TEMA DE CASĂ #3 – IMPLEMENTAREA UNUI SERVICIU SECURIZAT DE GESTIUNE A DOCUMENTELOR

RESPONSABIL DE TEMĂ: CIPRIAN DOBRE (CIPRIAN.DOBRE@CS.PUB.RO)

DATA PUBLICĂRII: 25.11.2012

TERMENUL DE PREDARE: 09.12.2012

OBIECTIVE

După realizarea acestei teme de casă studentul va fi capabil să:

- Realizeze o aplicaţie bazată pe folosirea unor soluţii de securitate în Java.
- Gestioneze eficient structuri de date folosite în rezolvarea unor probleme similare.
- Realizeze programe ce folosesc transmisii sigure de mesaje, autentificare, criptare.

Cunoştinţele necesare rezolvării acestei teme de casă:

- Programare folosind Java
- Cunoştinţe privind mecanismele de securitate puse la dispoziţie de Java pentru securitate.
- Detaliile de folosire ale claselor din pachetele Java pentru securitate prezentate în cadrul laboratorului.

ENUNŢUL PROBLEMEI

Se cere implementarea unui sistem de gestiune a documentelor. Sistemul este format din aplicaţii client, o aplicaţie server şi un Serviciu de Autorizare.

Presupunem că există o instituţie ce pune la dispoziţia angajaţilor săi un sistem de gestiune a documentelor. Compania este formată din mai multe departamente (Contabilitate, Resurse Umane, IT, etc.). În cadrul sistemului se consideră că un utilizator este identificat unic prin intermediul unui certificat X.509 eliberat de Autoritatea de Certificare din cadrul companiei şi care are numele unic (DN) de forma :

CN=IonPopescu, OU=Accounting, O=MyCompany, L=Bucharest, ST=Romania, C=RO

Presupunem că există un server central de gestiune a documentelor ce deserveşte angajaţii companiei. Aplicaţia server acceptă doar conexiuni SSL şi utilizează un certificat propriu semnat de CA-ul companiei. Pentru început clientul se conectează la unul dintre servere folosind o conexiune SSL cu autentificare bidirecţională (atât serverul cât şi clientul îşi trimit certificatele). Handshake-ul reuşeşte numai dacă certificatul prezentat de client este semnat de către CA. În cadrul sistemului clienţii sunt identificaţi prin numele câmpului CN din certificatul prezentat.

În sistem clienţii au rolul de a citi documente şi de a încărca pe server noi documente. În aplicaţia server pentru fiecare document veţi păstra o informaţie referitoare la identitatea proprietarului, a clientului ce a uploadat respectivul document, şi la identitatea

departamentului de care acesta aparţine. Pentru ca această informaţie să nu se piardă nici după repornirea serverului veţi folosi un fişier. În acest fişier veţi scrie mapări (nume_fişier, identitate_proprietar, departament), fişierul fiind întotdeauna ţinut criptat pentru a face imposibilă divulgarea informaţiei (metoda de criptare o alegeţi voi, însă trebuie să o amintiţi în fişierul Readme). Pentru departament se va folosi de câmpul OU ce desemnează departamentul din care face parte utilizatorulul proprietar. La pornirea aplicaţiei server veţi citi conţinutul fişierului în memorie.

Un client va încerca iniţial stabilirea conexiunii cu serverul de gestiune a documentelor. Dacă conexiunea reuşeşte, acesta va intra într-un mod interactiv în care poate primi comenzi din partea utilizatorului (implementaţi primirea de comenzi fie din consola text, fie o interfaţă grafică minimală). Comenzile posibile sunt: list, download, upload.

Comanda list va întoarce lista de fişiere existente în aplicaţia server. Comanda upload va avea ca rezultat încărcarea unui fişier pe server. Comanda download are rolul de a indica dorinţa începerii unui download (primeşte ca parametru numele unui fişier). Pentru download aplicaţia server va „autoriza” mai întâi clientul. Aplicaţia server va stabili o conexiune SSL cu Serviciul de Autorizare (serviciul central) şi pe această conexiune aplicaţia interoghează Serviciul pentru a afla dacă DN-ul clientului este autorizat sau nu să downloadeze documentul aparţinând unui anumit departament. Fiecare departament al companiei are ataşată o anumită prioritate pe baza căreia un utilizator are acces sau nu pe un server de documente. Un utilizator aparţinând unui anumit departament nu are access decât la fişierele departamentelor având o prioritate mai mică sau egală. De exemplu departamentul Management are o prioritate maximă, aceasta însemnând că un utilizator din acest departament va avea acces la documentele tuturor celorlalţi utilizatori. Pe baza politicii pe care o veţi defini voi Serviciul de Autorizare va trimite serverului un răspuns prin care va informa dacă respectivul utilizator are drept sau nu de download.

Dacă răspunsul este favorabil, comanda download va avea ca rezultat transferarea fişierului şi salvarea acestuia local, la client, într-un director numit „download”. Tot aplicaţia server va avea şi rolul de a monitoriza fişierele curent uploadate. Dacă aplicaţia server întâlneşte de exemplu „bomba” sau „greva” în numele unui fişier (lista de cuvinte cheie o definiţi voi) atunci va "bloca" utilizatorul respectiv. Pentru aceasta numele de utilizator va fi trimis Serviciului de Autorizare care il va scrie într-un fişier de utilizatori "banned" într-o formă încriptată cu formatul "nume time". "Time" înseamnă momentul de timp când a fost scrisă respectiva intrare. Un utilizator blocat va rămâne blocat pentru un timp maxim prestabilit de voi, timp în care el nu are voie să mai downloadeze de la server nici un document (în pasul de verificare a departamentului veţi întoarce răspuns negativ dacă respectivul client este în lista de banned). Dacă Serviciul de Autorizare este repornit el va reciti datele despre utilizatorii blocaţi din respectivul fişier.

