



FAKULTET TEHNIČKIH NAUKA



Gavrilo Prodanović, PR158/2015

Kosta Ćurčić, PR60/2015

- Primenjeno softversko inženjerstvo -

Sadržaj

1. OPIS REŠAVANOG PROBLEMA	3
2. TEORIJSKE OSNOVE	4
3. DIZAJN IMPLEMENTIRANOG SISTEMA	6
4. TESTIRANJE SISTEMA	8

1. OPIS REŠAVANOG PROBLEMA

Za potrebe klijenta potrebno je dizajnirati aplikaciju čija namena je bezbedna i sigurna komunikacija između komponenti klijenta i servis provajdera. Kao i implementacija komponente za administraciju RBAC konfiguracije.

Bezbednosne komponente predstavljaju SecureHost, za hostovanje WCF servisa, kao i SecureProxy koji uspostavlja komunikacioni kanala.

SecureHost

SecureHost treba podržavati komunikaciju preko TCP protokola na transportnom nivou, gde takođe treba imati podršku za dva vida autentifikacije: Windows (*Kerberos/NTLM*) i Certificate). Takođe je potrebno da podržava RBAC model autorizacije koji ispunjava sledeće uslove:

Da u slučaju Windows autentifikacije korisnik bude povezan sa privilegijama posredstvom

Da u slučaju Certificate autentifikacije korisnik bude opisan atributima samog sertifikata, sertifikata bude definisan na sledeći način: CN će predstavljati korisničko ime, OU će predstavljati ulogu

Podžavaće zapis bezbednosnih događaja u specifični Windows Event Log. I to tako što svi servisi hostovani preko ove klase na istoj mašini treba da loguju u istu log datoteku, a u zavisnosti od tipa implementirane autentifikacije na servisu menjaće se i ime izvora događaja.

i način izbegavanja ponovnog kreiranja *Iprincipal* objekta pri svakom pozivu metoda servisa, već da se obezbedi in-memory keširanje već postojećih zahteva kao i periodično osvežavanje u slučaju izmena

SecureProxy

treba da ima implementiran način odabira koji od tipova implentiranih autentifikacionih modela će gađati. Takođe da se uspostavi komunikacioni kanal ka servisu odnosno SecureHost-u.

RBAC Admin

nosno implementiraće mehanizam periodičnog ažuriranja pri svakoj izmeni konfiguracije. Time će ažurirati i dobiti nove informacije o korisnicima.

SIEM

SIEM je komponenta koja će biti zadužena za centralizovani nadzor i logovanje bezbednosnih događajagenerisanih od strane svih instanci servisa.

Zajedničke usluge za oba tipa servisa će biti:

CREATE(...) – Pravo imaju svi korisnici sa *Administrate* privilegijom

MODIFY(...) – Pravo imaju svi korisnici sa *Write* privilegijom

READ(...) – Pravo imaju svi korisnici sa *Read* privilegijom

DELETE(...) – Pravo imaju svi korisnici sa *Administrate* privilegijom ili su kresatori fajla

Takođe da bi korisnik uopšte mogao pristupiti metodama servisa mora imati *Access* privilegiju.

2. TEORIJSKE OSNOVE

Kako je problem obezbediti kontrolisan pristup informacionom sistemu i podacima, u tu svrhu implementirani su tri osnovna bezbednosna mehanizma poznata kao „AAA“ (A = *Authentication*, A = *Authorization*, A = *Accounting/Auditing*). U ovom poglavlju će biti razmatratrana teorijska stanovišta koja stoje iza ovih bezbednosnih mehanizama.

Autentifikacija

Autentifikacija je jedan od osnovnih bezbednosnih mehanizama kojim se obezbeđuje validacija identiteta u okviru informacionog sistema. Da bi entitet mogao da pristupi sistemu potrebno je da dokaže da je on upravo onaj za koga se izjašnjava. Podaci kojima se korisnici predstavljaju sistemu i potvrđuju identitet nazivaju se kredencijali i mogu se podeliti u tri kategorije: 1) nešto što korisnik zna (npr. Šifra, ili odgovor), 2) nešto što korisnik ima (npr. Lična karta), 3) nešto što korisnik jeste (npr. otisak prstiju, lice).

