Progetto del corso di Sicurezza

autore: Alberto De Bortoli

14 agosto 2009

A.A. 2008-'09

Prof. Dante Baldan

Abstract

Il progetto richiesto prevede la creazione di una PKI (Public Key Infrastructure) minimale da sviluppare con la libreria CryptLib di Peter Gutmann. Tale PKI dovrà permettere la richiesta, l'aggiornamento, la revoca e la verifica di certificati.

1 Tecnologie utilizzate

Il progetto è stato sviluppato su sistema operativo Mac OS X 10.5.7, utilizzando l'IDE NetBeans v.6.7 e il linguaggio di programmazione Java. E' stato necessario inoltre l'uso di un database MySQL v.5.

Lo sviluppo su sistema operativo Mac OS è stato fortemente voluto in quanto è il sistema operativo che utilizzo principalmente. La scelta del linguaggio Java è dovuta invece al fatto che non è un linguaggio che è stato molto approfondito nel mio percorso di studi e ho pensato che questa fosse l'occasione giusta per prendere dimestichezza sia con il linguaggio ma soprattutto con l'IDE. In un primo periodo avevo pensato di sviluppare il progetto sotto ambiente Windows con l'IDE Visual Studio e il linguaggio C# in quanto erano tecnologie che avevo già affrontato abbondantemente in un periodo di stage presso un'azienda. Vi è, inoltre, un altro motivo, ovvero la difficoltà riscontrata nella compilazione della libreria cryptlib e nel suo utilizzo con il linguaggio Java, cosa tutt'altro che facile e non esplicata nel manuale della libreria.

La procedura ardua per riuscire a lavorare con la tecnologia scelta è di seguito riassunta:

- scaricare la libreria e decomprimerla da terminale per preservarne il formato UNIX.
- aprire il file "misc/config.h" e decommentare la riga "#define USE_JAVA"
- aprire il file "bindings/java_jni.c" e modificare la riga "#include <jni.h>" in "#include "/System/Library/Frameworks/JavaVM.framework/Versions /1.6.0/Headers/jni.h""

- eseguire la compilazione e spostare il file "libcl3.3.2.dylib" risultato della compilazione nella cartella "/usr/local/lib"
- aprire il file di profilo del sistema con "nano ~/.profile" e settare il valore della variabile "LD_LIBRARY_PATH" a "/usr/local/lib" e includere la libreria compilata nel progetto.

E' stato necessario inoltre utilizzare JDBC connector per effettuare la connessione al database dal linguaggio Java. Reperire le azioni adatte per impostare il sistema adeguatamente non è stato immediato e sono servite molte ricerche web e confronti con colleghi per trovare il modo corretto di operare.

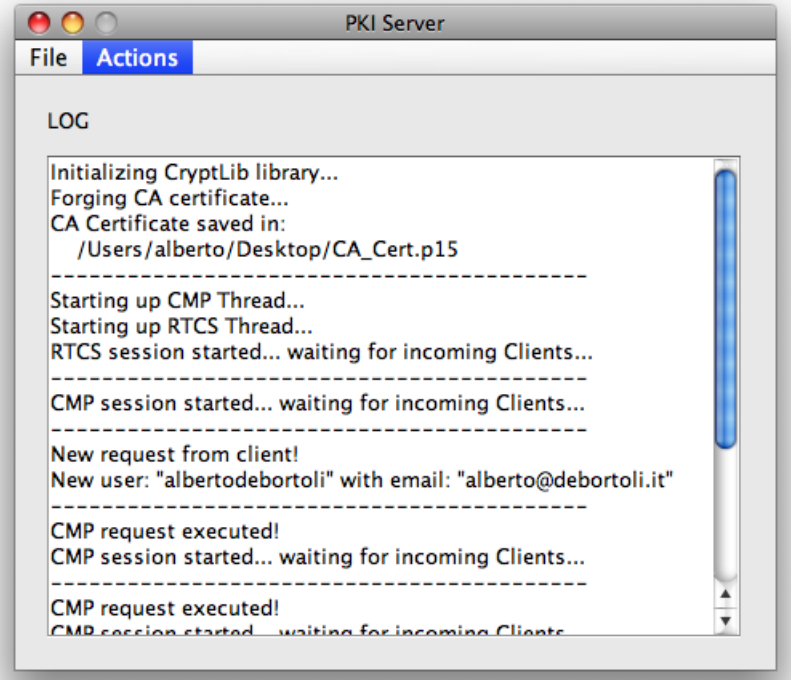
Un altro punto a sfavore è stato il manuale di CryptLib, purtroppo a mio parere non l'ho trovato ben scritto e gli esempi di codice non sono mai stati ben pensati. E' servito molto tempo solo per capire come riuscire a compilare anche le più semplici righe di codice di chiamate a funzioni della libreria, fatto che ha rallentato notevolmente la fase di start-up del progetto.

2 Descrizione

Il progetto nel suo insieme è diviso in 2 parti: la parte Client (PKI Client) e la parte Server (PKI Server). Entrambe sono Desktop Application eseguibili su Java Virtual Machine.

2.1 PKI Server

Il Server permette di avviare la sessione server che consta di 3 thread. Un primo thread serve per accettare richieste CMP (Certificate Management Protocol) sulla porta 2042 (di default), il secondo per accettare richieste RTCS (Real Time Certificate Status) sulla porta 204 (di default), mentre il terzo per attendere richieste di creazione di account sulla porta 2222 (di default). Qualsiasi azione e richiesta viene segnalata sull'area di log. Tramite GUI è possibile scegliere su quali porte mettere in ascolto i thread, stessa cosa avverrà per il client. Dal menù "Actions" è possibile avviare e terminare il server.



