

# Création d'une CA

sécurité, ssl, CA

Afin de pouvoir administrer nos certificats, nous créons une CA. Dans l'immédiat, tout se fera en ligne de commande.

## Configuration du CA

Nous créons le répertoire qui va accueillir le CA, et les répertoires qui le compose, et sécurisons le répertoire "private"

```
# sudo mkdir -p /etc/ssl/CA/{certs,crl,newcerts,private}
# chmod 700 /etc/ssl/CA/private
# cd /etc/ssl/CA
```

Nous créons le fichier d'index

```
# touch index.txt
# echo 1000 > serial
```

Nous copions le fichier openssl.cnf depuis /etc/ssl/ dans /etc/ssl/CA

```
# cp -v /etc/ssl/openssl.cnf /etc/ssl/CA/openssl.cnf
```

Nous adaptons le fichier openssl.cnf pour qu'il ressemble à cela

```
#
# OpenSSL example configuration file.
# This is mostly being used for generation of certificate requests.
#
# This definition stops the following lines choking if HOME isn't
# defined.
HOME                = .
RANDFILE            = $ENV::HOME/.rnd

# Extra OBJECT IDENTIFIER info:
#oid_file           = $ENV::HOME/.oid
oid_section         = new_oids

# To use this configuration file with the "-extfile" option of the
# "openssl x509" utility, name here the section containing the
# X.509v3 extensions to use:
# extensions        =
# (Alternatively, use a configuration file that has only
# X.509v3 extensions in its main [= default] section.)
```

```
[ new_oids ]

# We can add new OIDs in here for use by 'ca', 'req' and 'ts'.
# Add a simple OID like this:
# testoid1=1.2.3.4
# Or use config file substitution like this:
# testoid2=${testoid1}.5.6

# Policies used by the TSA examples.
tsa_policy1 = 1.2.3.4.1
tsa_policy2 = 1.2.3.4.5.6
tsa_policy3 = 1.2.3.4.5.7

#####
[ ca ]
default_ca = CA_default          # The default ca section

#####
[ CA_default ]

dir = /etc/ssl/CA                # Where everything is kept
certs = $dir/certs               # Where the issued certs are kept
crl_dir = $dir/crl               # Where the issued crl are kept
database = $dir/index.txt        # database index file.
#unique_subject = no             # Set to 'no' to allow creation of
                                # several certs with same subject.
new_certs_dir = $dir/newcerts    # default place for new certs.

certificate = $dir/certs/cacert.pem # The CA certificate
serial = $dir/serial              # The current serial number
crlnumber = $dir/crlnumber        # the current crl number
                                # must be commented out to leave a V1 CRL
crl = $dir/crl/crl.pem           # The current CRL
private_key = $dir/private/cakey.pem # The private key
RANDFILE = $dir/private/.rand    # private random number file

x509_extensions = usr_cert       # The extensions to add to the cert

# Comment out the following two lines for the "traditional"
# (and highly broken) format.
name_opt = ca_default            # Subject Name options
cert_opt = ca_default            # Certificate field options

# Extension copying option: use with caution.
# copy_extensions = copy

# Extensions to add to a CRL. Note: Netscape communicator chokes on V2 CRLs
# so this is commented out by default to leave a V1 CRL.
# crlnumber must also be commented out to leave a V1 CRL.
crl_extensions = crl_ext
```

```
default_days      = 375          # how long to certify for
default_crl_days= 30           # how long before next CRL
default_md       = default      # use public key default MD
preserve         = no           # keep passed DN ordering

# A few difference way of specifying how similar the request should look
# For type CA, the listed attributes must be the same, and the optional
# and supplied fields are just that :-)
policy           = policy_match

# For the CA policy
[ policy_match ]
countryName      = match
stateOrProvinceName = match
organizationName = match
organizationalUnitName = optional
commonName       = supplied
emailAddress     = optional

# For the 'anything' policy
# At this point in time, you must list all acceptable 'object'
# types.
[ policy_anything ]
countryName      = optional
stateOrProvinceName = optional
localityName     = optional
organizationName = optional
organizationalUnitName = optional
commonName       = supplied
