Gestione di una Certification Authority con OpenSSL

Introduzione a OpenSSL

OpenSSL è una realizzazione in forma di software libero dei protocolli SSL/TLS (*Secure socket layer* e *Transport layer security*) per la certificazione e la comunicazione cifrata. OpenSSL si compone di alcune librerie che permettono di incorporare le funzionalità dei protocolli SSL/TLS all'interno di programmi di comunicazione, oltre a una serie di programmi di servizio per la gestione delle chiavi e dei certificati, arrivando eventualmente anche alla gestione di un'autorità di certificazione. Esistono oramai versioni delle librerie OpenSSL sia per ambienti UNIX-like che per ambienti Windows. In questa esercitazione utilizzeremo le la libreria OpenSSL disponibile all'interno dell'emulatore Linux Cygwin.

Collocazione e impostazione

L'ambiente Cygwin contiene le librerie OpenSSL nel percorso `/usr/ssl`. All'interno di questa directory vi sono una serie di altre directory ed il file `openssl.cnf` che contiene le impostazioni di configurazione della libreria. La sintassi per i comandi relativi si esprime sinteticamente nel modo seguente:

```
openssl comando [opzioni]
```

Tabella 1.1. Alcuni comandi di OpenSSL.

Comando	Descrizione
`openssl req`	Gestione delle richieste di certificazione.
`openssl ca`	Gestione relativa all'autorità di certificazione (firma dei cert.).
`openssl crl`	Gestione del certificato delle revoche.
`openssl genrsa`	Generazione di parametri RSA.
`openssl rsa`	Conversione del formato di una chiave privata o di un certificato.
`openssl x509`	Gestione dei dati dei certificati X.509.

Accanto alla lista dei comandi riportati nella tabella precedente è utile cominciare a prendere familiarità con le principali opzioni da associare ai suddetti comandi:

Tabella 1.2. Alcune opzioni frequenti nei comandi di OpenSSL.

Opzione	Descrizione
`-in` *file*	Definisce un file in ingresso adatto al contesto.

`-out` *file*	Definisce un file in uscita adatto al contesto.
`-noout`	Non emette il risultato.
`-text`	Emette le informazioni in forma di testo leggibile.
`-hash`	Emette il codice di controllo relativo al contesto.
`-inform` *formato*	Specifica il formato dei dati in ingresso.
`-outform` *formato*	Specifica il formato dei dati in uscita.

Cenni sulla configurazione di OpenSSL

La configurazione di OpenSSL si attua normalmente attraverso il file `openssl.cnf`, che si trova collocato nella directory `/usr/ssl`. Osservandone il contenuto, si intuisce che il simbolo **#** serve a introdurre un commento, fino alla fine della riga relativa; inoltre si comprende che le righe vuote e quelle bianche vengono ignorate come i commenti; infine, si vede che le direttive del file sono degli assegnamenti a variabili, che se necessario si espandono con il prefisso **$**, e le direttive sono raggruppate in sezioni individuabili da un titolo tra parentesi quadre. È importante osservare che le sezioni sono organizzate in modo gerarchico, a partire dai nomi dei comandi di OpenSSL. In pratica, per il comando **`openssl req`** si prende in considerazione la sezione **`[req]`**, che poi può a sua volta richiamare altre sottosezioni. Vale dare un'occhiata a un estratto della configurazione relativa `ad` **`openssl req`**.

```
[ req ]
default_bits            = 1024
default_keyfile         = privkey.pem
distinguished_name      = req_distinguished_name
attributes              = req_attributes
x509_extensions = v3_ca # The extentions to add to the self signed cert

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

stateOrProvinceName             = State or Province Name (full name)
stateOrProvinceName_default     = Some-State

localityName                    = Locality Name (eg, city)
```

Nelle prossime sezioni viene mostrato come simulare la gestione di un'autorità di certificazione attraverso OpenSSL. Il file di configurazione standard dovrebbe essere neutro rispetto a questo problema, incorporando una sezione **`[ ca ]`** particolare, utile per fare delle prove:

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

####################################################################
[ CA_default ]

dir             = ./demoCA              # Where everything is kept
certs           = $dir/certs            # Where the issued certs are kept
```

```
crl_dir         = $dir/crl              # Where the issued crl are kept
database        = $dir/index.txt        # database index file.
new_certs_dir   = $dir/newcerts         # default place for new certs.

