSTALL qu'inclut l'archive tar d'OpenSSL. Ce guide pratique n'a pas pour objectif de traiter de l'installation mais de l'utilisation.

Je décris quelques options d'installation standard dont la connaissance est nécessaire pour les exemples qui suivent. Votre installation peut différer.

Les certificats OpenSSL sont stockés dans /var/ssl. Toutes les commandes et répertoires de ce document partent de /var/ssl. Ce n'est pas indispensable mais les exemples en sont facilité.

OpenSSL cherche par défaut son fichier de configuration dans /usr/lib/ssl/openssl.cnf. Ajoutez donc systématiquement −config /etc/openssl.cnf aux commandes telles openssl ca et openssl req (par exemple). Je me sers de /etc/openssl.cnf pour maintenir l'ensemble des fichiers de configuration dans /etc.

Les utilitaires et diverses bibliothèques se trouvent dans /usr/lib/ssl

2.1.1. L'utilitaire CA.pl

Vérifiez que CA.pl se trouve dans un répertoire qui figure dans le chemin d'accès par défaut aux programmes. Il peut se trouver dans le répertoire /usr/lib/ssl. CA.pl permet de masquer la complexité des commandes openssl. Je l'utilise dans tous les exemples en indiquant entre crochets l'équivalent openssl.

Modifiez CA.pl de façon à ce que les appels à openssl ca et openssl req incluent l'option −config /etc/openssl.cnf.

```
#$SSLEAY_CONFIG=$ENV{"SSLEAY_CONFIG"};
$SSLEAY_CONFIG="-config /etc/openssl.cnf";
#$CATOP="./demoCA";
$CATOP="/var/ssl";
```