PRECIZĂRI LEGATE DE IMPLEMENTAREA TEMEI

Aplicaţia va fi implementată în Java. Pentru implementare folosiţi JSSE şi JCE. În fişierul Readme veţi detalia aspecte legate de implementare, veţi explica deciziile pe care le luaţi asupra algoritmilor de criptare folosiţi.

Q&A

Q: Care sunt entităţile/programele ce trebuie implementate?

A: Vor trebui implementate: un *client* care comunică cu un *server* (conexiune ssl cu autentificare bidirecţională), iar server-ul şi el comunică cu un *serviciu de autorizare* (conexiune ssl). Deci sunt 3 programe diferite ce trebuie implementate.

Q: Se consideră că numele documentelor sunt unice? La download aplicaţia server nu are cum să-şi dea seama la care document se referă clientul dacă există cel puţin două documente cu acelaşi nume stocate pe departamente diferite sau aparţinând unor client diferiţi. Problema se poate rezolva în diverse moduri. Există vreo restricţie în acest sens sau putem să implementăm cum dorim, de exemplu prin a adăuga parametri la comandă?

A: Fişierele sunt ţinute într-ul acelaşi director şi numele vor fi unice. Legat de suprascriere, aveţi mai multe variante: 1) puteţi fie considera varianta în care doar utilizatorii având acces superior pot suprascrie un anumit fişier; 2) puteţi asigna un nume ad-hoc (File.createTempFile???) unui fişier pe care doriţi să îl salvaţi şi să reţineţi intern, în aplicaţia server, o mapare între numele real şi numele creat pentru fişierul salvat.

Q: În cartea de aplicaţii de laborator există exemplul 6, care constituie un bun model asupra funcţionalităţilor ce trebuie implementate şi în cadrul temei (din manual). Noi putem porni de la acel model?

A: Da, puteţi să porniţi de la respectivul exemplu şi să-l folosiţi ca schelet.

Q: Serverul trebuie să comunice cu mai mulţi clienţi în acelaşi timp ?

A: Da, serverul va trebui să comunice cu mai mulţi clienţi în acelaşi timp.

Q: Pentru documente uploadate serverul va tine minte identitatea ownerului si a clientului care face upload. Aceste doua identitati nu sunt identice? Daca nu, atunci cine este owner?

A: Owner-ul inseamna proprietarul documentului, clientul care face upload.

Q: Cum vom compila proiectul? Folosind ant sau make?

A: Folositi ant (mai ales ca e vorba de o tema in Java).

Q: Exista vreun nume pe care trebuie sa le dam claselor principale?

A: Probabil ca veti numi clasa care implementeaza client cu numele Client, pe cea server o veti numi Server (speram ca nu veti numi clasa Client cu numele TheNotServerButTheOppositeFunctionality). Noi nu impunem nici o restrictie privind numele claselor principale.

Q: Cum se genereaza un certificat folosind keytool pentru userul X, dar care sa fie semnat (issued) de autoritatea Y ?

A: In primul rand trebuie sa creati certificatul pentru CA. In al doilea rand atat serverul cat si clientul trebui sa aiba certificate semnate de CA. Pentru generarea de certificate si key pentru CA si pentru client|server puteti urma pasii din documentatia de laborator, cu mentiunea ca fiecare intrare in keystore este indexata dupa un alias. Dupa ce faceti acest lucru ar trebui sa obtineti ceva asemanator cu:

```
station:~/workspace/sprc-tema3-client$ keytool -list -keystore clnKeystore
Enter keystore password:

Keystore type: JKS
Keystore provider: SUN
```

Your keystore contains 3 entries

ca, Dec 1, 2009, trustedCertEntry,
Certificate fingerprint (MD5): 4F:67:C9:CE:92:93:64:A2:4A:64:43:D2:C4:4D:8F:F1
client1, Dec 1, 2009, PrivateKeyEntry,
Certificate fingerprint (MD5): AE:B7:B5:5A:E3:C2:6E:BF:F3:72:16:F4:34:66:27:2C
client1-cert, Dec 1, 2009, trustedCertEntry,
Certificate fingerprint (MD5): 89:A8:11:9D:DE:1E:13:66:18:79:4F:48:BD:6B:CF:FE

station:~/workspace/sprc-tema3-server$ keytool -list -keystore srvKeystore
Enter keystore password:

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 3 entries

ca-cert, Dec 1, 2009, trustedCertEntry,
Certificate fingerprint (MD5): 4F:67:C9:CE:92:93:64:A2:4A:64:43:D2:C4:4D:8F:F1
server, Dec 1, 2009, PrivateKeyEntry,
Certificate fingerprint (MD5): 20:B7:8B:81:60:EE:CC:13:D7:42:A7:8F:5B:47:5C:8D
server-cert, Dec 1, 2009, trustedCertEntry,
Certificate fingerprint (MD5): 57:85:F1:BF:DF:21:4C:61:8C:A7:53:1B:4D:05:7A:AC

Acest sistem functioneaza dar nu cu autentificare -> bad_certificate