emailAddress     = optional

#####
[ req ]
default_bits     = 2048
default_keyfile  = privkey.pem
distinguished_name = req_distinguished_name
attributes       = req_attributes
x509_extensions = v3_ca      # The extensions to add to the self signed cert

# Passwords for private keys if not present they will be prompted for
# input_password = secret
# output_password = secret

# This sets a mask for permitted string types. There are several options.
# default: PrintableString, T61String, BMPString.
# pkix    : PrintableString, BMPString (PKIX recommendation before 2004)
# utf8only: only UTF8Strings (PKIX recommendation after 2004).
# nombstr : PrintableString, T61String (no BMPStrings or UTF8Strings).
# MASK:XXXX a literal mask value.
# WARNING: ancient versions of Netscape crash on BMPStrings or UTF8Strings.
string_mask = utf8only
```

```
# req_extensions = v3_req # The extensions to add to a certificate request

[ req_distinguished_name ]
countryName          = Country Name (2 letter code)
countryName_default  = FR
countryName_min      = 2
countryName_max      = 2

stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = Var

localityName         = Locality Name (eg, city)
localityName_default = Vinon

0.organizationName   = Organization Name (eg, company)
0.organizationName_default = Grohub Certificate Authority

# we can do this but it is not needed normally :-)
#1.organizationName   = Second Organization Name (eg, company)
#1.organizationName_default = World Wide Web Pty Ltd

organizationalUnitName = Organizational Unit Name (eg, section)
organizationalUnitName_default = Security Team

commonName           = Common Name (e.g. server FQDN or YOUR name)
commonName_max       = 64

emailAddress         = Email Address
emailAddress_max     = 64
emailAddress_default = sec@grohub.org

# SET-ex3           = SET extension number 3

[ req_attributes ]
challengePassword    = A challenge password
challengePassword_min = 4
challengePassword_max = 20

unstructuredName     = An optional company name

[ usr_cert ]

# These extensions are added when 'ca' signs a request.

# This goes against PKIX guidelines but some CAs do it and some software
# requires this to avoid interpreting an end user certificate as a CA.

basicConstraints=CA:FALSE

# Here are some examples of the usage of nsCertType. If it is omitted
# the certificate can be used for anything *except* object signing.
```

```
# This is OK for an SSL server.
# nsCertType = server

# For an object signing certificate this would be used.
# nsCertType = objsign

# For normal client use this is typical
nsCertType = client, email

# and for everything including object signing:
# nsCertType = client, email, objsign

# This is typical in keyUsage for a client certificate.
# keyUsage = nonRepudiation, digitalSignature, keyEncipherment

# This will be displayed in Netscape's comment listbox.
nsComment = "OpenSSL Generated Certificate"

# PKIX recommendations harmless if included in all certificates.
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer

# This stuff is for subjectAltName and issuerAltname.
# Import the email address.
# subjectAltName=email:copy
# An alternative to produce certificates that aren't
# deprecated according to PKIX.
# subjectAltName=email:move

# Copy subject details
# issuerAltName=issuer:copy

#nsCaRevocationUrl = http://www.domain.dom/ca-crl.pem
#nsBaseUrl
#nsRevocationUrl
#nsRenewalUrl
#nsCaPolicyUrl
#nsSslServerName

# This is required for TSA certificates.
# extendedKeyUsage = critical,timeStamping
extendedKeyUsage = clientAuth, emailProtection

[ v3_req ]

# Extensions to add to a certificate request

basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment

[ v3_ca ]
```

```
# Extensions for a typical CA

# PKIX recommendation.

subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid:always,issuer

basicConstraints = critical,CA:true, pathlen:0

# Key usage: this is typical for a CA certificate. However since it will
# prevent it being used as an test self-signed certificate it is best
# left out by default.
# keyUsage = cRLSign, keyCertSign
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