2.1.2. Le fichier openssl.cnf

/etc/openssl.cnf doit être configuré de façon à minimiser les données à renseigner à chaque invocation des utilitaires.

```
#---Begin---
#
# Fichier de configuration pour OpenSSL.
# S'emploie surtout pour les demandes de certificats.
#
# NdT: autre chose que de l'ASCII dans les fichiers de
# configuration me rend nerveux. Il y a donc un zest de triche
# dans ce qui suit. Vous ne voudriez pas me rendre nerveux, non ? :o)
#

RANDFILE  = $ENV::HOME/.rnd
oid_file  = $ENV::HOME/.oid
oid_section  = new_oids

# Section des extension X.509v3 pour se servir du
# fichier avec l'option -extfile de la commande
# "openssl x509"
# extensions  =
# (Variante: employer un fichier de configuration qui
# n'a que des extensions X.509v3 dans sa section
# principale [= default])

[ new_oids ]

# On ajoute des OID pour les commandes 'ca' et 'req'.
# Par exemple:
# testoid1=1.2.3.4
# L'emploi de substitutions est possible:
# testoid2=${testoid1}.5.6

####################################################################
[ ca ]
default_ca = CA_default  # Section de la CA standard

####################################################################

[ CA_default ]
dir             = /var/ssl                # Emplacement de base
certs           = $dir/certs              # Emplacement de stockage des nouveaux certificats
crl_dir         = $dir/crl                # Emplacement des nouvelles CRL
database        = $dir/index.txt          # Fichier d'index
```

```
new_certs_dir   = $dir/newcerts           # Emplacement des nouveaux certificats

certificate     = $dir/cacert.pem         # Certificat de la CA
serial          = $dir/serial             # Numero de serie en cours
crl             = $dir/crl.pem            # CRL en cours
private_key     = $dir/private/cakey.pem  # Clef privee
RANDFILE        = $dir/private/.rand      # Fichier d'alea
x509_extensions = usr_cert                # Extensions pour les certificats

# Extensions pour la CRL. Remarque: Netscape communicator n'aime pas les CRL de version 2,
# on commente donc pour avoir une CRL version 1
# crl_extensions = crl_ext

default_days    = 365                     # Duree de vie d'un certificat
default_crl_days= 7                       # Intervalle entre CRLs
default_md      = sha1                    # Algorithme de hachage
preserve        = no                      # Conserve-t-on l'ordre du DN ?

# Diverses facons de specifier l'allure des requetes
# Pour une requete de type CA, les attributs doivent etre les memes
# Les champs 'optional' et 'supplied' correspondent respectivement
# a des champs optionnels et fournis :o)
policy  = policy_match

# Typage CA
[ policy_match ]
countryName            = match
stateOrProvinceName    = optional
localityName           = match
organizationName       = match
organizationalUnitName = optional
commonName             = supplied
emailAddress           = optional

# Typage tout-venant
# Tous les types d'objets acceptables doivent etre enumeres
[ policy_anything ]
countryName            = optional
stateOrProvinceName    = optional
localityName           = optional
organizationName       = optional
organizationalUnitName = optional
commonName             = supplied
emailAddress           = optional

####################################################################
[ req ]
default_bits       = 1024
default_keyfile    = privkey.pem
distinguished_name = req_distinguished_name
attributes         = req_attributes
default_md         = sha1
x509_extensions    = v3_ca # Extensions pour un certificat auto-signant

# Mot de passe pour les clefs privees (l'application le demande s'il est vide).
# input_password = secret
# output_password = secret

# Masque pour les types de chaines valides. Plusieurs choix sont possibles.
# default: PrintableString, T61String, BMPString.
# pkix : PrintableString, BMPString.
# utf8only: only UTF8Strings.
# nombstr : PrintableString, T61String (pas de BMPStrings ni de UTF8Strings).
# MASK:XXXX valeur litterale.
# Attention: certaines versions de Netscape plantent sur les BMPStrings ou les UTF8Strings.
```

```
# A utiliser avec prudence!
string_mask = nombstr

# req_extensions = v3_req # Extensions pour une demande de certificat

[ req_distinguished_name ]
countryName         = Nom du pays (code sur 2 lettres)
countryName_default = FJ
countryName_min     = 2
countryName_max     = 2

stateOrProvinceName         = Etat ou province (nom complet)
stateOrProvinceName_default = Fiji

localityName          = Ville
localityName_default  = Suva

0.organizationName         = Organisation (nom de l'entreprise par exemple)
0.organizationName_default = SOPAC

# Possible mais pas normalement pas necessaire :-)
#1.organizationName         = Second nom de l'organization
#1.organizationName_default = World Wide Web SARL

organizationalUnitName         = Nom du departement dans l'organisation
organizationalUnitName_default = ITU

commonName       = Nom d'usage
commonName_max   = 64
emailAddress     = Addresse e-mail
emailAddress_max = 40

# SET-ex3   = SET extension number 3

[ req_attributes ]
challengePassword     = Un mot de passe de challenge
challengePassword_min = 4
challengePassword_max = 20

unstructuredName      = Nom optionnel

[ usr_cert ]

# Extensions ajoutees quand 'ca' signe une requete.
# Contraire aux suggestions PKIX mais des CA le font et certains
# logiciels le demandent pour ne pas confondre un certificat
# utilisateur avec un certificat de CA

basicConstraints=CA:FALSE

# Examples d'utilisation de nsCertType. En cas d'omission,
# le certificat peut servir a tout sauf a signer des objets

# Serveur SSL
# nsCertType   = server

# Certificat promis a signer des objets.
# nsCertType = objsign

# Client normal.
# nsCertType = client, email

# Tout.
# nsCertType = client, email, objsign
```

```
# Classique pour un certificat client.
# keyUsage = nonRepudiation, digitalSignature, keyEncipherment

# Pour la boiboite de commentaire de Netscape.
nsComment  = "Certificate issued by https://www.sopac.org/ssl/"

# Recommandations PKIX sans effets secondaires facheux.
subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid,issuer:always

# On importe l'adresse e-mail
# pour les attributs subjectAltName et issuerAltname.
# subjectAltName=email:copy

# Informations relatives au sujet.
# issuerAltName=issuer:copy

# Adresse de base des autres URL si on n'en donne pas
# au cas par cas.
nsBaseUrl  = https://www.sopac.org/ssl/

# Adresse de la CRL du moment.
nsCaRevocationUrl = https://www.sopac.org/ssl/sopac-ca.crl

# Adresse pour revoquer un certificat.
nsRevocationUrl  = https://www.sopac.org/ssl/revocation.html?

# Adresse pour renouveller un certificat.
nsRenewalUrl  = https://www.sopac.org/ssl/renewal.html?

# Adresse des pratiques de la CA.
nsCaPolicyUrl  = https://www.sopac.org/ssl/policy.html.

# Adresse du certificat du signataire.
issuerAltName = URI:https://www.sopac.org/ssl/sopac.crt.

# Adresse de la CRL du moment.
crlDistributionPoints = URI:https://www.sopac.org/ssl/sopac-ca.crl

[ v3_ca ]

# Extensions d'une CA standard

# Recommandation PKIX

subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid:always,issuer:always

# Recommandation PKIX que certains bugware n'aiment pas
# basicConstraints = critical,CA:true
# On utilise donc plutot ceci
basicConstraints = CA:true

# Emploi de la clef: typique d'un certificat de CA.
# On le laisse inactif pour prevenir l'emploi d'un
# certificat auto-signant de test.
# keyUsage = cRLSign, keyCertSign

# En fonction des besoins.
# nsCertType = sslCA, emailCA

# On inclut l'adresse e-mail dans le nom alternatif du sujet (recommendation PKIX)
# subjectAltName=email:copy
```

```
# On copie les informations du signataire
# issuerAltName=issuer:copy

# Encodage DER en base 16 d'une extension: licence de pilotage requise!
# 1.2.3.5=RAW:02:03
# On peut surcharger une extension standard:
# basicConstraints= critical, RAW:30:03:01:01:FF

# Message pour la boite d'affichage de Netscape.
nsComment  = "Certificat en provenance de https://www.sopac.org/ssl/"

# Adresse de base des autres URL si on n'en donne pas
# au cas par cas.
nsBaseUrl  = https://www.sopac.org/ssl/

# Adresse de la CRL du moment.
nsCaRevocationUrl = https://www.sopac.org/ssl/sopac-ca.crl

# Adresse pour revoquer un certificat.
nsRevocationUrl  = https://www.sopac.org/ssl/revocation.html?

# Adresse pour renouveller un certificat.
nsRenewalUrl  = https://www.sopac.org/ssl/renewal.html?

# Adresse des pratiques de la CA.
nsCaPolicyUrl  = https://www.sopac.org/ssl/policy.html

# Adresse du certificat du signataire.
issuerAltName = URI:https://www.sopac.org/ssl/sopac.crt

# Adresse de la CRL du moment.
crlDistributionPoints = URI:https://www.sopac.org/ssl/sopac-ca.crl

[ crl_ext ]
# Extensions de CRL
# Seuls issuerAltName et authorityKeyIdentifier se justifient dans une CRL.
# issuerAltName=issuer:copy
authorityKeyIdentifier=keyid:always,issuer:always


#----End----
```

Quelques remarques au sujet du fichier openssl.cnf

- Plusieurs variables se présentent avec les suffixes _default, _min et _max pour leurs valeurs minimales ou leurs nombres minimaux et maximaux de caractères respectivement.
- Le fichier est basé sur des [Sections] de variables.

dir :
: répertoire de base.

default_ca :
: désigne la section des variables pour un certificat dont le type n'est pas spécifié.

basicConstraints :
: définit l'usage du certificat. CA:TRUE désigne un certificat de CA racine par exemple.

