</>

Libreria OpenSSL

CERTIFICATI, CHIAVI E AUTORITÀ DI CERTIFICAZIONE

Perchè OpenSSL?

OpenSSL rappresenta la principale implementazione software utilizzata dai fornitori per il protocollo TLS.
Essa funge da libreria completa e supporta un'ampia gamma di primitive crittografiche.
Nella mia attività, adotterò la versione a linea di comando di OpenSSL per stabilire un'autorità di certificazione all'interno della mia rete virtuale privata. Quest'autorità sarà in grado di rilasciare certificati ai diversi nodi presenti nella rete.
Inoltre, OpenSSL offre strumenti per testare l'implementazione corretta dello stack TLS. Questa funzionalità riveste particolare importanza durante le fasi di troubleshooting e nel Poc (Proof of Concept).

La mia attività di documentazione si suddivide quindi in due macro aree relative a OpenSSL:

1. Gestione e creazione delle chiavi e dei certificati
2. Creazione di un' Autorità di Certificazione privata

Generazione chiavi

OpenSSL mette a disposizione il comando "genpkey" per la generazione di chiavi.
Gli switch più importanti sono quelli che permettono di specificare l' algoritmo e le opzioni relative all' algoritmo scelto:

```
openssl genpkey -out fd.key \
    -algorithm RSA
    -pkeyopt rsa_keygen_bits:2048
```

Nell' esempio viene creata la chiave RSA fd.key a 2048 bit.
Essa può anche essere protetta con AES-128 utilizzando un passphrase come chiave aggiungendo lo switch "-aes-128-cbc" al comando precedente.

La chiave creata ovviamente è in formato PKCS#8 e quindi per visualizzare la struttura si usano gli switch "-in INPUTKEY -text -noout" dove INPUTKEY è la chiave generata.

Creazione di una CSR (Certificate Signing Requests)

Una volta generata la chiave privata, è possibile creare una CSR con OpenSSL.
La CSR è una richiesta formale alla CA privata di farsi generare un certificato. Questa richiesta verrà inviata, nell' esercizio pratico, dalle terminazioni TLS per farsi generare un certificato firmato dalla CA privata.

```
openssl req -new -key fd.key -out fd.csr
```

Il comando chiederà all' utente in modo interattivo le varie informazioni relative al Certificate Subject, le quali verranno concatenate alla chiave pubblica e il tutto sarà firmato con la chiave privata fd.key per generare la CSR fd.csr.
NOTA BENE: la chiave pubblica non è richiesta dal comando perché viene ricavata da quella privata.
INFO: E' possibile, per automatizzare, specificare un file .cnf e usare lo switch -config per inviare già tutte le informazioni al comando senza interattività.

La CSR creata ovviamente è in formato PKCS#10 e quindi per visualizzare la struttura si usano gli switch: "-in INPUTCSR -text -noout" dove INPUTCSR è la CSR creata.

Creazione certificato

Con la CSR è possibile farsi generare un certificato da una Public CA o generarsi da soli un certificato self-signed. Nella mia attività pratica la CSR generata da ogni nodo di terminazione TLS verrà inviata alla CA Privata che provvederà a restituire un certificato firmato.

```
openssl x509 -req -days 365 -in fd.csr -signkey fd.key -out fd.crt
```

Il comando verifica la firma della CSR fd.csr estraendo la chiave pubblica e poi usa la CSR per riempire le informazioni del certificato. Imposta la validità a 365 giorni e poi firma il tutto con la sua chiave privata fd.key generando il certificato fd.crt.
Molte volte i certificati hanno bisogno di aggiungere più informazioni, infatti nel formato dei certificati x.509 c'è la possibilità di aggiungere estensioni ai certificati. Queste estensioni possono essere raggruppate in un file fd.ext e incluse nel certificato aggiungendo al comando precedente lo switch "-extfile fd.ext".

Per visualizzare la struttura del certificato si usano gli switch: "-in INPUTCRT -text -noout" dove
INPUTCRT è il certificato creato.

