ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

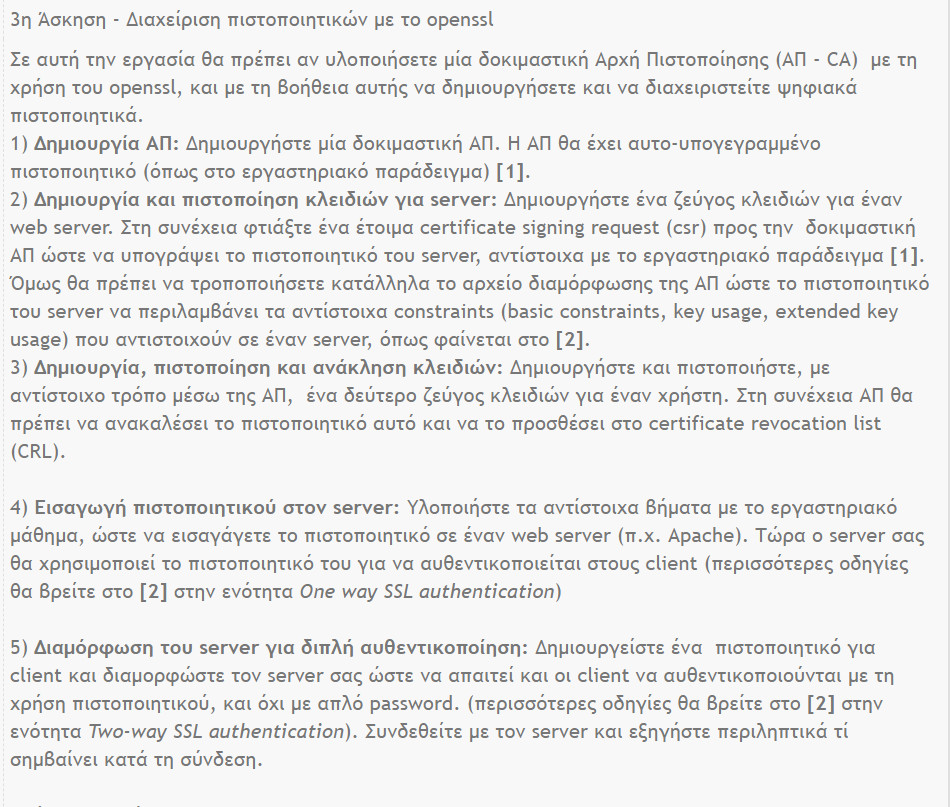
Τμήμα Πληροφορικής



Εργασία Μαθήματος ***Ασφάλεια Πληροφοριακών Συστημάτων***

|  |  |
| --- | --- |
| Άσκηση *<<αριθμός άσκησης>>* | 3η Άσκηση - Διαχείριση πιστοποιητικών με το openssl |
| Όνομα φοιτητών Αριθμός μητρώων | Μπαρμπίνη Χρυσούλα Π15094 |
| Ημερομηνία παράδοσης | 20/11/2018 |

**Εκφώνηση της Άσκησης**



**Περιεχόμενα**

1. Δημιουργία ΑΠ……………………………………………………………………………………………
2. Δημιουργία και πιστοποίηση κλειδιών για server…………………………………….…
3. Δημιουργία πιστοποίηση και ανάκληση κλειδιών……………….………………………
4. Εισαγωγή πιστοποιητικού για server……………………………………………………………
5. Διαμόρφωση του server για διπλή αυθεντικοποίηση……..……………………………

**1) Δημιουργία Αρχής Πιστοποίησης**

Για την δοκιμαστική αρχή πιστοποίησης δημιουργούμε ένα φάκελο **~ /lab**  ο οποίος περιλαμβάνει τους παρακάτω καταλόγους :

i) **~/lab** – Βασικός κατάλογος για την Α.Π.