2.1.3. Création de l'autorité de certification

Utilisez la commande suivante après avoir modifié de façon adéquate le fichier openssl.cnf :

```
CA.pl -newca
```

L'utilitaire demande de choisir un fichier contenant le certificat de l'AC ou bien il propose d'en créer un. A titre d'exercice, laissez vous guider dans la création d'une AC. La partie suivante remplace cette AC par une AC de durée de vie plus importante. CA.pl ne génère que des certificats valables 365 jours.

2.2. Création d'une autorité de certification racine

La commande suivante crée un certificat auto−signé (pour une autorité de certification donc).

```
CA.pl -newcert
openssl req -config /etc/openssl.cnf -new -x509 -keyout newreq.pem \
        -out newreq.pem -days 365
```

Le nouveau certificat se trouve dans le fichier newreq.pem. Utilisez un CN du type « ACME root Certificate ». Le fichier est à diviser en deux parties, cacert.pem et private/cakey.pem. La partie délimitée par −RSA PRIVATE KEY− va dans le fichier private/cakey.pem tandis que la partie −CERTIFICATE− est destinée au fichier cacert.pem. Supprimez newreq.pem une fois l'opération effectuée.

Vérifiez à présent que le fichier index.txt est vide et que le fichier serial contient la valeur 01.

Afin de ne pas avoir à renouveler tous les certificats émis par une AC lorsque son certificat racine expire, il est souhaitable d'augmenter la durée de vie de ce dernier. Les autorités de certification commerciales emploient des durées de 5 à 10 ans.

```
openssl req -config /etc/openssl.cnf -new -x509 -keyout private/cakey.pem \
       -out cacert.pem -days 3650
```

La commande précédente est plus efficace que « CA.pl −newcert » car elle place les fichiers créés à l'emplacement souhaité et génère une AC racine valable 10 ans.

Il reste à s'assurer que le certificat auto−signé racine n'est employé que pour signer des certificats. La confidentialité de la clef privée est très importante. Ne la compromettez jamais en ôtant le mot de passe. On peut envisager de la ranger sur un support amovible et de n'y accéder qu'en cas de besoin. Si la machine est compromise, la clef privée est ailleurs.

On dispose maintenant d'une autorité de certification racine. Les utilisateurs doivent faire confiance à votre certificat de CA et donc le récupérer et le charger dans leurs navigateurs.

Il faudra renseigner le mot de passe lors de la signature de chaque certificat.

2.3. Création d'une autorité de certification non−racine

CORRIGEZ−MOI parce que je ne suis pas tout à fait sûr de la procédure.

Un certificat peut être employé pour en signer un autre pourvu qu'il soit valide et ait été autorisé à le faire lors de sa création. On peut donc créer une demande de certificat et une clef privée, faire signer le certificat par un tiers et installer le certificat signé et la clef privée. La partie −PRIVATE KEY− va dans le fichier private/cakey.pem tandis que la partie −CERTIFICATE− va dans cacert.pem.

2.4. Mise en place du certificat de CA racine comme certificat fiable.

On commence par ôter du certificat tout ce qui n'appartient pas à la section −CERTIFICATE−.

```
openssl x509 -in cacert.pem -out cacert.crt
```

On place ensuite le fichier obtenu sur son site web, par exemple http://mysite.com/ssl/cacert.crt. Le serveur web doit intégrer une description MIME pour les fichiers .crt. Le certificat est prêt à être téléchargé et utilisé par n'importe quel navigateur.

Il est important de publier son certificat racine de CA sur un site web car il est peu probable que les utilisateurs l'aient déjà d'inclus dans leur navigateur. Notez que n'importe qui peut imiter votre site web et votre certificat de CA racine. Dans la mesure où on dispose de plusieurs chemins pour transmettre l'information, il devient fortement improbable qu'un attaquant arrive à tout contrôler.

Microsoft propose un service de mise à jour de Windows qui propagera votre certificat racine aux programmes Internet Explorer en circulation. Contactez Microsoft pour que votre certificat soit ajouté à leur base de données et intégré à leurs futurs paquets.

2.4.1. Netscape/Mozilla

On récupère le certificat depuis le web ou le système de fichiers avec Netscape. Ce dernier identifie automatiquement le fichier comme un certificat racine et propose de l'intégrer. Un assistant guide dans l'installation. A la fin du processus, on précise le domaine d'application du certificat : sécurisation web, signature de courrier électronique ou signature de code.

2.4.2. Galeon

Galeon fonctionne comme Mozilla en ce qu'il repose sur le même moteur de rendu HTML. Il n'inclut toutefois pas d'outil de gestion des certificats.

2.4.3. Opera

COMPLÉTEZ–MOI

2.4.4. Internet Explorer

On commence par aller à l'adresse de téléchargement du certificat avant de le sauvegarder en local. L'assistant d'installation de certificat est invoqué par double–clic sur le fichier de certificat. Internet Explorer installe spontanément les certificats auto–signés dans la liste des autorités de certification. A partir de ce point, Internet Explorer cesse d'émettre des avertissements et considère comme fiables les certificats signés avec ce certificat de CA racine.

Il est également possible d'ouvrir le certificat depuis Internet Explorer qui en affiche alors le contenu. L'assistant démarre ensuite en appuyant sur le bouton d'installation du certificat.

2.5. Gestion des certificats

2.5.1. Création et signature de demandes de certificats

```
CA.pl -newreq
(openssl req -config /etc/openssl.cnf -new -keyout newreq.pem \
         -out newreq.pem -days 365)
```

La commande précédente crée une nouvelle clef privée et une demande de certificat qu'elle place dans newreq.pem. Pour sécuriser un site web, par exemple www.sopac.org, on utilise le nom d'usage (CN) www.sopac.org. Pour le courrier électronique de franck@sopac.org, on utiliserait franck@sopac.org.

```
CA.pl -sign
(openssl ca -config /etc/openssl.cnf -policy policy_anything \
        -out newcert.pem -infiles newreq.pem)
```

Cette commande signe la demande avec cacert.pem et sauve le certificat dans newcert.pem. Le mot de passe de cacert.pem (le certificat de CA) est nécessaire. Un fichier newcerts/xx.pem est créé et les fichiers index.txt, serial sont mis à jour.