Creazione di una Certificate Authority Privata

- Configurazione della Root CA
- Creazione Struttura Directory della Root CA
- Generazione chiavi della Root CA
- Specifica operazioni della Root CA
- Creazione del certificato per OCSP Responder
- Configurazione della Subordinate CA
- Generazione chiavi della Subordinate CA
- Specifica operazioni della Subordinate CA

Configurazione della Root CA

Prima di poter creare una CA, si prepara il file di configurazione root-ca.conf che dirà a OpenSSL le impostazioni della Root CA.
La prima parte del file contiene informazioni base della Root CA ed è divisa in due sezioni: default e ca_dn.

```
[default]
name                    = root-ca
domain_suffix           = example.com
aia_url                 = http://$name.$domain_suffix/$name.crt
crl_url                 = http://$name.$domain_suffix/$name.crl
ocsp_url                = http://ocsp.$name.$domain_suffix:9080
default_ca              = ca_default
name_opt                = utf8,esc_ctrl,multiline,lname,align

[ca_dn]
countryName             = "GB"
organizationName        = "Example"
commonName              = "Root CA"
```

- La prima specifica suffisso del dominio, a quale url corrispondono le liste CRL e il responder OCSP ecc.. Specifica importante è quella legata al defalut_ca che specifica la sezione all' interno del file che è responsabile del comportamento di default della Root CA.

- La seconda specifica il distinguished name della Root CA

Configurazione della Root CA

La seconda parte controlla le operazioni della Root CA e i percorsi dove verranno salvati i file che serviranno per il corretto funzionamento.
Ad esempio il file database o il percorso dove verranno salvati i certificati rilasciati.
Nel "ca_default" è anche presente l' impostazione per l'algoritmo di hashing per le firme e una policy che viene controllata prima che la Root CA rilasci il certificato.
In questo caso è stata creata la policy "policy_c_o_match" che controlla che i certificati rilasciati dalla Root CA abbiano countryName e organizationName uguali a quelli della Root CA

```
[ca_default]
home                    = .
database                = $home/db/index
serial                  = $home/db/serial
crlnumber               = $home/db/crlnumber
certificate             = $home/$name.crt
private_key             = $home/private/$name.key
RANDFILE                = $home/private/random
new_certs_dir           = $home/certs
unique_subject          = no
copy_extensions         = none
default_days            = 3650
default_crl_days        = 365
default_md              = sha256
policy                  = policy_c_o_match

[policy_c_o_match]
countryName             = match
stateOrProvinceName     = optional
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
```

Configurazione della Root CA

La terza parte contiene la configurazione per il comando "req", ovvero quello relativo alla creazione delle CSR.
Questa sezione verrà usata solo una volta per la creazione del certificato di root che sarà self-signed.
Visto che il certificato di root deve essere ben protetto si usa una chiave a 4096 bit.
Il certificato creato aggiungerà le estensioni x.509 specificate nella sezione "ca_ext".
La sezione "ca_ext" stabilisce che il certificato generato deve essere flaggato come appartenente ad un CA e l' utilizzo della chiave deve essere adibito solo alle firme dei certificati e alla firma delle liste CRL.

```
[req]
default_bits            = 4096
encrypt_key             = yes
default_md              = sha256
utf8                    = yes
string_mask             = utf8only
prompt                  = no
distinguished_name      = ca_dn
req_extensions          = ca_ext

[ca_ext]
basicConstraints        = critical,CA:true
keyUsage                = critical,keyCertSign,cRLSign
subjectKeyIdentifier    = hash
```

Configurazione della Root CA

La quarta parte contiene informazioni sui vincoli alle CA subordinate. Le estensioni dei certificati rilasciati dai Sub CA possono rilasciare certificati solo relativi ai subject all' interno della sezione "name_constraints".
Inoltre i Sub CA non possono avere altri Sub CA grazie al basicConstraint pathlen:0.
Grazie all' extendedKeyUsage le chiavi dei certificati rilasciati dai Sub CA possono autenticare client e server.
"crl_info" è la sezione che inserisce l'url delle liste CRL nei certificati.
"issuer_info" è la sezione che inserisce l'url del servizio OCSP nei certificati.

```
[sub_ca_ext]
authorityInfoAccess     = @issuer_info
authorityKeyIdentifier  = keyid:always
basicConstraints        = critical,CA:true,pathlen:0
crlDistributionPoints   = @crl_info
extendedKeyUsage        = clientAuth,serverAuth
keyUsage                = critical,keyCertSign,cRLSign
nameConstraints         = @name_constraints
subjectKeyIdentifier    = hash

[crl_info]
URI.0                   = $crl_url

[issuer_info]
caIssuers;URI.0         = $aia_url
OCSP;URI.0              = $ocsp_url

[name_constraints]
permitted;DNS.0=example.com
permitted;DNS.1=example.org
excluded;IP.0=0.0.0.0/0.0.0.0
excluded;IP.1=0:0:0:0:0:0:0:0/0:0:0:0:0:0:0:0
```