Aplikacija podržava dva modela autentifikacije: Windows (*Kerberos/NTLM*) i Certificate (*ChainTrust*).

Windows autentifikacioni model je zasnovan na SPNEGO (*Simple and Protected GSS API negotiation mechanism*) mehanizmu za pregovaranje između različitih realnih autentifikacionih mehanizama u zavisnosti od okruženja. SSPI (*Security Support Provider Interface*) je Windows API koji implementira SPNEGO i predstavlja zajednički interfejs za različite Windows autentifikacione protokole (*Secure Service Provider*). Negotiate SSP je aplikativni protokol kojim je implementirano pregovaranje u Windows-u. Protokoli koji su podržani u aplikaciji su NTLM i Kerberos:

NTLM (*NT Lan Manager*) je autentifikacioni protokol zasnovan na *challenge/response* šemi, čime je omogućena autentifikacija bez razmena poverljivih podataka (šifre). Jedan od problema ovakvih protogola jeste to što servis mora znati originalnu šifru korisnika kako bi mogao validirati pristigli response. Takođe ovakav protokol ne omogućuje obostranu autentifikaciju, odnosno klijent ne zna da li je servis od poverenja.

Kako bi se prevazišla slabost NTLM protokola, odnosno nemogućnost obezbeđivanja obostrane autentifikacije uvodi se treća strana poverenja. Autentifikacioni protokoli koji se zasnivaju na obostranoj autentifikaciji i koji podrazumevaju razmenu poruka između dva učesnika u autentifikaciji na potpuno isti način su generalno nesigurni protokoli. Iz tog razloga je neophodno uvođenje trećeg entiteta kome će verovati svi ostali učesnici u komunikaciji. Protokol koji ispunjava date uslove naziva se Kerberos i predstavlja dvosmerni autentifikacioni protokol koji se zasniva na postojanju treće strane od poverenja i razmeni propusnica u cilju obezbeđivanja bezbedne obostrane autentifikacije. Kerberos je namenjen za domenska okruženja gde uslugu treće strane, tj. strane poverenja, ima posebno konfigurisani server. Taj server se naziva domen kontroler (*DC*), i on predstavlja autoritet na nivou celog domena kome pripada skup računara i korisničkih naloga.

Certificate autentifikacioni model se zasniva na izdavanju sertifikata od sertifikacionih tela (*CA – certificate authority*). Gde sertifikat predstavlja digitalni identitet korisnika i sa svojim atributima dodaje detaljnost. Podaci koje sadrži sertifikat su: podaci o vlasniku sertifikata (*Subject*), period važenja sertifikata (*Valid from/to*), informacije o izdavaocu sertifikata (*issuer*). U sertifikat se ugrađuje javni ključ korisnika (uz identifikator algoritma primenjenog za generisanje ključa, npr. RSA), dok se tajni ključ ne razmenjuje.

Svaki sertifikat je digitalno potpisan od strane sertifikacionog tela koje ga izdaje čime se potvrđuje da sertifikat zaista pripada podnosiocu zahteva. Na ovaj način je takođe moguće detektovati izmene u okviru samog sertifikata jer digitalni potpis obezbeđuje integritet podataka.

Odgovornosti koja snosi svako sertifikaciono telo su:

- **Verifikacija identiteta** (*Verification of a certificate requestor*) – Pre nego što izda sertifikat, odgovornost ovlašćenog lica, odnosno CA-a, je verifikacija identiteta onoga ko šalje zahtev. Kada korisnik šalje zahtev za sertifikat, on mora da pošalje sve neophodne

informacije koje CA zatim ugrađuje u sertifikat. Tip sertifikata određuje njegov sadržaj, kao i namenu.

