

Sigurnost Apache Cassandra baze podataka

Studijski program: Računarstvo i informatika

Modul: Softversko inženjerstvo

Predmet: Sistemi za upravljanje bazama podataka

|  |  |
| --- | --- |
| Student: | Profesor: |
| Milica Sokolov,  broj indeksa 1655 | Prof. dr Aleksandar Stanimirović |

Niš, Maj 2024. godina

**Sadržaj:**

[Uvod 2](#_Toc167394299)

[Bezbednost 3](#_Toc167394300)

[Autentifikacija i autorizacija 3](#_Toc167394301)

[system and system\_auth keyspaces 4](#_Toc167394302)

[Keyspace "system" je neizmenjiv 5](#_Toc167394303)

[Pristup system\_auth Keyspace-u sa omogućenom autentifikacijom 6](#_Toc167394304)

[Upravljanje korisničkim dozvolama 8](#_Toc167394305)

[Pristupanje system\_auth koristeći AllowAllAuthorizer 13](#_Toc167394306)

[SSL Enkripcija u Cassandri 16](#_Toc167394307)

[Priprema sertifikata servera za SSL enkripciju 17](#_Toc167394308)

[Enkripcija između čvorova 19](#_Toc167394309)

[Enkripcija između klijenta i čvora 19](#_Toc167394310)

[Povezivanje sa SSL enkripcijom 20](#_Toc167394311)

[Povezivanje putem Cassandra-cli 20](#_Toc167394312)

[Povezivanje putem cqlsh 22](#_Toc167394313)

[Zaključak 24](#_Toc167394314)

[Literatura: 25](#_Toc167394315)

# Uvod

U današnjem digitalnom dobu, bezbednost podataka postaje sve važnija tema. Organizacije, bez obzira na svoju delatnost, suočavaju se sa izazovom zaštite osetljivih podataka i aplikacija baza podataka od neovlašćenog pristupa. Bez obzira na to da li se radi o finansijskim transakcijama ili o ličnim informacijama korisnika, bezbednost podataka je ključna za održavanje poverenja korisnika i poštovanje regulatornih zahteva.

Cassandra je poznata po svojoj skalabilnosti i visokoj dostupnosti, ali kako garantovati sigurnost podataka u ovom okruženju?

Kroz ovaj seminarski rad, biće obrađene metode zaštite podataka putem kontrole pristupa i enkripcije, kao i važnost autorizacije i autentifikacije u Cassandra bazi podataka.

Pored toga, detaljno će biti istraženi ključni koncepti kao što su keyspace-ovi sistema i system\_auth u Cassandri, upravljanje dozvolama korisnika, kao i SSL enkripcija između čvorova i klijentskog servera.

Kroz ove teme, cilj seminara je pružiti uvid u najbolje prakse i tehnike za obezbeđivanje sigurnosti podataka u Cassandra bazi podataka, omogućavajući organizacijama da održe visok nivo zaštite podataka i odgovore na izazove savremenog poslovnog okruženja.

# Bezbednost

Postoje tri glavna komponente bezbednosnih funkcija koje pruža Cassandra:

* TLS/SSL enkripcija za komunikaciju klijenta i međusobnu komunikaciju čvorova
* Autentikacija klijenta
* Autorizacija

Podrazumevano, ove funkcije su isključene kako bi olakšale pronalaženje i povezivanje sa drugim članovima klastera. Međutim, to znači da postoji veći rizik od napada. Čak i kada omogućimo autentikaciju za klijente, to nije dovoljno da zaštitimo klaster. Zlonamerni korisnici koji mogu pristupiti internim komunikacijama i JMX portovima i dalje mogu dodati lažne korisnike u autentikacionu šemu, promeniti šemu ili obrisati podatke, koristiti alate za promenu podataka u sistemskim tabelama ili pristupiti klasteru direktno i presresti podatke koji se šalju.

Da bi se smanjila ova opasnost, potrebno je pravilno konfigurisati sve tri komponente bezbednosti. Razumevanje ovih funkcija je ključno za podešavanje sistema u skladu sa vašim bezbednosnim zahtevima.