La scelta di RTCS in particolare si è preferita perché a differenza di OCSP permette di ottenere risultati "freschi" sulla validità dei certificati e non fa uso di CRL. All'avvio del server viene forgiato un certificate self-signed per la CA e per semplicità viene salvato sul Desktop col nome "CA_Cert.p15", viene creata una cartella con il nome "CA_Users" all'interno della quale verranno salvati dei file, ognuno per ogni utente. Tale cartella fungerà da database degli utenti. I file che verranno creati avranno il nome "[user_email].txt" e al loro interno conterranno 5 righe di testo in chiaro corrispondenti a:

1. e-mail
2. common name
3. username generato dalla CA
4. initialization password generata dalla CA
5. revocation password generata dalla CA

Con i dati di default dell'applicazione Client, l'account viene creato nel file "alberto@debortoli.it.txt" contenente le seguenti 5 righe:

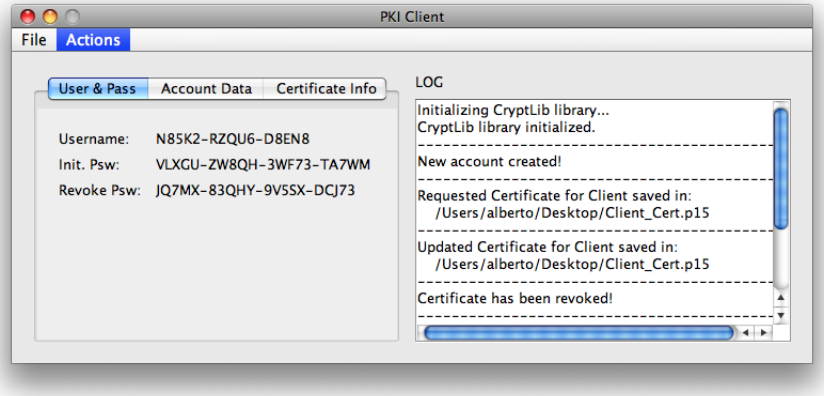
```
alberto@debortoli.it
```

```
albertodebortoli
N85K2-RZQU6-D8EN8
VLXGU-ZW8QH-3WF73-TA7WM
JQ7MX-83QHY-9V5SX-DCJ73
```

Il "certificate store" è gestito interamente da cryptlib ma ha bisogno di un database preesistente. E' stato utilizzato MySql con un DSN (Data Source Name) denominato "CA_DSN" collegato a un database chiamato "CA_DB". Esso è acceduto tramite ODBC ed è utilizzato da criptlib per salvare i certificati.

2.2 PKI Client

Il Client permette di creare un account con i dati desiderati per l'utente, richiedere successivamente un certificato al server, aggiornarlo, revocarlo e verificare che un certificato sia valido. Qualsiasi azione e richiesta è segnalata sull'area di log. Dal menù "File" è possibile richiedere al server la creazione di un account, solo successivamente le voci per la richiesta, l'aggiornamento, la revoca e la verifica risultano attive nel menù "Actions".

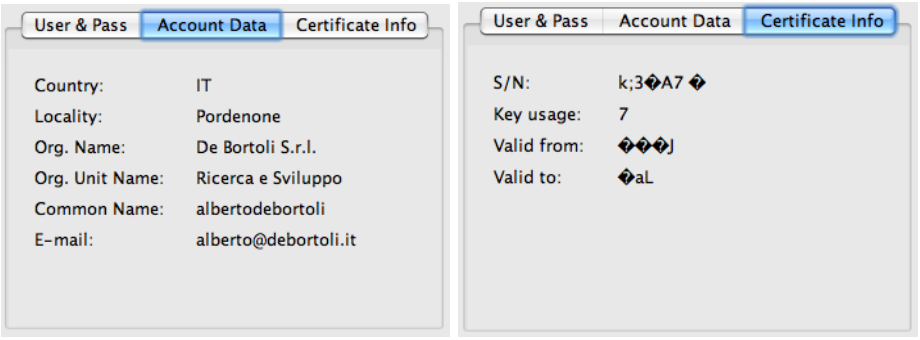


Dopo che è stato richiesto di creare un nuovo account, vengono visualizzate le informazioni relative nella scheda "Account Data" e nella scheda "User & Pass" vengono visualizzate le informazioni che la CA ha creato per l'utente. Tali informazioni vengono salvate dal server su disco e da quel percorso il server le reperisce per fornirle al client.

Dopo che è stato richiesto un certificato le informazioni ad esso pertinenti vengono visualizzate nella scheda "Certificate Info". Purtroppo non sono in un formato intelligibile ma non è ritenuto un particolare rilevante.

2.3 Dettagli

Seppur il progetto è di modeste dimensioni, è stato realizzato cercando di scrivere del codice il più pulito e manutenibile possibile, secondo i dettami



dell'ingegneria del software e seguendo uno stile di programmazione quanto più elegante possibile appreso in ambito accademico.

Tutte le richieste sono parametrizzabili dall'utente per quanto riguarda l'indirizzo IP e la porta verso cui inoltrare la richiesta o avviare l'ascolto di un thread. Il sistema è stato testato su una rete locale con più client su più macchine con esito positivo.