- **Izdavanje sertifikata** (*Issuing certificates to requestors*) – Nakon uspješne verifikacije identiteta korisnika, računara, servisa, mrežnog uređaja CA izdaje digitalno potpisani sertifikat. Odnosno, sertifikat sa svim potrebnim informacijama će dodatno biti potpisan javnim ključem sertifikacionog tela kako bi se omogućila detekcija neovlašćenih izmena sadržaja sertifikata, ali i potvrda da je upravo određeni CA izdao sertifikat.
- **Povlačenje sertifikata** (*Certificate revocation*) – U nekim situacijama kao što je kompromitovanje sertifikata, potrebno je povući sertifikat iako je još uvek validan. CA ima listu povučenih sertifikata odnosno njihovih serijskih brojeva, uključujući i razlog zbog čega je svaki sertifikat povučen – CRL.

Autorizacija

Mehanizam autentifikacije treba da obezbedi validaciju identiteta i tako spreči pristup nevalidnim korisnicima. Međutim, različiti korisnici mogu imati različita ovlašćenja u sistemu kome pristupaju. Dakle mehanizmom autentifikacije nije moguće odrediti nivo ovlašćenja korisnika usluga. Proces definisanja ovlašćenja validnim entitetima u sistemu, kao i proces odlučivanja kojim resursima tog sistema entitet može da pristupi i koje operacije nad resursima može da izvrši se naziva autorizacija.

Konkretno, za implementaciju autorizacije korišćen je RBAC autorizacioni model. RBAC model se zasniva na korisničkim ulogama u sistemu i koje privilegiju su u sklopu date uloge. Tako sva tri procesa: identifikacija, autentifikacija i autorizacija čine mehanizam kontrole pristupa resursima.

Auditing

Auditing se odnosi na proces praćenja, snimanja/logovanja, analize i izveštavanja o bezbednosnim događajima u sistemu.

Bezbednosni događaji mogu biti kako uspešno izvršene akcije u sistemu, tako i neuspešni pokušaji pristupa resursima. Audit log predstavlja zapis bezbednosno relevantnih događaja u sistemu. S obzirom da audit log predstavlja vremenski obeležene zapise o aktivnostima u sistemu, učesnici ne mogu naknadno poricati izvršene akcije čime se obezbeđuje neporecivost. Integritet, odnosno tačnost podataka koje sadrže audit logovi se obezbeđuje primenom mehanizama kontrole pristupa logovima, digitalnim potpisima, itd.

Analizom prikupljenih informacija moguće je detektovati kako uspešne tako i neuspešne pokušaje kako redovnih tako i malicioznih aktivnosti u sistemu, odnosno naknadno utvrditi uzroke grešaka ili otkaza u sistemu.

Ovim su potkrepljeni svi bezbednosni mehanizmi korišteni za realizaciju projekta sa teorijskog stanovišta. U daljem tekstu će biti praktična realizacija istih kao i dizajn i arhitektura samog projekta.

3. DIZAJN IMPLEMENTIRANOG SISTEMA

Arhitektura

Arhitektura korišćena u svrhu implementacije jeste klijent-server arhitektura jer problem nalaže postojanje SecureProxy komponente koja želi da uspostavi komunikacio kanal ka SecureHost komponenti i time koristi usluge. Prednosti ovakve arhitekture jeste centralizovani nadzor kao i centralizovana administracije čime je znatno olakšano rešenja problema bezbednosne komunikacije. Bilo da su u pitanju izdati sertifikati ili kolekcije sa autentifikovanim korisnicima, sve se nalazi na jednoj lokaciji.