# Autentifikacija i autorizacija

Autentifikacija predstavlja proces kontrole korisnika koji pokušavaju pristupiti skladištu podataka. Identitet korisnika mora biti potvrđen prilikom osiguravanja veze sa bazom podataka. Postoje tri osnovna tipa autentifikacije:

1. **Interna autentifikacija** - Obično podrazumeva upravljanje korisničkim prijavljivanjem putem podataka kao što su korisničko ime i lozinka.
2. **Eksterna autentifikacija** - Eksterna autentifikacija, poput Kerberosa, koristi mrežne protokole koji potvrđuju identitet klijenta kroz sistem tiketa.
3. **Enkripcija između klijenta i servera** - Ovaj metod obezbeđuje siguran prenos podataka između klijenta i servera ili između čvorova u mrežnom klasteru. Enkripcija zahteva da javni sertifikati klijenata ili čvorova budu instalirani na odgovarajućim mrežnim tačkama radi uspostavljanja sigurne komunikacije.

Proces autentifikacije u osnovi se svodi na proveru i identifikaciju korisnika. S druge strane, obrada kontrole pristupa korisnika poznata je kao autorizacija. Kada govorimo o autorizaciji u kontekstu baza podataka, to uključuje upravljanje ulogom i privilegijama korisnika na nivou šeme, tabela i kolona.

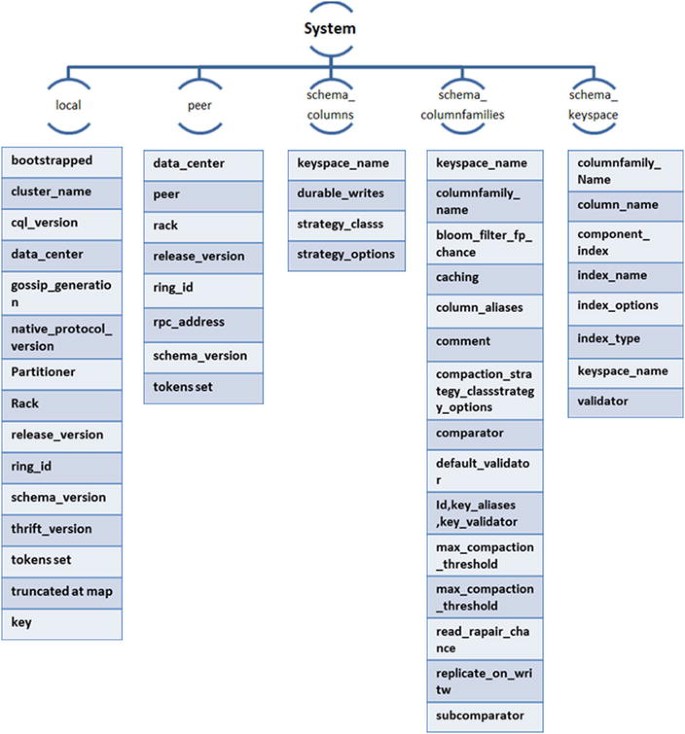
Interna autentifikacija u Cassandri se oslanja na mehanizam SSL enkripcije. Ovaj mehanizam obezbeđuje sigurnu komunikaciju između različitih delova sistema.

## system and system\_auth keyspaces

Keyspace "system" sadrži informacije o dostupnim column family, kolonama i klasterima. U njemu se nalaze osnovni podaci o arhitekturi baze podataka. Keyspace "system\_auth" uglavnom sadrži informacije o autentifikaciji, korisničkim akreditacijama i dozvolama. Tabele/column family u keyspace-u "system" su:

* schema\_keyspace
* schema\_columns
* schema\_columnfamilies
* local
* peers

Na slici 1 prikazana je hijerarhija keyspace "system", gde je korenski čvor keyspace "system", podređeni čvorovi su porodične tabele, i svaka tabela predstavlja kolone definisane sa tim porodičnim tabelama.



*Slika 1. Keyspace "system" i osnovne column family sa odgovarajućim kolonama*

### Keyspace "system" je neizmenjiv

Budući da keyspace "system" internim mehanizmima upravlja arhitektura skladištenja u Cassandri, eksterni korisnici ne mogu da izmene ili ažuriraju ovaj keyspace. U nastavku ovog dela će biti sprovođeni testovi kako bi se dokazalo ovo ograničenje. Iako obično nije neophodno testirati ovakve negativne scenarije prilikom dizajniranja šeme u Cassandri, u kontekstu bezbednosti je važno razmotriti koliko je keyspace "system" zaštićen od neovlašćenih izmena.