**ii)** **~ /lab/certs** – Περιέχει τα πιστοποιητικά που έχει εκδώσει η Α.Π.

**iii)** **~/lab/crl** – Λίστα ανάκλησης πιστοποιητικών

**iv)** **~/lab/newcerts** – Νέα πιστοποιητικά

**v)** **~/lab/private** – Ιδιωτικά κλειδιά για την Α.Π.

**vi)** **~/lab/public** –Περιλαμβάνει τα δημόσια κλειδιά

**vii)** **~ /lab/req** – Αιτήματα πιστοποιητικών προς υπογραφή(csr)

**viii)** **~/lab/CAcnf.txt** – Αρχείο διαμόρφωσης λειτουργίας Α.Π.

**iix)** **~/lab/servercnf.txt** – Αρχείο διαμόρφωσης για έκδοση πιστοποιητικών server

**x) ~/lab/usercnf.txt** – Αρχείο διαμόρφωσης για έκδοση πιστοποιητικών χρήστη

* Δημιουργία του **self-signed** κλειδιού της Αρχής Πιστοποίησης με την εντολή:

**openssl req -new -x509 -keyout private/CAkey.pem -out certs/CAcert.pem -days 365 -config CAcnf.txt -sha1**

Εδώ γίνεται αναφορά στο αρχείο διαμόφωσης CAcnf.txt που μπορεί να κληθεί bydefault με την εντολή “ca” όπως θα δούμε και παρακάτω ενώ περιέχει διάφορους περιορισμούς και παραμέτρους που αφορούν την πολιτική έκδοσης πιστοποιητικών αλλά και διάφορους περιορισμούς όπως οι ημέρες ισχύς του και ο αλγόριθμος md που θα χρησιμοποιηθεί ενώ μπορεί να τροποποιηθούν δίνοντας τα κατάλληλα ορίσματα όπως στην παραπάνω εντολή. Για παράδειγμα , στο CAcnf αναφέρεται ότι μπορεί να ισχύσει το εν λόγω πιστοποιητικό για 1825 ημέρες ενώ εγώ το έχω ορίσει για 365 ημέρες και ούτω καθεξής .

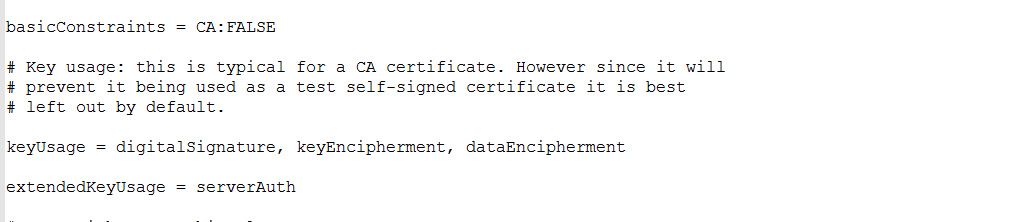
**2) Δημιουργία και πιστοποίηση κλειδιών για Server**

* Δημιουργία **ιδιωτικού κλειδιού του server** με την εντολή

**openssl genrsa -out private/serverKey.pem 2048**

* Aίτημα προς υπογραφή από την Αρχή πιστοποίησης με την εντολή

Για το αίτημα προς υπογραφή έχω δημιουργήσει το αρχείο διαμόρφωσης **servercnf.txt** το αίτημα του server .



**openssl req -new -config servercnf.txt -nodes -keyout private/serverKey.pem -out req/server.csr -days 365 -md5**

* Έκδοση πιστοποιητικού του server από την Αρχή Πιστοποίησης με την εντολή

**openssl ca -config CAcnf.txt -days 365 -policy policy\_anything -cert certs/CAcert.pem -keyfile private/CAkey.pem -out certs/serverCert.pem -infiles req/server.csr**

**3. Δημιουργία , πιστοποίηση και ανάκλησης κλειδιών**

* Δημιουργία **ιδιωτικού κλειδιού του client** με την εντολή

**openssl genrsa -out private/client.pem 2048**

* Aίτημα προς υπογραφή από την Αρχή πιστοποίησης με την εντολή

Για το αίτημα προς υπογραφή έχω δημιουργήσει το αρχείο διαμόρφωσης **usercnf.txt** το αίτημα του client .

**openssl req -new -config usercnf.txt -nodes -keyout private/clientKey.pem -out req/client.csr -days 365 -md5**

* Έκδοση πιστοποιητικού του client από την Αρχή Πιστοποίησης με την εντολή

**openssl ca -config CAcnf.txt -days 365 -policy policy\_anything -cert certs/CAcert.pem -keyfile private/CAkey.pem -out certs/clientCert.pem -infiles req/client.csr**

**Σημείωση :** Όσον αφορά την πολιτική έκδοσης πιστοποιητικού επέλεξα να ακολουθήσω τα παραδείγματα του εργαστηρίου . Βέβαια , σε ρεαλιστικά δεδομένα καλό θα ήταν να ακολουθούσαμε μια πιο αυστηρή πολιτική όπως η policy match που και αυτή αναφέρεται στο εκάστοτε αρχείο διαμόρφωσης .