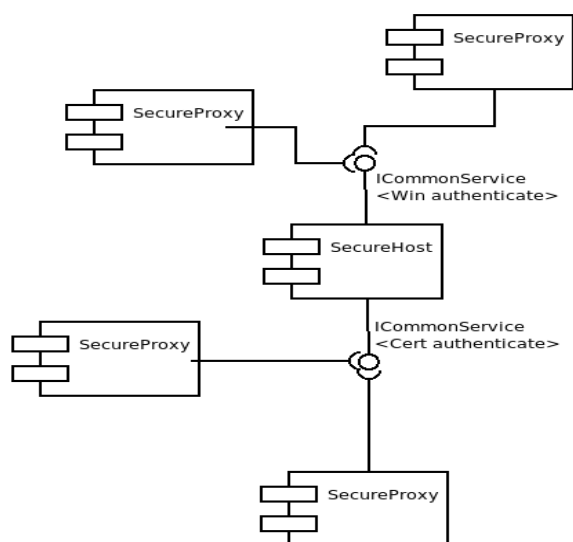
Klijent-server arhitektura opravdava postojanje administrativne komponente zadužene za ažuriranje RBAC konfiguracije autorizacionog modela. Kako komponenta vodi računa samo o korisničkim ulogama i privilegijama na jednoj lokaciji neće doći do kolizija podata o autorizacionim pravima korisnika u *thread-safe* okruženju.

Dizajn

Dizajn aplikacije „Zadatak 4.“ biće izložen kroz dijagrame koji će predstavljati komponente i veze koje postoje među njima. Dijagrami košćeni za predstavljanje i pojašnjenje dizajna aplikacije su: komponentni, use-case, dijagram aktivnosti, klasni.

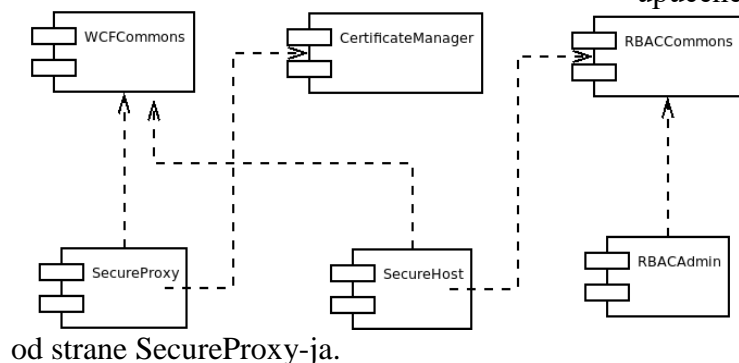
Komponentni dijagram

Dve osnovne komponente sistema su SecureProxy i SecureHost. Dijagram produkcije (Slika 1.) prikazuje višestruke klijente koji uspostavljaju komunikaciju SecureHost-om. Medjusobno mogu biti spregnuti preko ICommonService-a i u slučaju Windows autentifikacije i u slučaju Certificate modela.



Slika 1. Dijagram produkcije

SecureProxy po odabiru načina autentifikacionog modela se „kači“ na tačku pristupa koju SecureHost specifikira za dati tip autentifikacionog modela. Na taj način SecureHost zna tačan način kako da usluži zahteve upućene