certificate     = $dir/cacert.pem       # The CA certificate
serial          = $dir/serial           # The current serial number
crl             = $dir/crl.pem          # The current CRL
private_key     = $dir/private/cakey.pem# The private key
RANDFILE        = $dir/private/.rand    # private random number file
```

Per il momento è bene osservare che con la direttiva **`dir`** viene definita una variabile, che poi viene presa in considerazione di nuovo, espandendola con l'aggiunta del prefisso **`$`** (**`$dir`**), nei valori da assegnare ad altre variabili. Questa variabile serve a definire la directory di partenza a partire dalla quale vanno collocati una serie di file che riguardano l'amministrazione dell'autorità di certificazione. Inizialmente, viene indicata una directory che appare volutamente improbabile, `./demoCA/`, proprio per fare capire che prima di lavorare sul serio occorre pensarci bene e mettere mano alla configurazione

Politica dell'autorità di certificazione

Nella sezione che descrive il funzionamento del comando **`openssl ca`**, deve apparire anche l'indicazione del tipo di politica che l'autorità di certificazione intende attuare per rilasciare i certificati. Naturalmente, quello che può essere definito qui è solo qualche aspetto che riguarda la definizione del nome distintivo del titolare. Quello che segue è un altro estratto del file di configurazione in cui si vede l'assegnamento del nome di una sottosezione alla variabile **`policy`**.

```
policy                  = policy_match

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

[ policy_anything ]
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
```

In questo caso, la sottosezione **`[policy_match]`** specifica che i campi del paese, della regione e dell'organizzazione, devono corrispondere con gli stessi dati del certificato della stessa autorità di certificazione. In pratica, questo servirebbe a limitare l'accesso all'autorità soltanto a chi appartiene alla stessa area e anche alla stessa organizzazione (ciò fa pensare a un'autorità di certificazione aziendale, competente solo nell'ambito della propria azienda). Per il resto, solo il campo CN deve essere fornito, mentre gli altri sono facoltativi.

Sotto alla sottosezione appena descritta, appare anche un'altra sottosezione simile, con il nome **`[policy_anything]`**, in cui verrebbe concesso quasi tutto, a parte l'obbligo di fornire il CN.

Simulazione dell'allestimento e del funzionamento di un'autorità di certificazione

L'utilizzo di OpenSSL per la gestione di un'autorità di certificazione richiede la conoscenza di molti dettagli sul funzionamento di questo sistema. In generale, il file di configurazione predefinito consente di ottenere delle richieste di certificati o di generare dei certificati fittizi auto-firmati. In questo gruppo di sezioni si vuole mostrare schematicamente l'uso di OpenSSL nella gestione di un'autorità di certificazione, anche con qualche esempio, ma senza l'intenzione di arrivare a ottenere dei certificati realistici.

Autorità di certificazione autonoma

La creazione di un'autorità di certificazione autonoma, ovvero di un'autorità principale (*root*), che non abbia ottenuto a sua volta un certificato da un'autorità di livello superiore, deve realizzare la sua chiave privata e il suo certificato auto-firmato. Diversamente, se dipendesse dalla certificazione di un'altra autorità, dovrebbe predisporre la propria richiesta, sottoporla all'autorità superiore da cui dovrebbe ottenere il certificato. Per la creazione di una nuova autorità di certificazione, l'ambiente OpenSSL mette a disposizione uno script perl (CA.pl) che si trova nella directory `/usr/ssl/misc`. Attraverso questo script, in base alla configurazione contenuta nel file `openssl.cnf` viene alle gestita la nuova CA. Si provi ad eseguire il comando `./CA.pl -newca` (premere invio non inserire nessun nome). Viene avviata la procedura per la creazione di una nuova CA con il relativo certificato.

```
Making CA certificate ...
Generating a 2048 bit RSA private key
.....................................................+++
.............+++
writing new private key to './demoCA/private/cakey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IT
State or Province Name (full name) [Some-State]:CS
Locality Name (eg, city) []:Rende
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Corso
Organizational Unit Name (eg, section) []:C
Common Name (e.g. server FQDN or YOUR name) []:CC
Email Address []:g@g.com
```

Dopo aver inserito tutte le informazioni insieme ad una password per la protezione della chiave privata della CA il risultato ottenuto è il seguente:

```
Enter pass phrase for ./demoCA/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number:
            9b:e5:f7:b9:3a:07:5f:cd
        Validity
            Not Before: Oct 30 09:06:39 2017 GMT
            Not After : Oct 29 09:06:39 2020 GMT
        Subject:
            countryName               = IT
            stateOrProvinceName       = CS
            organizationName          = Corso
            organizationalUnitName    = C
            commonName                = CC
            emailAddress              = g@g.com
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                B1:04:82:2B:AF:43:74:2E:D5:63:8F:AA:5F:37:3B:1C:98:6B:48:36
            X509v3 Authority Key Identifier:
                keyid:B1:04:82:2B:AF:43:74:2E:D5:63:8F:AA:5F:37:3B:1C:98:6B:48:36