La clef privée se trouve donc dans la section –PRIVATE KEY– de newreq.pem et le certificat dans la section –CERTIFICATE– de newcert.pem.

Une copie de newcert.pem est émise dans le répertoire newcerts/ et l'enregistrement correspondant ajouté à index.txt de telle sorte qu'un client puisse accéder à l'information via un serveur web et vérifier l'authenticité du certificat.

On note que le fichier newreq.pem contient aussi bien la demande de certificat que la clef privée. La section –PRIVATE KEY– n'est pas nécessaire à la signature et il faut absolument la retirer avant de transmettre la demande à quelqu'un d'autre pour qu'il la signe. De même, pour signer une demande de certificat, il suffit d'obtenir la section –CERTIFICATE REQUEST–. Il n'y a aucune raison de demander la clef privée.

2.5.2. Révocation de certificat

La commande suivante révoque un certificat :

```
openssl -revoke newcert.pem
```

La base de données index.txt est modifiée en conséquence et le certificat est annoté comme révoqué. Il reste à régénérer la liste de révocation des certificats :

```
openssl ca -gencrl -config /etc/openssl.cnf -out crl/sopac-ca.crl
```

La liste de révocation (CRL/Certificate Revokation List) doit être rendue publique, par exemple sur un site web.

Les paramètres crldays, crlhours permettent de préciser la prochaine date de mise à jour de la CRL. crlexts spécifie la section du fichier openssl.cnf à employer pour créer une CRL de version 2 en place d'une CRL de version 1.

```
openssl ca -gencrl -config /etc/openssl.cnf -crldays 7 \
          -crlexts crl_ext -out crl/sopac-ca.crl
```

2.5.3. Renouvellement d'un certificat

L'utilisateur transmet son ancienne demande de certificat ou en régénère une nouvelle basée sur sa clef privée.

Il faut alors révoquer l'ancien certificat et signer la nouvelle demande de certificat.

L'ancien certificat se retrouve en cherchant dans le fichier index.txt le nom qualifié (DN) qui correspond à la requête. La procédure de révocation s'effectue ensuite avec le numéro de série <xx> et le fichier de certificat cert/<xx>.pem.

La signature de la nouvelle requête peut être manuelle pour s'assurer que les dates de validité du certificat seront correctes.

```
openssl ca -config /etc/openssl.cnf -policy policy_anything -out newcert.pem \
-infiles newreq.pem -startdate [now] -enddate [previous enddate+365days]
```

[now] et [previous enddate+365days] sont à remplacer par des valeurs adéquates.

2.5.4. Affichage d'un certificat

L'affichage sous forme textuelle du contenu d'un certificat sous forme codée s'effectue avec la commande suivante :

```
openssl x509 -in newcert.pem -noout -text
```

2.5.5. Le fichier index.txt

Le fichier index.txt contient les références des certificats créés par OpenSSL. Les enregistrements sont annotés avec un R pour indiquer que le certificat est révoqué, V qu'il est valide et E qu'il a expiré.

2.5.6. Élaborer son service web d'autorité de certification

Être une autorité de certification implique certaines tâches :

1. publier le certificat de CA racine pour qu'il puisse être déployé dans les applications ;
2. publier la liste de révocation ;
3. afficher le contenu détaillé des certificats, notamment leur numéro de série ;
4. fournir un formulaire de demande de certificats.

Un serveur web et quelques scripts permettent de réaliser toutes ces fonctions.

COMPLÉTEZ–MOI : code de l'interface web.

Chapitre 3. Emploi des certificats avec les applications

3.1. Sécurisation des protocoles Internet

3.1.1. mod_ssl (Apache) et les certificats

Il ne faut jamais se servir du certificat auto–signé de la CA racine avec quelque application que ce soit, notamment avec Apache qui oblige à supprimer le mot de passe de protection de la clef privée.

On commence par créer et signer une demande de certificat dont le nom d'usage (CN) est du type www.mysite.com. On ne conserve que la section –––CERTIFICATE ––– du certificat.

La clef doit être déverrouillée en en autorisant l'usage sans devoir rentrer le mot de passe. On supprime donc le mot de passe du fichier newreq.pem :

```
openssl rsa -in newreq.pem -out wwwkeyunsecure.pem
```

Comme la clef privée n'est plus protégée, on a intérêt à savoir ce qu'on fait et notamment à vérifier les droits d'accès aux fichiers. Si quelqu'un arrive à s'en emparer, le site est compromis. On peut à présent utiliser newcert et wwwkeyunsecure.pem avec Apache.

On copie wwwkeyunsecure.pem et newcert.pem dans le répertoire /etc/httpd/conf/ssl/ dans les fichiers wwwkeyunsecure.pem et wwwcert.crt respectivement.

On édite ensuite /etc/httpd/conf/ssl/ssl.default−vhost.conf :

```
----
# Server Certificate:
# SSLCertificateFile doit pointer vers un certificat en PEM.
# Si on verrouille le certificat, un mot de passe est requis.
# kill -HUP demande de nouveau le certificat. On cree un
# certificat de test via `make certificate' lors de la
# compilation.
#SSLCertificateFile conf/ssl/ca.crt
SSLCertificateFile wwwcert.crt
# Clef secrete du serveur:
# A employer lorsque la clef est separee du certificat.
#SSLCertificateKeyFile conf/ssl/ca.key.unsecure
SSLCertificateKeyFile wwwkeyunsecure.pem
----
```

On arrête httpd (/etc/rc.d/init.d/httpd stop) puis on vérifie que tous les processus ont disparu (killall httpd) avant de relancer le démon (/etc/rc.d/init.d/httpd start).