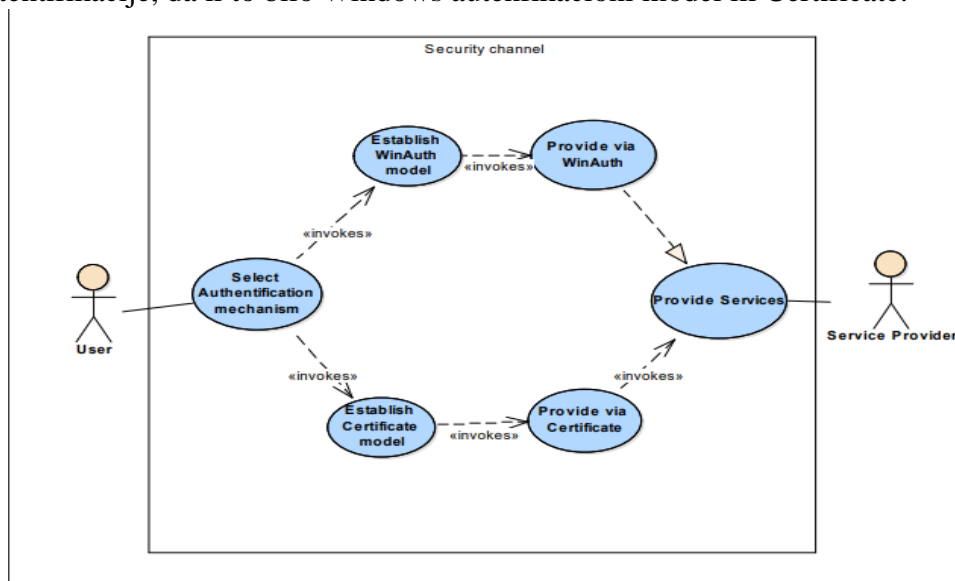
Slika 2. Dijagram projektnih komponenti

Dijagram projektnih komponenti (Slika 2.) predstavlja sve komponente i način na koje su spregnute. RBACAdmin je komponenta koja ažurira uloge i privilegije koje su dodeljene korisnicima. Za realizaciju se koristi RBACCommons komponentom koja predstavlja skup interfejsa zaduženih za administraciju RBAC konfiguracionih fajlova, kao manipulaciju XML dokumentima.

WCFCommons je biblioteka koja sadži potrebne ugovore za uspostavu TCP komunikacije. Njome se koriste SecureProxy i SecureHost.

USE-CASE dijagram

Kao što se vidi sa dijagrama, akteri u slučaju aplikacije su korisnik koji predstavlja SecureProxy i „Service Provider“ koji predstavlja SecureHost. Kada korisnik želi da uspostavi komunikacioni kanal sa provajderom usluga SecureProxy po uspostavljanju TCP konekcije nudi mogućnost odabira načina tipa autentifikacije, da li to bilo Windows autentifikacioni model ili Certificate.



Po odabiru tipa autentifikacije uspostavlja se komunikacioni kanal koji će na svojevrsno implementiran način pristupiti uslugama provajdera i na taj način će klijentu biti omogućeno korišćenje usluga SecureHost-a.

Zaključno, SecureProxy po odabiru modela autentifikacije kreira se objekat konfigurisan na taj način da uspostavi željenu TCP konekciju sa SecureHost-om. Oba tipa klijenta i WinAuth i Cert implementiraju zajednički interfejs IWCFCClient koji u sebi sadrži objekat preko čega će se dalje vršiti razmena zahteva između klijenta i servera.

SecureHost je onaj koji implementira specificirani bezbednosni mehanizam, odnosno ima implementiranu podršku za oba tipa autentifikacije. Takođe ima podršku za RBAC autorizacioni model, kao i praćenje log-a. CommonService implementira ICommonService gde se prvo vrši provera identiteta korisnika kao i njegove privilegije, potom se svaka akcija zapisuje u event log preko SIEM objekta koji komunicira sa Windows event log-erom. Ukoliko bude nepravilnosti u radu dolaziće do bacanja izuzetaka. Kako bi Windows autorizacija imala željenu ulogu, neophodno je implementirati .NET interfejs vezane za autorizacionu politiku kako bi ponašanje aplikacije rešavalo dati problem. Takođe, SecureHost mora imati podršku rada sa sertifikatima, što znači da mora postojati implementiran mehanizam baratanja sertifikatima. Za rad sa identitetima zadužena je klasa WinRBACAuthorizationPolicy koja manipuliše sa autentifikovanim korisnicima, kako Windows autentifikacija tako i one koji su putem sertifikata.

RBACAdmin kao i SecureHost implementira usluge RBACCommons biblioteke za rad sa RBAC konfiguracijom. Korisnik (admin) se može služiti uslugama kao što su dodavanje i uklanjanje grupa i/ili privilegija.