# Some might want this also
# nsCertType = sslCA, emailCA

# Include email address in subject alt name: another PKIX recommendation
# subjectAltName=email:copy
# Copy issuer details
# issuerAltName=issuer:copy

# DER hex encoding of an extension: beware experts only!
# obj=DER:02:03
# Where 'obj' is a standard or added object
# You can even override a supported extension:
# basicConstraints= critical, DER:30:03:01:01:FF

[ crl_ext ]

# CRL extensions.
# Only issuerAltName and authorityKeyIdentifier make any sense in a CRL.

# issuerAltName=issuer:copy
authorityKeyIdentifier=keyid:always

[ proxy_cert_ext ]
# These extensions should be added when creating a proxy certificate

# This goes against PKIX guidelines but some CAs do it and some software
# requires this to avoid interpreting an end user certificate as a CA.

basicConstraints=CA:FALSE

# Here are some examples of the usage of nsCertType. If it is omitted
# the certificate can be used for anything *except* object signing.

# This is OK for an SSL server.
```

```
# nsCertType          = server

# For an object signing certificate this would be used.
# nsCertType = objsign

# For normal client use this is typical
# nsCertType = client, email

# and for everything including object signing:
# nsCertType = client, email, objsign

# This is typical in keyUsage for a client certificate.
# keyUsage = nonRepudiation, digitalSignature, keyEncipherment

# This will be displayed in Netscape's comment listbox.
nsComment          = "OpenSSL Generated Certificate"

# PKIX recommendations harmless if included in all certificates.
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer

# This stuff is for subjectAltName and issuerAltname.
# Import the email address.
# subjectAltName=email:copy
# An alternative to produce certificates that aren't
# deprecated according to PKIX.
# subjectAltName=email:move

# Copy subject details
# issuerAltName=issuer:copy

#nsCaRevocationUrl    = http://www.domain.dom/ca-crl.pem
#nsBaseUrl
#nsRevocationUrl
#nsRenewalUrl
#nsCaPolicyUrl
#nsSslServerName

# This really needs to be in place for it to be a proxy certificate.
proxyCertInfo=critical,language:id-ppl-anyLanguage,pathlen:3,policy:foo

#####
[ tsa ]

default_tsa = tsa_config1 # the default TSA section

[ tsa_config1 ]

# These are used by the TSA reply generation only.
dir      = ./demoCA      # TSA root directory
serial   = $dir/tsaserial # The current serial number (mandatory)
```

```
crypto_device = builtin          # OpenSSL engine to use for signing
signer_cert = $dir/tsacert.pem   # The TSA signing certificate
                                # (optional)
certs = $dir/cacert.pem         # Certificate chain to include in reply
                                # (optional)
signer_key = $dir/private/tsakey.pem # The TSA private key (optional)
signer_digest = sha256          # Signing digest to use. (Optional)
default_policy = tsa_policy1     # Policy if request did not specify it
                                # (optional)
other_policies = tsa_policy2, tsa_policy3 # acceptable policies
(optional)
digests = sha1, sha256, sha384, sha512 # Acceptable message digests
(mandatory)
accuracy = secs:1, millisecs:500, microsecs:100 # (optional)
clock_precision_digits = 0 # number of digits after dot. (optional)
ordering = yes # Is ordering defined for timestamps?
            # (optional, default: no)
tsa_name = yes # Must the TSA name be included in the reply?
            # (optional, default: no)
ess_cert_id_chain = no # Must the ESS cert id chain be included?
                    # (optional, default: no)

[ v3_intermediate_ca ]
# Extensions for a typical intermediate CA (`man x509v3_config`).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true, pathlen:0
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ server_cert ]
# Extensions for server certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = server
nsComment = "OpenSSL Generated Server Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth

[ ocsf ]
# Extension for OCSP signing certificates (`man ocsf`).
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, digitalSignature
extendedKeyUsage = critical, OCSPSigning
```