3.1.2. IMAPS et les certificats

Se reporter au paragraphe « POPS et les certificats » pour davantage de détails.

3.1.3. POPS et les certificats

Un fichier pem pour ipop3sd se crée en générant un certificat, en déverrouillant la clef privée et et combinant les deux dans le fichier /etc/ssl/imap/ipop3sd.pem. Ce dernier fichier correspond à l'emplacement attendu par le rpm de la Mandrake 9.0. La même procédure s'applique à imap avec le fichier /etc/ssl/imap/imapsd.pem.

Le CN doit correspondre au nom auquel le client de messagerie se connecte (par exemple mail.xyz.org). Avec MS−Outlook, le nom du serveur pop se rentre dans l'onglet à cet effet et on active l'option « This server requires a secure connection (SSL) » dans les propriétés avancées. Les connexions s'effectuent alors sur le port 995 (imaps). Le certificat de la CA doit être installé dans MS Internet Explorer pour que le certificat de mail.xyz.org soit validé.

3.1.4. Postfix et les certificats

COMPLÉTEZ−MOI

3.1.5. Stunnel et les certificats

COMPLÉTEZ−MOI

3.1.6. Création et signature de clef avec Microsoft Key Manager

Dans Microsoft Key Manager, on choisit le service pour lequel on souhaite créer un certificat, par exemple IMAP (ou WWW). L'assistant permet de générer une nouvelle clef. Il faut s'assurer de ce que le nom qualifié n'est pas identique à celui d'une clef existante. Un nom d'usage (CN) tel imap.mycompany.com fait l'affaire. L'assistant sauve la demande dans le fichier C:\NewKeyRq.txt. Key Manager indique la clef avec un signe qui signale qu'elle n'est pas signée.

On copie ce fichier dans le répertoire /var/ssl d'OpenSSL sous le nom newreq.pem et on signe la requête comme à l'accoutumé.

```
CA.pl -sign
```

newcert.pem n'est pas intelligible par Key Manager car il contient du texte et une section −CERTIFICATE−. Il faut supprimer le texte :

```
openssl x509 -in newcert.pem -out newcertx509.pem
```

Un éditeur remplit également cette tâche.

newcertx509.pem ne contient plus qu'une section −CERTIFICATE−.

On recopie newcertx509.pem sur la machine où s'exécute Key Manager. Il suffit alors de cliquer du bouton droit dessus dans l'application Key Manager et de renseigner le mot de passe. La clef est alors fonctionnelle.

3.2. Sécurisation du courrier électronique

3.2.1. Création et emploi d'un certificat S/MIME

Il faut simplement générer et signer une demande de certificat avec l'adresse électronique en guise de nom d'usage (CN).

On peut alors signer un message de test test.txt avec le nouveau certificat newcert.pem et la clef newreq.pem :

```
openssl smime -sign -in test.txt -text -out test.msg -signer newcert.pem -inkey newreq.pem
```

test.msg peut être transmis à n'importe qui et cette procédure s'applique pour émettre des notes signées ou tout autre document signé destiné à être publié électroniquement.

3.2.2. Emploi du certificat S/MIME avec MS Outlook

Il faut l'importer dans Outlook sous forme de fichier pkcs12. La création du fichier pkcs12 s'effectue comme suit :

```
CA.pl -pkcs12 "Franck Martin"
(openssl pkcs12 -export -in newcert.pem -inkey newreq.pem \
         -out newcert.p12 -name "Franck Martin")
```

Il est possible de lier le certificat de signature au fichier pkcs12 :

```
openssl pkcs12 -export -in newcert.pem -inkey newreq.pem \
        -certfile cacert.pem -out newcert.p12 -name "Franck Martin"
```

Ce certificat contient les clefs privées et publique. Seul le mot de passe protège cette dernière. Le fichier ne doit donc pas traîner n'importe où.

On suit les menus Tools, Options et Security de MS Outlook. Un clic sur le bouton import/export active l'import du fichier pkcs12. Il n'y a plus qu'à renseigner le mot de passe d'export et l'identité de l'utilisateur, "Franck Martin" dans mon cas (à adapter).

On clique ensuite sur le bouton Settings puis, MS Outlook ayant normalement sélectionné les choix par défaut, sur New. Il est alors possible d'envoyer des courriers électroniques signés. La clef publique est transmise en même temps que les messages signés et le destinataire est donc en mesure de renvoyer des données chiffrées.

Comme le certificat a été généré à partir d'un certificat auto−signé, le chemin de validation ne sera pas valide tant que l'application ne connaît pas le certificat de l'AC racine. On se reportera à la section d'installation du certificat de l'AC racine dans Internet Explorer pour la détail de la marche à suivre.

Le message est envoyé signé/chiffré ou simplement signé en clair. Le chiffrement n'en est pas vraiment un dans la mesure où le message contient tout le nécessaire pour effectuer le déchiffrement.

Certaines anciennes versions de MS Outlook pour XP parcourent Internet pour vérifier la validité du certificat. Plusieurs secondes peuvent alors s'écouler avant que le courrier ne soit affiché, voire plusieurs minutes pour que le hors−temps de MS Outlook ne se déclenche si on n'est pas connecté en permanence. Ce processus bloque toute la machine jusqu'à ce que MS Outlook l'ait achevé d'une façon ou d'une autre.