4. TESTIRANJE SISTEMA

RBAC Autorizacija

RBAC testiranje dokazuje da klijent ne može da pristupi funkcijam CommonService-a u koliko nema neophodne dozvoje definisane u RBAC konfiguraciji.

```
3) Read file
4) Delete file
5) Exit
1
Enter file name:
test.txt
Error: Access is denied.

-----
2) Modify file
3) Read file
4) Delete file
5) Exit
1
Enter file name:
test.txt

Enter key to use function:
1) Create file
```

```
1 <Groups>
2 <Group>
3   <Name>wcfAdminGroup</Name>
4   <Permissions>
5     <Perm>Access</Perm>
6     <Perm>Write</Perm>
7     <Perm>Read</Perm>
8   </Permissions>
9 </Group>
10
11 <Group>
```


Nakon dodavanje neophodnih dozvola u RBAC konfiguraciju, klijent dobija mogućnost izvršavanja funkcija.

Windows Autorizacija

Ovaj test pokazuje funkcionisanje windows autorizacije pomoću klijent grupa. U koliko se klijent ne nalazi u grupi koja ima dozvolu potrebnu za izvršavanje funkcije, biće odbijen.

```
Enter key to use function:
1) Create file
2) Modify file
3) Read file
4) Delete file
5) Exit
1
Enter file name:
test.txt
Error: Access is denied.

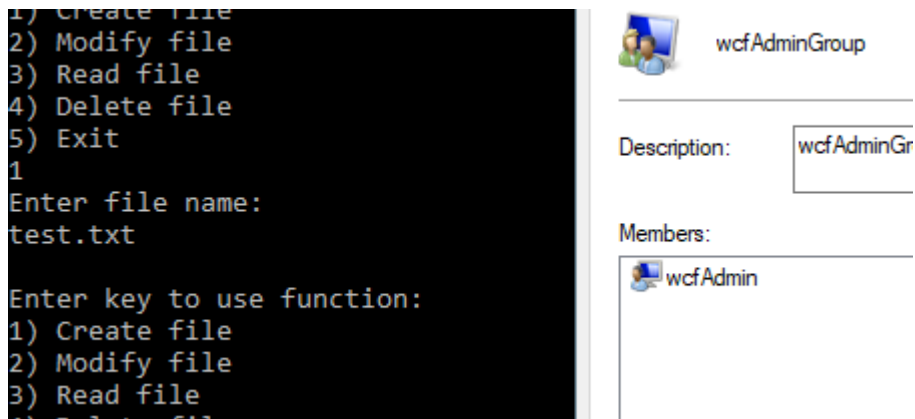
Enter key to use function:
```

**wcfAdminGroup**

Description:

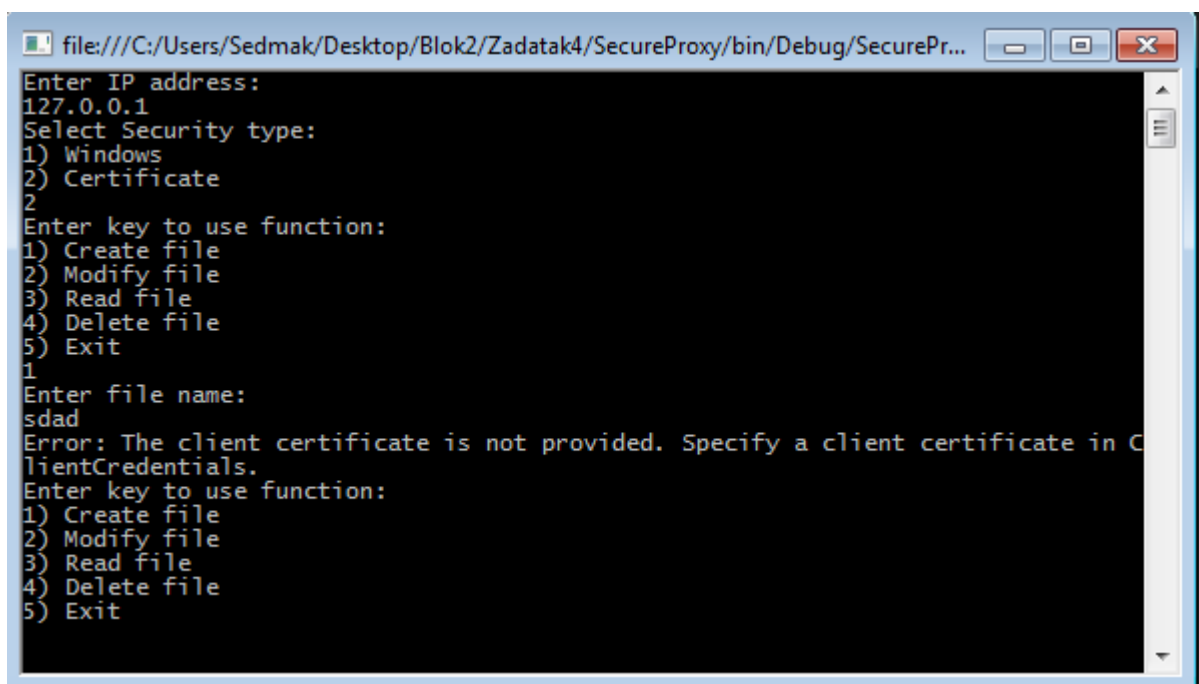
Members:

Dodavanjem klijenta u grupu omogućuje pristup.

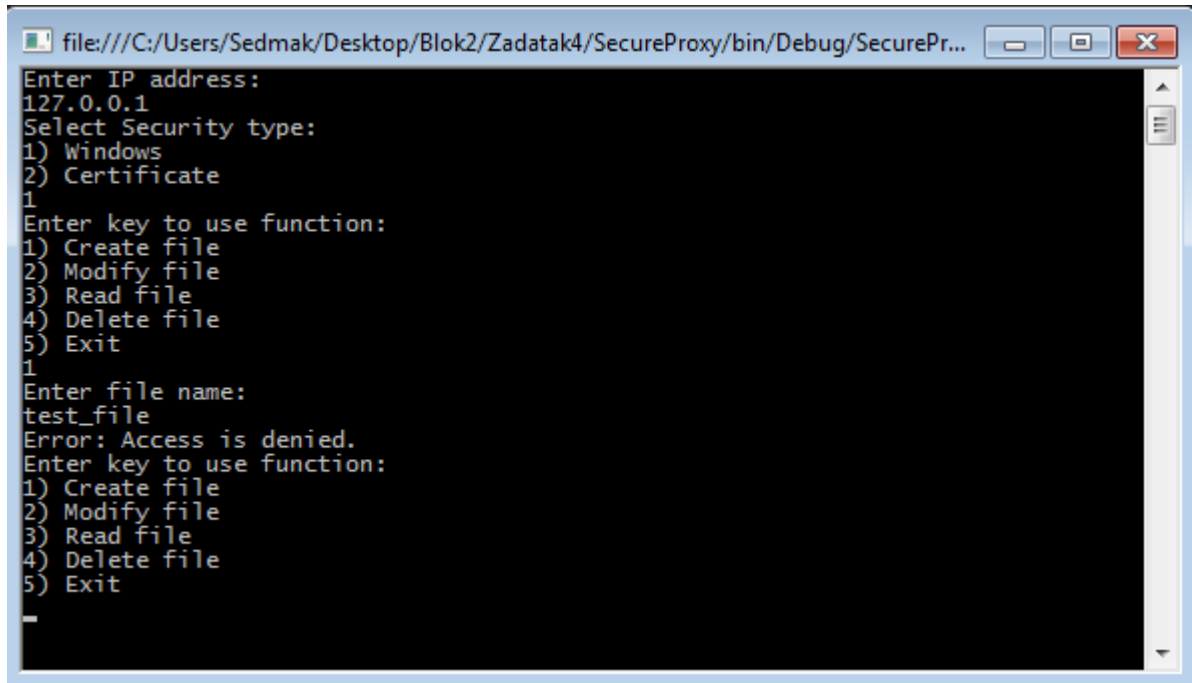


Autorizacija i Autentifikacija Sertifikatima

Ovim testovima se pokazuje mogućnost autorizacije i autentifikacije pomoću sertifikata. U koliko klijent nema potreban sertifikat, pristup servisima mu neće biti omogućen.