Creazione della struttura di directory della Root CA

Le directory da creare sono:

1. certs/ che è la directory adibita allo storage di certificati rilasciati. Questa directory viene usata quando si vuole revocare un certificato rilasciato.
2. db/ che è la directory utilizzata per il database. Usata per i numeri seriali delle liste CRL.
3. private/ che è la directory che archivia le due chiavi private, una per il Root CA e l' altra per il responder OCSP. Per questioni di sicurezza è meglio impostare permessi solo all' owner con chmod.

```
$ mkdir root-ca
$ cd root-ca
$ mkdir certs db private
$ chmod 700 private
$ touch db/index
$ openssl rand -hex 16  > db/serial
$ echo 1001 > db/crlnumber
```

Generazione chiavi della Root CA

Per creare adesso la Root CA dobbiamo generare la Root Key e la CSR. Le informazioni per generarli le prendiamo dal file di configurazione, nello specifico dalla sezione "req".
Con lo switch -config è possibile indicare quale sia il file di configurazione.

```
$ openssl req -new \
    -config root-ca.conf \
    -out root-ca.csr \
    -keyout private/root-ca.key
```

Poi dobbiamo generare il self-signed root certificate usando il comado ca e lo switch -selfsign.
Le estensioni da aggiungere al certificato sono presenti nel file di configurazione nella sezione ca_ext.
NOTA BENE: in questo caso invece del comando x509 si utilizza il comando ca, questo perchè ca è adatto proprio a fare quello che fa un CA mentre x509 serve solo per la gestione e la conversione di certificati.

```
$ openssl ca -selfsign \
    -config root-ca.conf \
    -in root-ca.csr \
    -out root-ca.crt \
    -extensions ca_ext
```

Specifica operazioni della Root CA

Con lo switch -gencrl è possibile generare la lista CRL (lista dei certificati revocati) root-ca.crl, i nodi della rete scaricando questa lista possono controllare se i certificati che ricevono sono stati revocati o no.

```
$ openssl ca -gencrl \
    -config root-ca.conf \
    -out root-ca.crl
```

Per rilasciare il certificato della Sub CA è possibile usare il comando ca aggiungendo le estensioni specificate nel file di configurazione nella sezione "sub_ca_ext".
NOTA BENE: la sezione ca_ext non viene usata per rilasciare certificati perche serviva solo per generare il root certificate.

```
$ openssl ca \
    -config root-ca.conf \
    -in sub-ca.csr \
    -out sub-ca.crt \
    -extensions sub_ca_ext
```

Per revocare un certificato si usa lo switch -revoke con il certificato presente nella directory certs/. E' necessario identificare anche la motivazione con lo switch -crl_reason.

```
$ openssl ca \
    -config root-ca.conf \
    -revoke certs/1002.pem \
    -crl_reason keyCompromise
```

Creazione del certificato per OCSP Responder

OCSP (Online Certificate Status Protocol) è un' alternativa alle liste CRL, per controllare lo stato di un certificato in real time.
Prima cosa da fare è la generazione della chiave e la CSR per l' OCSP Responder. In questo caso non è stata specificata nessuna sezione nel file di configurazione per questa procedura quindi si definisce l' algoritmo della chiave con lo switch -newkey.
A questo punto si utilizza il root CA per firmare e rilasciare il certificato utilizzando il file di configurazione alla sezione "ocsp_ext" che specifica le giuste estensioni per il certificato OCSP, inoltre si inserisce la validità a 30 giorni.
Infine si starta il servizio OCSP sulla stessa macchina del CA Root sulla porta 9080.
Si noti come certificato scambiato e chiave usata per le firme siano quelle del OCSP Responder e non del Root CA.

```
$ openssl req -new \
    -newkey rsa:2048 \
    -subj "/C=GB/O=Example/CN=OCSP Root Responder" \
    -keyout private/root-ocsp.key \
    -out root-ocsp.csr
```

```
$ openssl ca \
    -config root-ca.conf \
    -in root-ocsp.csr \
    -out root-ocsp.crt \
    -extensions ocsp_ext \
    -days 30
```

```
$ openssl ocsp \
    -port 9080
    -index db/index \
    -rsigner root-ocsp.crt \
    -rkey private/root-ocsp.key \
    -CA root-ca.crt \
    -text
```