* Δημιουργία της λίστας ανάκλησης πιστοποιητικών (CRL certificate revocation list) με την εντολή :

**openssl ca -gencrl -out crl/crl1.pem -config CAcnf.txt**

* Ανάκληση του πιστοποιητικού του client με την εντολή :

**openssl ca -gencrl -revoke certs/clientCert.pem -config CAcnf.txt**

Και παρατηρώντας στο index.txt την πρώτη στήλη που αναγράφει R δηλαδή ότι έχει ανακληθεί το συγκεκριμένο πιστοποιητικό

* Επαναδημιουργία της CRL(certificate revocation list) η οποία τώρα αναγράφει και το πιστοποιητικό που ανακλήθηκε με την εντολή

**openssl ca -gencrl -out crl/client.crl -config CAcnf.txt**

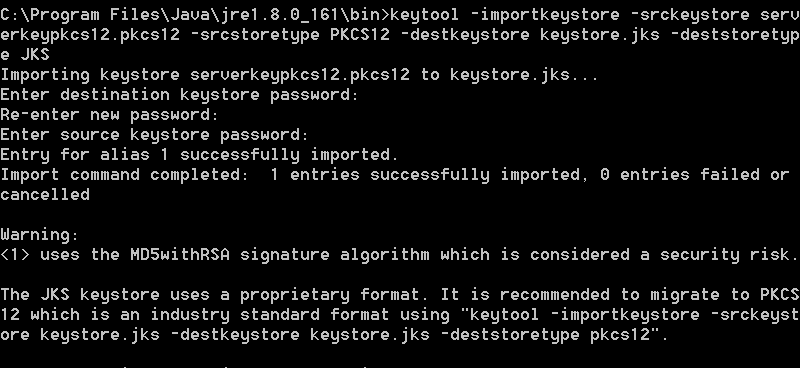
**4) Εισαγωγή πιστοποιητικού στον server (**One way ssl authentication**)**

Σε μια one-way SSL authentication ο client μπορεί να επιβεβαιώσει την ταυτότητα του server ενώ δεν μπορεί να γίνει το αντίθετο. Εκτελώντας τα βήματα 1 και 2 της άσκησης έχουμε δημιουργήσει μια αρχή πιστοποίησης καθώς και ένα ζεύγος κλειδιών για τον server τα οποία είναι υπογεγραμμένα από την root CA που δημιουργήσαμε . Για να πραγματοποιήσουμε one-way ssl στον tomcat server της εφαρμογής μου πρέπει το κλειδί που έχω δημιουργήσει να είναι με το κατάλληλο φορματ. Συγκεκριμένα , ο tomcat αυτή την στιγμή διαχειρίζεται κλειδιά που υπακούν σε JKS, PKCS11 ή PKCS12 . Οπότε με την παρακάτω εντολή δημιουργούμε ένα key store σε pkcs12 format.

**openssl pkcs12 -export -out serverkeypkcs12.p12 -in certs/serverCert.pem -inkey private/serverKey.pem**

Στην συνέχεια , χρησιμοποιώντας το εργαλείο της java keytool φτιάχνουμε ένα keystore δηλαδή ένα αρχείο jks για να μπορεί να το διαχειριστεί ο server μας.

**keytool -importkeystore -srckeystore serverkeystore.p12 -srcstoretype PKCS12 -destkeystore serverkeyStore.jks -deststoretype JKS**



Στην συνέχεια πρέπει ο tomcat να χρησιμοποιεί αυτό το keystore και να ενεργοποιήσει το HTTPS .Θα τροποποιήσουμε το αρχείο **TOMCAT\_HOME/config/server.xml**

