Gestire Certificati Digitali con openssl

Creare una Certification Authority e gestire localmente dei certificati digitali è ragionevolmente semplice, anche se i problemi aumentano, e di molto, se il sistema è di dimensioni medio-grande. Assumiamo di essere su di una piattaforma Unix/Linux e di adottare `openssl`. Una volta scaricato `openssl` ed installato sulla nostra macchina, creiamo un utente normale **miaCA**, con home-directory `/miaCA`. Possiamo infatti utilizzare `openssl` per tutte le operazioni richieste come un qualsiasi utente della macchina, e non come l'amministratore.

1. Per prima cosa dobbiamo creare la Certification Authority. Pertanto come l'utente **miaCA** nella cartella `/miaCA/` copiamo i file `openssl.cnf` e `CA.pl`, il primo è il file di configurazione di `openssl` che utilizzeremo per la nostra CA, il secondo è un programma in Perl che ci faciliterà la gestione. Per prima cosa creiamo la cartella `CA_Priv` ove metteremo tutti i dati privati della CA e modifichiamo i seguenti parametri del file `/miaCA/openssl.cnf`:

```
dir               = /miaCA/CA_Priv      # Where everything is kept
default_days      = 3650                # how long to certify for
default_bits      = 2048
nsCaRevocationUrl = http://www.domain.com/ca.crl
```

In questo modo il certificato per la nostra CA durerà 10 anni, avrà 2048 bit e l'elenco dei certificati revocati sarà all'URL http://www.domain.com/ca.crl. Modifichiamo ora `CA.pl`: per prima cosa verifichiamo il PATH a Perl, se è ad esempio `/usr/bin/perl` o `/usr/local/bin/perl` o altro, poi facciamo le seguenti modifiche:

```
$SSLEAY_CONFIG="-config /miaCA/openssl.cnf";
$DAYS="-days 3650";
$CATOP="/miaCA/CA_Priv";
```

A questo punto non ci rimane che generare la Certification Authority (se facciamo qualche errore siamo ancora in tempo a cancellare tutto e ricominciare da capo) dando il comando:

```
/miaCA/CA.pl -newca
```

Questo comando ci chiederà i dati identificativi della CA e, molto importante, la password con la quale cifrare la chiave segreta della CA. La password della chiave segreta va tenuta con molta cura perché è l'ultima difesa della CA contro utilizzi impropri dei suoi certificati.[1] La chiave segreta si trova nel file `/miaCA/CA_Priv/private/cakey.pem` e quella pubblica nel file `/miaCA/CA_Priv/cacert.pem`. Si verifichi che i file contengono le chiavi suddette guardando gli header all'interno dei file e dando il comando:

1 Sarebbe meglio utilizzare una SmartCard per tenere le chiavi su di un hardware indipendente e progettato appositamente.

```
openssl x509 -in /miaCA/CA_Priv/cacert.pem -text
```

Creiamo ora in `/miaCA/` due nuove cartelle: `Requests` per le richieste di certificati in arrivo, e `Revoked` per i certificati revocati. Creiamo anche una lista vuota di CRL, ovvero certificati revocati, con il comando

```
openssl ca -config /miaCA/openssl.cnf -gencrl -out \
    /miaCA/CA_Priv/crl/ca.crl
```

(possiamo controllare i contenuti del file con il comando `openssl crl -in /miaCA/CA_Priv/crl/ca.crl -text`) e pubblichiamolo sul nostro sito all'URL specificato precedentemente. Ogni volta che revocheremo un certificato, dovremo ripetere questo comando e pubblicare sul sito Web il nuovo file.
Come ultima cosa, modifichiamo di nuovo i file `openssl.cnf` e `CA.pl` riducendo ad esempio il `default_days` e `$DAYS` a 365 ed il `default_bits` a 1024.

2. Uno dei principali utilizzi dei certificati digitali è quello di autenticare i siti Web. Per questo è molto utile caricare nel proprio browser il certificato pubblico della nostra CA. Per far ciò possiamo procedere come segue: diamo il comando

```
openssl x509 -in /miaCA/CA_Priv/cacert.pem -out /miaCA/ca-cert.crt
```

Copiamo poi il certificato pubblico sul nostro sito Web alla pagina http://www.domain.com/ca-cert.crt ed aggiungiamo il tipo MIME `application/x-x509-ca-cert` ai file di tipo `.crt` nel nostro server http (in apache il comando è `AddType application/x-x509-ca-cert .crt`). Con Netscape e Mozilla basta scaricare il file per avviare la procedura di registrazione di una nuova CA, con IE bisogna scaricare il file sul disco e poi cliccarlo due volte. In questo modo la nostra CA è aggiunta alla lista di quelle già conosciute dai browser.