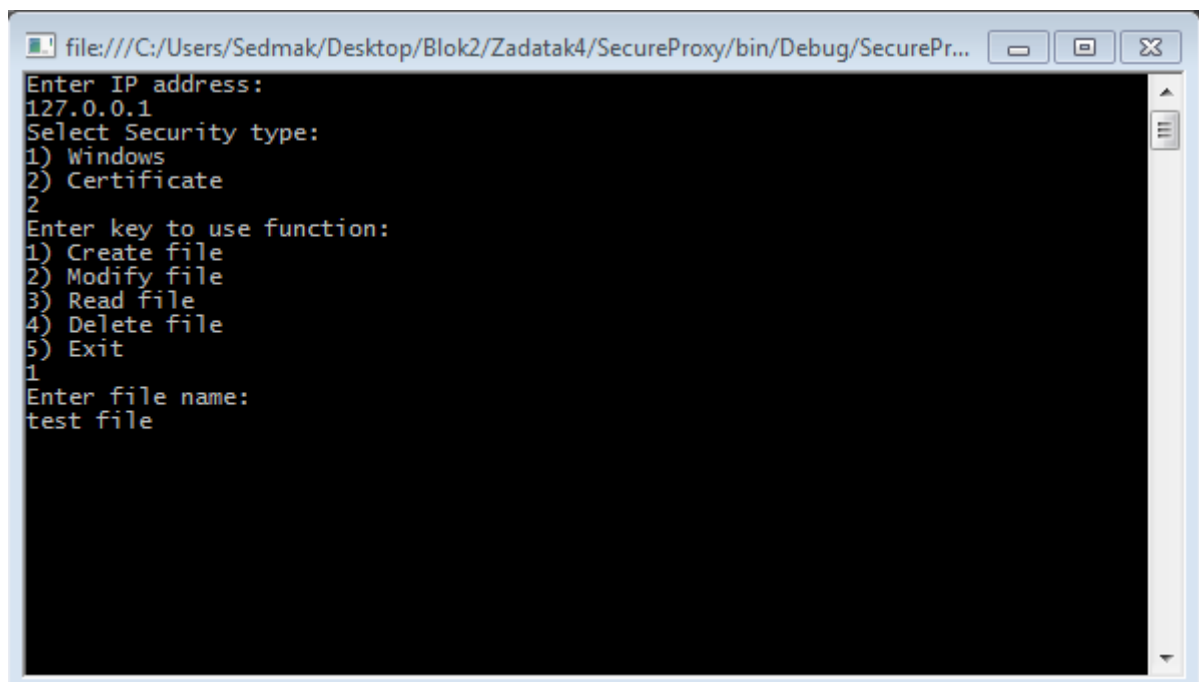


Dodavanjem sertifikata, klijent dobija pristup servisima, ali i dalje nema pristup funkcijama zbog nedostatka permisije u njegovom sertifikatu.



```
file:///C:/Users/Sedmak/Desktop/Blok2/Zadatak4/SecureProxy/bin/Debug/SecurePr...
Enter IP address:
127.0.0.1
Select Security type:
1) Windows
2) Certificate
1
Enter key to use function:
1) Create file
2) Modify file
3) Read file
4) Delete file
5) Exit
1
Enter file name:
test_file
Error: Access is denied.
Enter key to use function:
1) Create file
2) Modify file
3) Read file
4) Delete file
5) Exit
1
```

Po dodavanju prava u sertifikat, klijent dobija pristup funkcijama.



```
file:///C:/Users/Sedmak/Desktop/Blok2/Zadatak4/SecureProxy/bin/Debug/SecurePr...
Enter IP address:
127.0.0.1
Select Security type:
1) Windows
2) Certificate
2
Enter key to use function:
1) Create file
2) Modify file
3) Read file
4) Delete file
5) Exit
1
Enter file name:
test file
```