## Création de la clé root

Nous créons la clé root du CA

```
# openssl genrsa -aes256 -out private/cakey.pem 4096
```

Entrez deux fois la passphrase que vous avez défini

```
Enter pass phrase for private/cakey.pem:  
Verifying - Enter pass phrase for private/cakey.pem:
```

Et nous sécurisons la clé

```
# chmod 400 private/cakey.pem
```

## Création du certificat root

Nous créons le certificat

```
# openssl req -config openssl.cnf -key private/cakey.pem -new -x509 -days  
7300 -sha256 -extensions v3_ca -out certs/cacert.pem
```

Vous devez saisir la passphrase de la clé précédemment définie

```
Enter pass phrase for private/cakey.pem:
```

Répondez ensuite aux questions, ou laissez le choix par défaut (information entre crochets)

```
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) [FR]:  
State or Province Name (full name) [Var]:  
Locality Name (eg, city) [Vion]:  
Organization Name (eg, company) [Grohub Certificate Authority]:  
Organizational Unit Name (eg, section) [Security Team]:  
Common Name (e.g. server FQDN or YOUR name) []:Grohub Root CA  
Email Address [sec@grohub.org]:
```

Nous vérifions le certificat

```
# openssl x509 -noout -text -in certs/cacert.pem
```

## Configuration du certificat intermédiaire

Nous créons le répertoire qui va accueillir le certificat intermédiaire

```
# mkdir /etc/ssl/CA/intermediate
```

Nous nous plaçons dans le répertoire, et créons l'arborescence

```
# mkdir -p /etc/ssl/CA/intermediate/{certs,crl,csr,newcerts,private}
# chmod 700 /etc/ssl/CA/intermediate/private
# cd /etc/ssl/CA/intermediate
# touch index.txt
# echo 1000 > serial
```

Nous ajoutons un fichier pour garder une trace de la liste des révocation de certificats

```
# echo 1000 > /etc/ssl/CA/intermediate/crlnumber
```

Nous copions le fichier de configuration

```
# cp -v /etc/ssl/CA/openssl.cnf /etc/ssl/CA/intermediate/openssl.cnf
```

Nous changeons les éléments suivants

```
[ CA_default ]
dir                = /root/ca/intermediate
private_key        = $dir/private/intermediatekey.pem
certificate        = $dir/certs/intermediatecert.pem
crl                = $dir/crl/intermediatecrl.pem
policy             = policy_loose
```

## Création du certificat intermédiaire

```
cd /etc/ssl/CA
# openssl genrsa -aes256 -out intermediate/private/intermediatekey.pem 4096
```

Nous saisissons une passphrase

```
Enter pass phrase for intermediate/private/intermediatekey.pem:
Verifying - Enter pass phrase for intermediate/private/intermediatekey.pem:
```

Nous sécurisons la clé

```
chmod 400 intermediate/private/intermediatekey.pem
```

## Création du Certificat Signing Request (CSR)

Nous créons le CSR

```
# cd /etc/ssl/CA
```

```
# openssl req -config intermediate/openssl.cnf -new -sha256 -key
intermediate/private/intermediatekey.pem -out
intermediate/csr/intermediatecsr.pem
```

## Création du certificat

Nous créons le certificat

```
# cd /etc/ssl/CA
# openssl ca -config openssl.cnf -extensions v3_intermediate_ca -days 3650 -
notext -md sha256 -in intermediate/csr/intermediatecsr.pem -out
intermediate/certs/intermediatecert.pem
```

Nous confirmons deux fois avec "y"

```
Sign the certificate? [y/n]:y
1 out of 1 certificate requests certified, commit? [y/n]y
```

Nous sécurisons le certificat

```
# chmod 444 intermediate/certs/intermediatecert.pem
```

Pour vérifier le certificat

```
# openssl x509 -noout -text -in intermediate/certs/intermediatecert.pem
```

Pour vérifier

```
# openssl verify -CAfile certs/cacert.pem
intermediate/certs/intermediatecert.pem
```

Vous devriez avoir cette réponse

```
intermediate/certs/intermediatecert.pem: OK
```

## Création du fichier de chaîne de certificat

```
# cat intermediate/certs/intermediatecert.pem certs/cacert.pem >
intermediate/certs/ca-chaincert.pem
```

Nous sécurisons le fichier

```
# chmod 444 intermediate/certs/ca-chaincert.pem
```

## Liens

- [OpenSSL Certificate Authority](#)

From:

<https://wiki.grohub.org/> - **Grohub wiki**

Permanent link:

<https://wiki.grohub.org/infrastructure/secure/ssl/ca-creation>

Last update: **11/04/2021 14:54**