3.2.3. Emploi du certificat S/MIME avec MS Outlook Express

COMPLÉTEZ−MOI

3.2.4. Emploi du certificat S/MIME avec Netscape Messenger

COMPLÉTEZ−MOI

3.2.5. Emploi du certificat S/MIME avec Evolution

Evolution 1.0 ne gère pas S/MIME mais seulement PGP. La prise en charge S/MIME est prévue pour une version ultérieure (d'après la base de données de suivi des bogues d'Evolution). Evolution remarque quand même parfois que le document est signé en clair et l'affiche correctement bien qu'il ne puisse en vérifier la validité. Des versions antérieures d'Evolution n'assimilaient pas un des trois types MIME de signature que MS Outlook emploie hélas assez souvent.

3.2.6. Emploi du certificat S/MIME avec Balsa

COMPLÉTEZ−MOI

3.2.7. Emploi du certificat S/MIME avec KMail

COMPLÉTEZ−MOI

3.3. Protection des fichiers

3.3.1. WinCrypt

WinCrypt s'appuie sur l'API de chiffrage de Microsoft pour chiffrer et signer les fichiers. Il crée en outre de façon optionnelle une archive zip avant la signature. Il dispose d'un frontal pour gérer la liste des certificats, les parcourir, en installer, en supprimer et choisir celui à employer avec WinCrypt.

La procédure de création d'un certificat est identique à celle de Microsoft Outlook. La même liste de certificats est utilisée par les deux applications.

Un fichier filename.sgn signé avec WinCrypt se vérifie avec la commande :

```
openssl smime -verify -inform der -in filename.sgn -CAfile cacert.crt
```

La commande suivante permet de signer un fichier avec OpenSSL de façon compatible :

```
openssl smime -sign -outform der -nodetach -out filename.sgn \
-signer certificate.pem -in filename.txt
```

Pour visualiser l'organisation d'un fichier signé :

```
openssl asn1parse -inform der -in filename.sgn
```

3.4. Sécurisation des programmes

3.4.1. Micosoft Code

Il est possible de signer ses programmes et ses applets pour prouver qu'on en est l'auteur. Il est important pour les clients de savoir que personne n'a modifié le code en y insérant un virus ou un cheval de Troie. L'étape de signature nécessite le SDK Microsoft Authenticode. Ce dernier est disponible dans la section MSDN du site web de Microsoft.

On génère un certificat comme à l'accoutumé mais avec un nom d'usage de la forme « ACME Software Cert ». Le certificat doit ensuite être signé par l'autorité de certification et converti au format pkcs12.

```
CA.pl -newreq
CA.pl -sign
CA.pl -pkcs12 "ACME Software Cert"
```

On obtient un fichier newcert.p12 qu'on importe dans la liste des certificats en cliquant dessus une fois sous MS Windows.

Le certificat peut alors servir à signer du code :

```
signcode -cn "ACME Software cert" -tr 5 -tw 2 -n "My Application" \
-i http://www.acme.com/myapp/ \
-t http://timestamp.verisign.com/scripts/timstamp.dll myapp.exe
```

A l'installation et à l'exécution de l'application, une boite de dialogue apparaît sous le nom « My Application » avec le lien pointé par l'argument –i.

3.5. IPSec

IPSec est un nouveau protocole basé sur IP qui autorise les liens chiffrés à la demande entre deux machines. La mise en Suvre d'IPSec est obligatoire dans le cadre d'IPv6 et peut être incorporée dans IPv4. Qu'IPSec soit obligatoire pour IPv6 ne signifie pas pour autant qu'il est systématiquement déployé par les administrateurs réseau. IPSec n'est pas facile à mettre en Suvre à cause des mécanismes de déploiement automatiques des clefs. Le DNS peut aider mais cette méthode n'est pas encore courante. En outre, les autorités de certification ne proposent pas d'outils pour un déploiement à grande échelle en entreprise.

3.5.1. FreeS/WAN

FreeS/WAN est une souche IPSec renommée pour GNU/Linux. La version actuelle (1.9.7) doit être patchée pour gérer les certificats X.509v3. Une telle version modifiée est disponible sur ce site. Certaines distributions l'incluent en standard, il est donc conseillé de commencer par vérifier si c'est le cas pour celle qu'on utilise.

Cette version présente l'intérêt de permettre l'usage d'OpenSSL pour la création des certificats de FreeS/WAN et des enregistrements CERT du DNS. En outre, l'interaction avec la souche IPSec de Microsoft est réalisable. On se reportera à la page de Nate's pour davantage d'informations.

3.5.1.1. Passerelle FreeS/WAN

On crée un certificat dont le CN correspond au nom de domaine complètement qualifié de la passerelle IPSec, par exemple host.example.com. Ne pas oublier de signer le certificat. On dispose des deux fichiers newcert.pem et newreq.pem. Le fichier newreq.pem doit être édité pour ne plus contenir que la clef privée : on supprime tout ce qui n'est pas compris entre les balises −−BEGIN RSA PRIVATE KEY−− et −−END RSA PRIVATE KEY−−. Il faut ensuite copier ces fichiers sur la passerelle en faisant attention à ce que le transfert soit sécurisé. Les fichiers de configuration de FreeS/WAN se trouvent dans le répertoire /etc/freeswan pour ma distribution. À ajuster suivant les cas.

```
mv newreq.pem /etc/freeswan/ipsec.d/private/host.example.com.key
mv newcert.pem /etc/freeswan/ipsec.d/host.example.com.pem
```

On copie le certificat racine dans le répertoire de configuration de FreeS/WAN. Il ne faut copier que le certificat, pas la clef.

```
mv cacert.pem /etc/freeswan/ipsec.d/cacerts
```

On génère une liste de révocation ou on copie l'existante à l'emplacement adéquat :

```
openssl ca -genrcl -out /etc/freeswan/ipsec.d/crls/crl.pem
```