            X509v3 Basic Constraints:
                CA:TRUE
Certificate is to be certified until Oct 29 09:06:39 2020 GMT (1095 days)
```

Richiesta di certificazione

Teoricamente, il certificato che identifica e garantisce l'identità del servizio che si gestisce, deve essere fornito da un'autorità di certificazione. In questo caso, per farlo, deve ricevere un documento intermedio, definibile come una richiesta di certificazione. La chiave pubblica che vi viene inserita si ottiene a partire dalla chiave privata, mentre gli altri dati necessari per il certificato che si vuole ottenere si inseriscono in modo interattivo.

Vediamo come generare un nuova coppia di chiavi per poi richiedere un certificato. Per generare una chiave privata in chiaro, si utilizza il comando **`openssl genrsa`**, in un modo simile a quello seguente ottenendo il file `chiave_privata.pem` di 1024 bit:

```
$ openssl genrsa -out chiave_privata.pem 1024
```

Eventualmente, per creare una chiave privata cifrata, basta aggiungere un'opzione a scelta tra **`-des`**, **`-des3`** e **`-idea`**, che stanno a indicare rispettivamente gli algoritmi DES, DES-triplo e IDEA. Viene mostrato il caso in cui si utilizza l'opzione **`-des3`**. Per generare una nuova richiesta a partire dalla chiave generata si esegue:

```
$ openssl req -new -key chiave_privata.pem -out richiesta.pem
```

Viene richiesta la compilazione dei campi del certificato che deve essere firmato da una CA.

Per verificare il contenuto del certificato, che nel suo formato PEM non è leggibile direttamente, si può usare il comando **`openssl req`** con l'opzione **`-text`**:

```
$ openssl req -text -in richiesta.pem
```

Rilascio di certificazioni

Per le operazioni di rilascio dei certificati, ovvero della firma di questi a partire dai file di richiesta relativi, occorre prendere confidenza con l'uso di alcuni file, contenenti rispettivamente l'indice dei certificati rilasciati e il numero di serie successivo che può essere utilizzato. I certificati rilasciati da

un'autorità di certificazione hanno un numero seriale progressivo; in base al pezzo di configurazione mostrato in precedenza, questo numero viene conservato nel file `demoCA/serial`. Il numero in questione viene annotato secondo una notazione esadecimale, tradotta in caratteri normali, ma senza alcun prefisso. In pratica, dopo aver predisposto il certificato della stessa autorità, occorre mettere in questo file la riga seguente, conclusa da un codice di interruzione di riga finale e nulla altro:

```
01
```

La creazione dei certificati incrementa automaticamente questo numero; inoltre, se non viene specificato il file da creare, si ottiene direttamente un file corrispondente al suo numero di serie, con l'aggiunta dell'estensione consueta, collocato nella directory prevista per l'accumulo provvisorio: `demoCA/newcerts/` nel caso della configurazione di esempio a cui si continua a fare riferimento. La creazione di un certificato aggiorna anche il file che ne contiene l'indice, che potrebbe essere `demoCA/index.txt`. Inizialmente, dopo la creazione del certificato dell'autorità stessa, questo indice è semplicemente un file vuoto; con la creazione dei certificati successivi, viene aggiunta una riga per ognuno di questi, che va intesa come un record suddiviso in campi separati da un carattere di tabulazione **singolo**. Viene mostrato subito l'esempio del record relativo a un primo certificato (diviso in due righe per motivi tipografici):

```
V               001213190753Z                   01              unknown         \
  \/C=IT/ST=CS/O=UNICAL/CN=richiedente /Email=prova@reti.it
```

Nell'esempio non si vede, ma c'è un terzo campo nullo prima del valore **`01`**. I campi hanno il significato seguente:

1. lo stato del certificato, attraverso una lettera: «R», revocato, «E», scaduto, «V», valido;
2. la data di scadenza, scritta attraverso una stringa di cifre numeriche terminate da una lettera «Z» maiuscola, dove le coppie di cifre rappresentano rispettivamente: anno, mese, giorno, ore, minuti, secondi (***AAMMGGHHMMSS*z**);
3. la data di revoca del certificato, scritta esattamente come nel caso del secondo campo, solitamente assente, a indicare che il certificato è ancora valido;
4. il numero di serie in esadecimale;
5. la collocazione del certificato (attualmente si tratta sempre della parola chiave **`unknown`**);
6. i dati del titolare del certificato, ovvero il nome distintivo e l'indirizzo di posta elettronica di questo.

La creazione, ovvero la firma di un certificato si ottiene con il comando **`openssl ca`**, fornendo in particolare il file contenente la richiesta. Per esempio, se si vuole accettare la richiesta costituita dal file `richiesta.pem`, si potrebbe agire nel modo seguente:

```
$ openssl ca -in richiesta.pem -out certificato.cer
```

Dopo aver inserito la password che protegge la chiave private della CA si ottiene il certificato.

Revoca dei certificati

Se si incontra la necessità di revocare dei certificati prima della loro scadenza normale, si deve pubblicare un elenco di revoca, o CRL (*Certificate revocation list*). Questo elenco si produce con OpenSSL a cominciare dalla modifica del file contenente l'elenco dei certificati (`./demoCA/index.txt`), sostituendo la lettera «V» con la lettera «R» e inserendo la scadenza anticipata nel terzo campo. L'esempio seguente mostra il caso di due certificati che vengono revocati prima della scadenza:

```
R       001213192838Z   000113192840Z   01      unknown /C=IT/ST=Italia/...
R       001213202243Z   000113192840Z   02      unknown /C=IT/ST=Italia/...
```

Successivamente, basta usare il comando **`openssl ca`**, con l'opzione **`-gencrl`**:

```
$ openssl ca -gencrl -out ./demoCA/crl/crl.pem
```

Con questo esempio, viene creato il file `./demoCA/crl/crl.pem`, contenente questo elenco di revoca, il cui contenuto può essere riletto con il comando seguente:

```
$ openssl crl -text -in ./demoCA/crl/crl.pem
```

```
Certificate Revocation List (CRL):
        Version 1 (0x0)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: /C=IT/ST=Italia/L=Rende/O=UNICAL/CN=reti...
        Last Update: Jan 15 20:35:52 2000 GMT
        Next Update: Feb 14 20:35:52 2000 GMT
Revoked Certificates:
    Serial Number: 01
        Revocation Date: Jan 13 19:28:40 2000 GMT
    Serial Number: 02
        Revocation Date: Jan 13 19:28:40 2000 GMT
    Signature Algorithm: md5WithRSAEncryption
        32:e1:97:92:96:2f:0c:e4:df:bb:9c:82:a5:e3:5b:51:69:f5:
        51:ad:1b:b2:98:eb:35:a6:c8:7f:d9:29:1f:b2:1e:cc:da:84:
    ...
        31:27:4a:21:4c:7a:bc:85:73:cd:ff:15:9d:cb:81:b3:0b:82:73:50
```

Conversione nei formati

In generale, con OpenSSL si lavora con file (richieste, certificati, elenchi di revoca, ecc.) in formato PEM, che è in pratica una forma compatta dei dati, utilizzando però solo il codice ASCII a 7 bit. Ci sono situazioni in cui è necessario convertire questo formato in un altro, oppure è necessario acquisire dei dati da un formato diverso dal solito. In generale, quando si usano comandi che possono ricevere dati in ingresso, o quando si devono generare dati in uscita, sempre relativi a certificati e affini, si possono usare rispettivamente le opzioni **`-inform`** e **`-outform`**, seguite dalla sigla del formato (non sono disponibili sempre tutti). Vengono mostrati alcuni esempi.

```
$ openssl x509 -in certificato.pem -out certificato.der
```

In questo modo si ottiene la conversione del certificato `certificato.pem` nel file `certificato.der`, che risulta in formato DER (binario).

```
$ openssl crl -in crl.pem -out crl.der
```

Converte l'elenco di revoca `crl.pem` in formato DER, nel file `crl.der`.

Usare OpenSSL per firmare le Email

Ad esempio, la Maggio parte dei client di posta vuole in ingresso un certificato in formato PKCS#12. Questo formato è utile per salvare in un unico file il certificato di chiave pubblica e la chiave privata cifrata. Per la conversione possiamo usare il comando pkcs12.

```
$ openssl pkcs12 -in crl.pem –inkey privatekey.pem – export –out finalCert.p12
```

Il file .p12 può essere importato assieme al certificato della CA nel Keyring di OSX (o, ad esempio, nel gestore di certificati di Thunderbird). Il client email da questo momento in poi sarà automaticamente in grado di firmare le e-mail inviate dall'account specificato nel certificato.