3. Supponiamo ora di ricevere una richiesta di certificato da firmare (vedremo fra breve come generare la richiesta), poniamo il certificato contenente la richiesta, cioè la chiave pubblica da firmare, nel file `/miaCA/Requests/newreq.pem`, e diamo i comandi:

```
cd /miaCA/Requests
../CA.pl -sign
```

Prima della firma ci verrà chiesta la password per sbloccare la chiave privata della CA e poi il certificato firmato verrà creato in `newcert.pem`. Possiamo sempre visualizzare il certificato con il comando `openssl x509 -in newcert.pem -text.` Conviene a questo punto dare un nome diverso ed utile ai due file (ad esempio `req-user-2002-01-01.pem` e `cert-user-2002-01-01.pem`) ed inviare il certificato firmato al richiedente. Si noti che in `/`

`miaCA/CA_Priv/newcerts/` vi è una copia di tutti i certificati emessi e che nel file `/miaCA/CA_Priv/index.txt` vi è l'elenco di tutti i certificati con il loro stato, ad esempio Valido o Revocato.

Per revocare un certificato dobbiamo identificare il certificato firmato, come detto ne dovremmo avere ben due copie, e dare il comando

```
openssl ca -config /miaCA/openssl.cnf \
           -revoke /miaCA/Requests/cert-user-2002-01-01.pem
```

Bisogna poi rigenerare la lista dei CRL, come indicato precedentemente, e spostare il certificato annullato nella cartella `/miaCA/Revoked`.

4. L'ultima cosa che ci rimane da fare è di generare le richieste dei certificati degli utenti da inviare alla nostra CA. Possiamo fare questo con `openssl` in modo molto semplice dando il comando:

```
CA.pl -newreq
```

Questo comando dopo averci chiesto le informazioni necessarie, inclusa la password con cui proteggere la chiave privata, genera il file `newreq.pem` contenente però sia la chiave pubblica, che deve essere firmata dalla CA, che quella privata. La cosa più conveniente è creare le richieste per i certificati manualmente, magari anche come un altro utente utilizzando un altra copia del file `openssl.cnf`. Ricordiamo che vi è assoluta indipendenza tra la CA e chi invia alla firma il proprio certificato, pertanto è del tutto consigliato dividere le due fasi usando due utenti diversi, anche se alla fine è la stessa persona fisica a fare entrambe le operazioni. Manualmente possiamo dare i comandi:

```
openssl genrsa -des3 1024 > cert.priv.key
openssl req -config /User/openssl.cnf -new -key cert.priv.key \
          -out newreq.pem
```

Il primo comando crea la chiave privata a 1024bit, di cui chiede la password per cifrarla con 3Des, il secondo comando crea il certificato pubblico da inviare alla CA nel file `newreq.pem` nel quale questa volta non è presente la chiave privata e quindi è direttamente inviabile alla CA. Infine, ad esempio per un server Web, è alle volte necessario rimuovere la password dalla chiave privata (ma in questo caso bisogna proteggere molto bene il file della chiave privata), questo è possibile con il comando:

```
openssl rsa -in cert.priv.key -out cert.priv.nopasswd.key
```

Ovviamente abbiamo descritto i comandi essenziali alla gestione di Certificati Digitali tramite una CA, molti argomenti quale quello della gestione on-line e verifica in real-time dei certificati sono importanti per molte realizzazioni di PKI, ma richiedono delle infrastrutture molto più estese e

complicate.

Andrea Pasquinucci
Libero Professionista in Sicurezza Informatica
pasquinucci@ucci.it

Riferimenti Bibliografici:

[1] Openssl e la sua documentazione si possono ottenere dal sito:
http://www.openssl.org/

[2] Una implementazione Open-Source di CA è data dal progetto OpenCA,
http://www.openca.org/

[3] Una guida on-line a PKI e CA è reperibile all'indirizzo:
http://ospkibook.sourceforge.net/

[4] Working Group on Public-Key Infrastructure (X.509) (pkix),
http://www.ietf.org/html.charters/pkix-charter.html

[5] Public-Key Cryptography Standards, http://www.rsasecurity.com/rsalabs/pkcs/

[6] Utili collezioni di risorse: http://www.pki-page.org/,
http://www.dfn-pca.de/bibliothek/standards/ietf/