Toujours sur la passerelle, on inclut dans le fichier ipsec.secrets la ligne ci−dessous :

```
: RSA host.example.com.key
          "password"
```

Le mot de passe est celui qui a servi à créer la paire de clef. On configure le fichier ipsec.conf comme suit :

```
config setup
interfaces=%defaultroute
klipsdebug=none
plutodebug=none
plutoload=%search
plutostart=%search
uniqueids=yes

conn %default
keyingtries=1
compress=yes
disablearrivalcheck=no
authby=rsasig
leftrsasigkey=%cert
rightrsasigkey=%cert

conn roadwarrior-net
leftsubnet=<sous_reseau>/<masque_de_sous_reseau>
also=roadwarrior

conn roadwarrior
right=%any
left%defaultroute
leftcert=host.example.com.pem
auto=add
pfs=yes
```

Comme on peut le voir, deux liens sont établis, l'un avec la passerelle, l'autre avec le réseau situé derrière la passerelle. Ceci s'avère très pratique si un pare−feu ou un traducteur d'adresses est actif sur la passerelle. La configuration autorise tout propriétaire d'un certificat valide à se connecter à la passerelle.

3.5.1.2. Client FreeS/WAN

La procédure est semblable. Il faut créer un certificat dont le CN correspond au nom de domaine complètement qualifié de l'hôte, par exemple clienthost.example.com. Le certificat doit être signé par la même autorité de certification que la passerelle. Le lien est alors autorisé.

Comme pour la passerelle, on copie de façon sécurisée les fichiers suivants dans les répertoires de configuration :

```
mv newreq.pem /etc/freeswan/ipsec.d/private/clienthost.example.com.key
mv newcert.pem /etc/freeswan/ipsec.d/clienthost.example.com.pem
```

On copie le certificat racine dans le répertoire de configuration de FreeS/WAN. Il ne faut copier que le certificat, pas la clef.

```
mv cacert.pem /etc/freeswan/ipsec.d/cacerts
```

On génère une liste de révocation ou on copie l'existante à l'emplacement adéquat :

```
openssl ca −genrcl −out /etc/freeswan/ipsec.d/crls/crl.pem
```

Enfin, on copie le certificat (mais pas la clef privée) de la passerelle :

```
mv host.example.com.pem /etc/fresswan/ipsec.d/host.example.com.pem
```

Le fichier ipsec.secrets est édité de la même façon pour charger la clef privée :

```
: RSA clienthost.example.com.key
        "password"
```

On édite le fichier ipsec.conf pour activer la connexion :

```
config setup
interfaces=%defaultroute
klipsdebug=none
plutodebug=none
plutoload=%search
plutostart=%search
uniqueids=yes

conn %default
keyingtries=0
compress=yes
disablearrivalcheck=no
authby=rsasig
leftrsasigkey=%cert
rightrsasigkey=%cert

conn roadwarrior−net
left=(ip of host)
leftsubnet=(gateway_host_subnet)/(gateway_host_netmask)
also=roadwarrior

conn roadwarrior
```

```
left=(ip of host)
leftcert=host.example.com.pem
right=%defaultroute
rightcert=clienthost.example.com.pem
auto=add
pfs=yes
```

On met ensuite le VPN en action :

```
ipsec auto --up roadwarrior
ipsec auto --up roadwarrior-net
```

Afin que le lien s'établisse automatiquement, on remplace dans le fichier de configuration la directive « auto=add » par « auto=start ».

3.5.1.3. Client MS Windows 2000/XP

La procédure est identique à celle du client FreeS/WAN. On crée un certificat, par exemple pour winhost.example.com, qui doit à présent être converti au format pkcs12. On se reportera à la section « MS Outlook et les certificats » pour le détail de la marche à suivre. On s'assure de ce que le fichier .p12 est attaché au certificat de l'autorité de certification racine.

On note le résultat de la commande :

```
openssl x509 -in cacert.pem -noout -subject
```

Ce fichier doit être copié de façon sécurisée sur la machine MS Windows.

On installe à présent l'utilitaire ipsec.exe de Marcus Muller, par exemple dans le répertoire c:\ipsec.

On ouvre la console de gestion Microsoft (Microsoft Management Console/MMC) puis on clique sur « Add/Remove Snap–in », « Add », « Certificates », « Add », « Computer Account » et « Next ». On choisit alors « Local computer » et « Finish ». Dans « IP Security Policy Management », choisir « Add », « Local Computer » puis « Finish », « Close » et enfin « OK ».

On peut à présent ajouter le certificat .p12.

On clique sur la flèche d'ajout pour « Certificates (Local Computer) » puis avec le bouton droit sur « Personal » avant de cliquer sur « All Tasks », « Import » et « Next ». On renseigne ensuite le chemin d'accès au fichier .p12 (ou on navigue pour atteindre le fichier) et on clique sur « Next ». On rentre alors le mot de passe de protection du fichier puis on clique sur « Next ». On choisit « Automatically select the certificate store based on the type of certificate », « Next », « Finish » et on répond positivement à toutes les questions qui se présentent. On sort de la console et on enregistre la séquence correspondante dans un fichier pour ne pas avoir à se la farcir à chaque fois.

On installe ipsecpol.exe (Windows 2000) ou ipseccmd.exe (Windows XP) comme indiqué dans la documentation de l'utilitaire ipsec. On édite le fichier ipsec.conf de la machine MS Windows en remplaçant "RightCA" par la sortie de la commande « openssl x509 –in cacert.pem –noout –subject »; mise en forme comme indiqué ci dessous (il faut remplacer les / par des virgules et changer le nom de quelques champs comme indiqué dans l'exemple) :

```
conn roadwarrior
left=%any
right=(ip_du_systeme_distant)
rightca="C=FJ, ST=Fiji, L=Suva, O=SOPAC, OU=ICT, CN=SOPAC Root"
network=auto
```

```
auto=start
pfs=yes

conn roadwarrior-net
left=%any
right=(ip_du_systeme_distant)
rightsubnet=(sous_reseau)/(masque_de_sous_reseau)
rightca="C=FJ, ST=Fiji, L=Suva, O=SOPAC, OU=ICT, CN=SOPAC Root"
network=auto
auto=start
pfs=yes
```

On démarre à présent le lien.