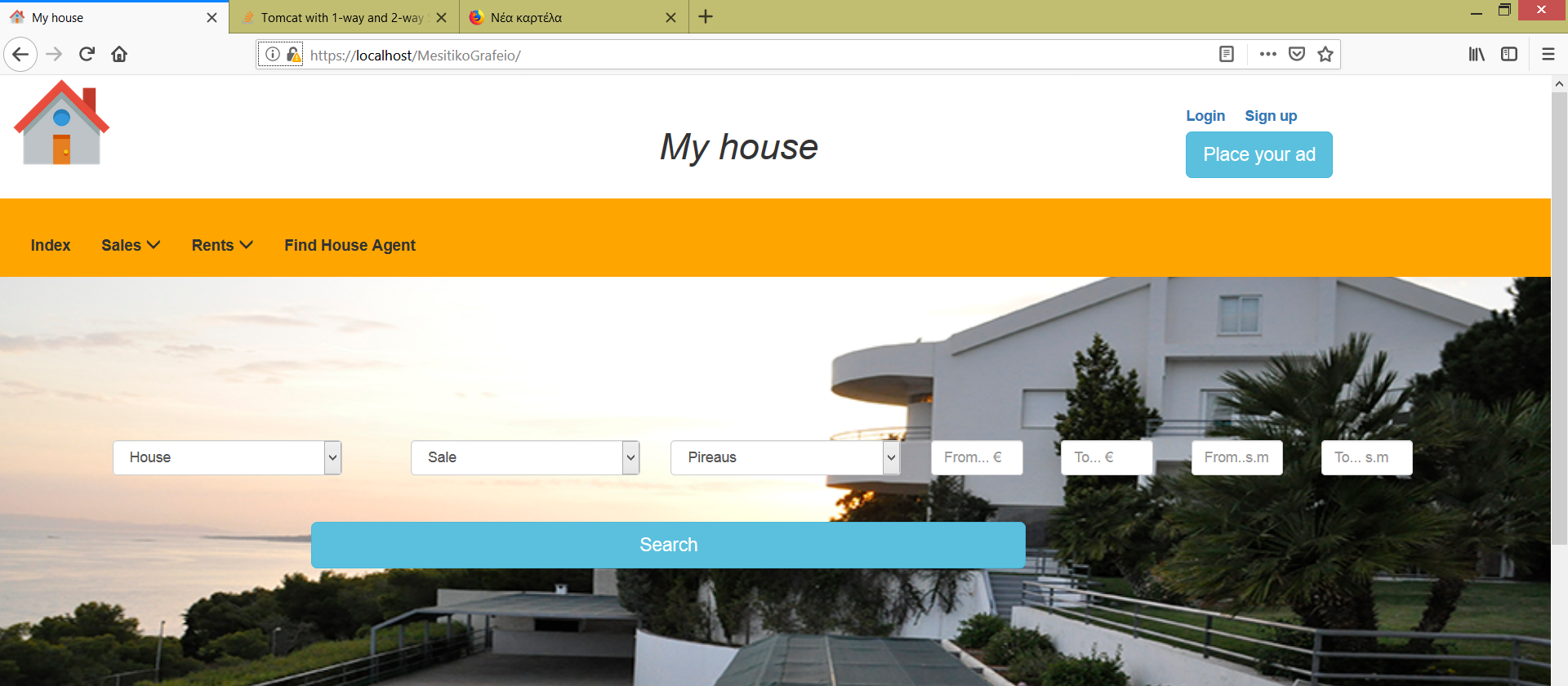
<Connector port=*"443"* protocol=*"org.apache.coyote.http11.Http11NioProtocol"*

maxThreads=*"150"* SSLEnabled=*"true"* scheme=*"https"* secure=*"true"*

keystoreFile=*"C:\Program Files\Java\jre1.8.0\_161\bin\serverkeyStore.jks"*

keystorePass=*"TESTIT"*

clientAuth=*"false"* sslProtocol=*"TLS"*>

</Connector>

**5) Διαμόρφωση του server για διπλή αυθεντικοποίηση(**Two way ssl authentication)

Σε μια two-way SSL authentication υπάρχει αμοιβαία επιβεβαίωση ταυτότητας τόσο από τον server όσο και από τον client . Και οι δυο έχουν τα αντίστοιχα ζεύγη κλειδιών υπογεγραμμένα από την αρχή πιστοποίησης CA. Εκτελώντας , λοιπόν , τα παραπάνω βήματα και ακολουθώντας την ίδια διαδικασία για το κλειδί του client δημιουργούμε ένα trustore για τον client.

**openssl pkcs12 -export -out clienttrustore.p12 -in clientCert.pem -inkey clientKey.pem**

**keytool -import -alias mockdis -keystore cTrustore.jks -file clientCert.pem**

Tροποποιούμε το αρχείο **TOMCAT\_HOME/config/tomcat-users.xml** συμπληρώνοντας τον παρακάτω κώδικα .