Configurazione della Subordinate CA

```
[default]
name                    = sub-ca
ocsp_url                = http://ocsp.$name.$domain_suffix:9081

[ca_dn]
countryName             = "GB"
organizationName        = "Example"
commonName              = "Sub CA"

[ca_default]
default_days            = 365
default_crl_days        = 30
copy_extensions         = copy
```

Anche nella configurazione della Subordinate CA è necessaria la generazione di un file di configurazione sub-ca.conf.
Però è possibile partire dal file root-ca.conf e modificare solo alcuni pezzi in quanto sono molto simili.
Il nome e il distinguished name vanno cambiati, come la porta del servizio OCSP del Sub CA (da 9080 a 9081).
Inoltre il copy_extensions è impostato a copy, il che significa che le estensioni specificate nel CSR sono copiate nel certificato, ma non tutte! Solo quelle che non abbiamo settato.
Inoltre la validità dei certificati sarà 365 giorni e la lista CRL verrà rigenerata ogni 30.

Configurazione della Subordinate CA

```
[server_ext]
authorityInfoAccess     = @issuer_info
authorityKeyIdentifier  = keyid:always
basicConstraints        = critical,CA:false
crlDistributionPoints   = @crl_info
extendedKeyUsage        = clientAuth,serverAuth
keyUsage                = critical,digitalSignature,keyEncipherment
subjectKeyIdentifier    = hash

[client_ext]
authorityInfoAccess     = @issuer_info
authorityKeyIdentifier  = keyid:always
basicConstraints        = critical,CA:false
crlDistributionPoints   = @crl_info
extendedKeyUsage        = clientAuth
keyUsage                = critical,digitalSignature
subjectKeyIdentifier    = hash
```

Alla fine del file aggiungiamo due nuove sezioni utilizzate per le estensioni da aggiungere nei certificati rilasciati per i server e per i client.
L' unica differenza è nel keyUsage e nel extendedKeyUsage dove il server può anche fare serverAuth e keyEncipherment.
Notare come nel basicConstraint il CA sia falso in quanto i certificati non appartengono ad una CA ma client o a server.

Per la creazione delle directory si usa la stessa struttura solo che la root directory si chiama sub-ca

Generazione chiavi della Subordinate CA

```
$ openssl req -new \
    -config sub-ca.conf \
    -out sub-ca.csr \
    -keyout private/sub-ca.key
```

```
$ openssl ca \
    -config root-ca.conf \
    -in sub-ca.csr \
    -out sub-ca.crt \
    -extensions sub_ca_ext
```

Come nel caso della Root CA si genera chiave e CSR per la Sub CA. Tutte le informazioni necessarie sono prese dal file di configurazione sub-ca.conf nella sezione "req" che non è stata modificata.

Poi dobbiamo far rilasciare dal Root CA il certificato firmato tramite le informazioni ottenute dalla CSR e come estensione utilizziamo la sezione "sub_ca_ext" del file di configurazione del Root CA.

Specifica operazioni della Subordinate CA

```
$ openssl ca \
    -config sub-ca.conf \
    -in server.csr \
    -out server.crt \
    -extensions server_ext
```

```
$ openssl ca \
    -config sub-ca.conf \
    -in client.csr \
    -out client.crt \
    -extensions client_ext
```

A questo punto il Subordinate CA può rilasciare i certificati ai server e ad i client utilizzando le apposite sezioni specificate nel file di configurazione sub-ca.conf.
Le estensioni da aggiungere ai certificati rilasciati ai server della rete sono specificate nella sezione "server_ext".
Le estensioni da aggiungere ai certificati rilasciati ai client della rete sono specificate nella sezione "client_ext".

L' ultima cosa che rimane da fare sono la generazione della lista CRL e la configurazione del Responder OCSP.
Tuttavia il procedimento è lo stesso l' unica cosa che cambia è la porta del OCSP Responder che deve essere diversa (9081).

A questo punto la Private Certificate Authority è pronta all'uso

FINE