Pour cela on invoque la commande « ipsec.exe ». Voici un exemple de sortie de cette commande :

```
C:\ipsec>ipsec
IPSec Version 2.1.4 (c) 2001,2002 Marcus Mueller
Getting running Config ...
Microsoft's Windows XP identified
Host name is: (local_hostname)
No RAS connections found.
LAN IP address: (local_ip_address)
Setting up IPSec ...

Deactivating old policy...
Removing old policy...

Connection roadwarrior:
MyTunnel : (local_ip_address)
MyNet : (local_ip_address)/255.255.255.255
PartnerTunnel: (ip_of_remote_system)
PartnerNet : (ip_of_remote_system)/255.255.255.255
CA (ID) : C=FJ, ST=Fiji, L=Suva, O=SOPAC, OU=ICT, CN=SOPAC Root...
PFS : y
Auto : start
Auth.Mode : MD5
Rekeying : 3600S/50000K
Activating policy...

Connection roadwarrior-net:
MyTunnel : (local_ip_address)
MyNet : (local_ip_address)/255.255.255.255
PartnerTunnel: (ip_of_remote_system)
PartnerNet : (remote_subnet)/(remote_netmask)
CA (ID) : C=FJ, ST=Fiji, L=Suva, O=SOPAC, OU=ICT, CN=SOPAC Root...
PFS : y
Auto : start
Auth.Mode : MD5
Rekeying : 3600S/50000K
Activating policy...
C:\ipsec>
```

On émet un ping vers la passerelle qui devrait afficher « Negotiating IP Security » pendant quelques instants avant de répondre au ping. Pour une T1 qui attaque un VPN derrière un modem cable, il s'écoule typiquement de trois à quatre pings. Une fois la même chose faite depuis le réseau interne à l'autre extrémité du lien, ce dernier devrait être opérationnel.

Chapitre 4. PKI Globale

4.1. PKIs existantes

Aujourd'hui on a le choix entre avoir recours à une PKI commerciale ou utiliser la sienne. Les PKI commerciales ont été créées à l'origine pour permettre le commerce électronqiue via Internet et plus particulièrement sur le web. Le prix des certificats dépendait du nombre d'hôtes. Le prix est plus élevé que pour un nom de domaine parce que les coûts d'identification des demandeurs sont plus élevés et parce qu'il n'y a pas de raison de se priver d'un pourcentage sur l'activité des sites marchands. Etrangement ce type de vision atteint assez vite ses limites quand il ne s'agit plus seulement de sécuriser quelques serveurs, fussent−ils POP ou IMAP, mais que chaque hôte a besoin de son certificat quand ce n'est pas chaque utilisateur ou chaque application. Le coût s'envole avec l'espoir d'un système un tant soit peu administrable.

Pourquoi ne pas voir un certificat pour signer tous les autres ? Pour l'instant la seule solution consiste à construire sa propre autorité de certification, ce qui permet une gestion souple mais limite la portée du procédé à l'organisation qui y a recours. Le reste du monde doit récupérer des clefs d'autorités de certification racine au cas par cas.

La solution consiste en une PKI unique gérée par une entité centrale de façon analogue au DNS. On parle alors de PKI globale.

4.2. Le besoin d'une PKI globale

La sécurité des ordinateurs personnels est devenu un sujet d'une telle importance aujourd'hui que Bill Gates a annoncé que Microsoft choisirait à présent la sécurité aux fonctionnalités si la décision devait être prise (NdT : il _peut_ le dire).

Le nombre de nuisibles sur l'Internet a augmenté. N'importe qui peut envoyer n'importe quoi n'importe où et essayer de faire installer diverses saloperies sur les machines des autres. Une fois tout le monde identifié, on sait au moins de qui se plaindre en cas de problème. C'est particulièrement vrai dans le cas du courrier non−sollicité pour lequel il est souvent difficile d'authentifier l'émetteur d'origine. Le traitement d'une donnée peut être différent si elle n'est pas traçable au moyen d'un certificat. Il s'agit de la même approche que celle de la présentation du numéro d'appel téléphonique. Les certificats X.509v3 l'autorisent pour toutes les applications sur Internet, pour le courrier électronique (S/MIME), les transactions commerciales (HTTPS), les installations de logiciel & Ils ne sont toutefois pas très répandus en raison de leur coût quand il faut les déployer à grande échelle. Combien d'utilisateurs de Yahoo mail, d'Hotmail, de CA Online peuvent s'offrir un certificat électronique ? Il existe des projets de certificats gratuits mais ils ne peuvent prouver que l'existence d'une adresse électronique et n'ont pas les moyens de garantir l'identité de leur propriétaire.

Une PKI Globale est nécessaire. Les protocoles et les standards existent sans qu'il y ait besoin de réinventer la roue. L'IETF dispose de l'outillage nécessaire. Un serveur LDAP peut stocker les certificats, le DNS les annoncer et S/MIME sécuriser les courriers. Il ne reste plus qu'un problème de politique : quels services doivent coopérer au sein d'une PKI globale, qui doit offrir un tel service, quel sera le niveau de sécurité offert ?

Cette partie sera tenue à jour au fur et à mesure de l'avancement des travaux du groupe de travail PKI de l'Internet Society. L'ISOC gère également le TLD .org et dispose donc de moyens pour s'attaquer au problème du spam.