<role rolename=*"tomcat"*/>

<role rolename=*"admin"*/>

<role rolename=*"user"*/>

<user username=*"CN=Chrysa Barbini,OU=Lab test client,O=Unipi, L=Athens, ST=ATTICA, C=GR"* password=*""* roles=*"user"* />

**Tροποποιούμε το αρχείο TOMCAT\_HOME/config/server.xml συμπληρώνοντας τον παρακάτω κώδικα .**

<Connector port=*"443"* protocol=*"org.apache.coyote.http11.Http11NioProtocol"*

maxThreads=*"150"* SSLEnabled=*"true"* scheme=*"https"* secure=*"true"*

keystoreFile=*"C:\Program Files\Java\jre1.8.0\_161\bin\serverkeyStore.jks"*

keystorePass=*"TESTIT"*

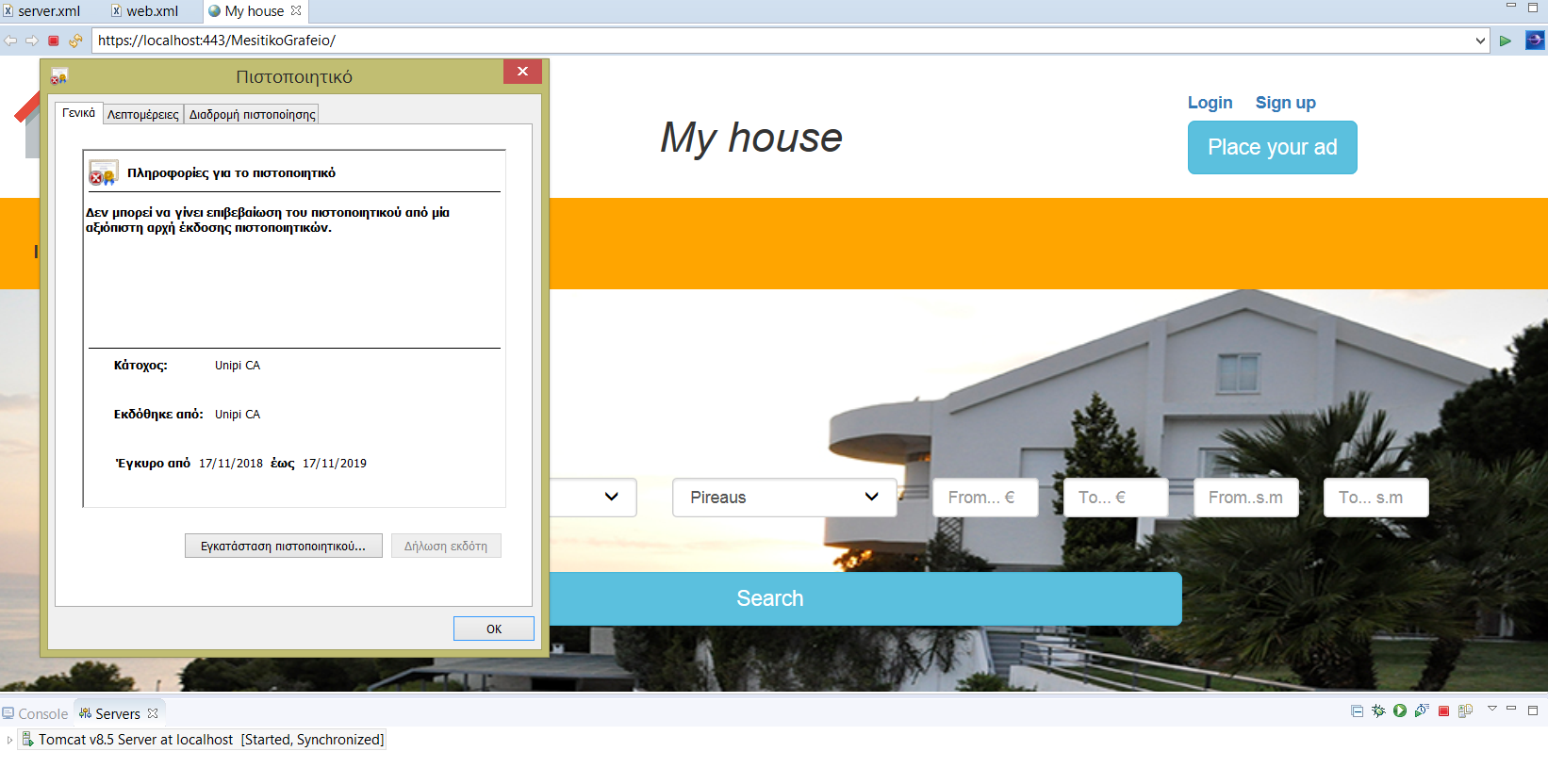
truststoreFile=*"C:\Program Files\Java\jre1.8.0\_161\bin\cTrustore.jks"*

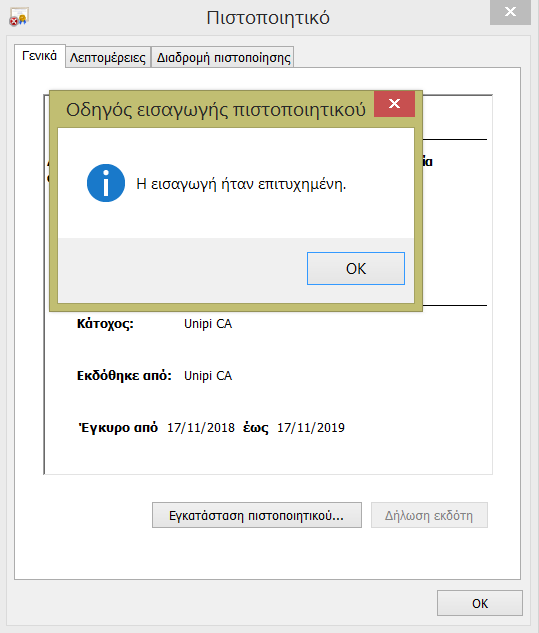
truststorePass=*"TESTIT"*

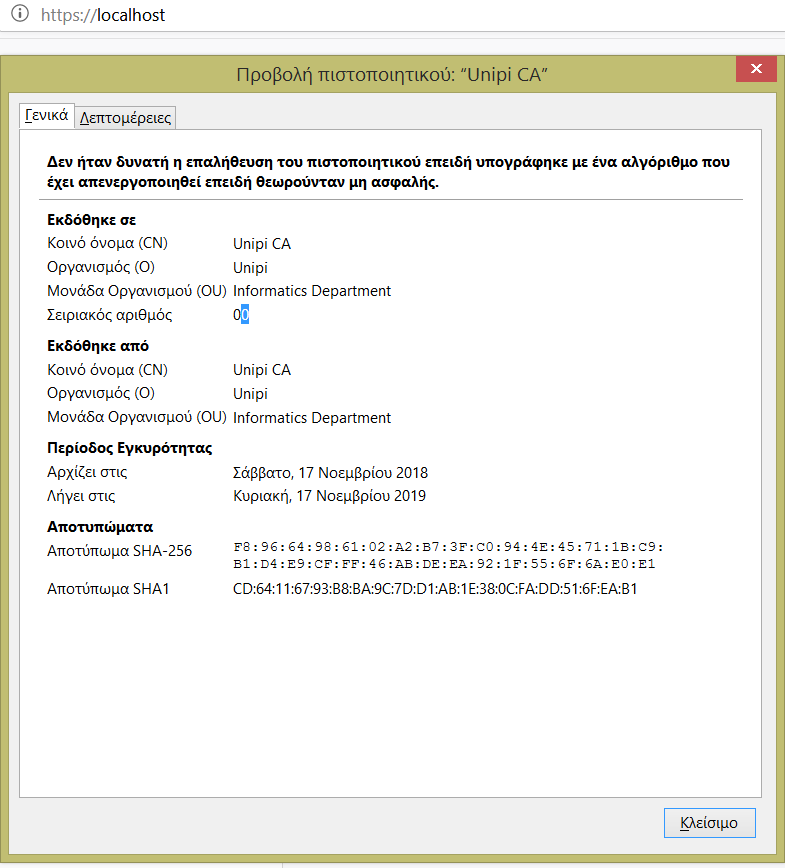
clientAuth=*"true"* sslProtocol=*"TLS"*>

</Connector>

Προσθέτουμε το πιστοποιητικό του χρήστη στον firefox .







**Σημείωση :** Στον ίδιο φάκελο περιλαμβάνεται και ο φάκελος lab που περιέχει τα παραπάνω ζεύγη κλειδιών , πιστοποιητικά καθώς και τις λίστες ανάκλησης.