Podrazumevani pristup za čitanje keyspace "system" je automatski omogućen. Sada cemo pokušati da pristupimo njemu kroz neke primere koristeći LocalStrategy.

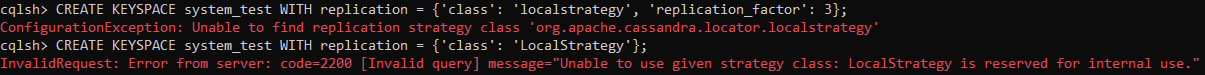
Keyspace "system" je konfigurisan sa LocalStrategy kao opcijom replikacije. "LocalStrategy" je dizajnirana za interni keyspace "system" u Cassandri. Ovaj keyspace sadrži metapodatke i interne informacije potrebne za upravljanje i održavanje baze podataka. Kao takva, "LocalStrategy" je specijalno optimizovana za ove potrebe i nije predviđena za korisničke keyspace-ove. Ova strategija replikacije ima jedinstvenu konfiguraciju koja ne odgovara uobičajenim scenarijima replikacije podataka.

Prvo, pokušajmo kreirati keyspace sa strategijom replikacije:

create keyspace system\_test with replication = { 'class':'localstrategy', 'replication\_factor':3};

create keyspace system\_test with replication = { 'class':'LocalStrategy'};

Slika 2 pokazuje da ne možemo kreirati drugi keyspace sa LocalStrategy.

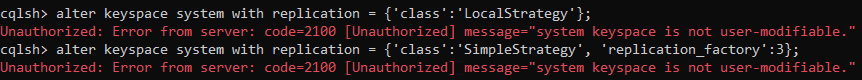


*Slika 2. LocalStrategy nije za javnu upotrebu*

Sada, ćemo da pokušajmo da izmenimo/obrišemo keyspace "system":

alter keyspace system with replication = { 'class' : 'LocalStrategy'};

alter keyspace system with replication = { 'class' : 'SimpleStrategy', 'replication\_factor':3};



*Slika 3. Nije moguće izmeniti keyspace "system" kako bi se promenila strategija.*

Ovo znači da ne možemo menjati keyspace "system" ni na koji način.

Zatim, pokušajmo da obrišemo keyspace "system":

drop keyspace system;

Slika 4 pokazuje da takođe ne možemo obrisati keyspace "system".



*Slika 4. Brisanje keyspace-a "system" nije opcija*

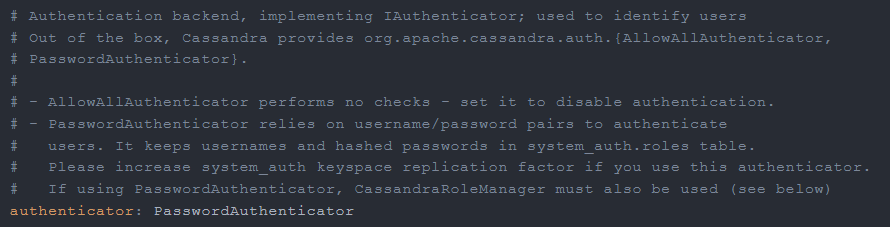
Sa ovim primerima možemo zaključiti da nije dozvoljeno izmenjivanje keyspace-a "system". Međutim, možemo izmeniti podatke prisutne u column family unutar keyspace-a "system".

### Pristup system\_auth Keyspace-u sa omogućenom autentifikacijom

Binarna distribucija Cassandre dolazi sa dva načina autentifikacije: PasswordAuthenticator i AllowAllAuthenticator. Kada se koristi PasswordAuthenticator, Cassandra proverava korisnike na osnovu njihovih korisničkih imena i heširanih lozinki. Ove korisničke informacije se čuvaju u tabeli system\_auth.credentials.

U ovom delu, biće diskotovano o upravljanju korisničkim akreditacijama i pristupanju system\_auth keyspace-u koristeći PasswordAuthenticator. Kada je autentifikacija omogućena, potrebno je uneti korisničko ime i lozinku prilikom povezivanja sa Cassandra serverom. Cassandra čuva korisnička imena i heširane lozinke unutar tabele akreditacija u ovom keyspace-u.

Pre nego što počnemo, moramo omogućiti autentifikaciju u cassandra.yaml na sledeći način:



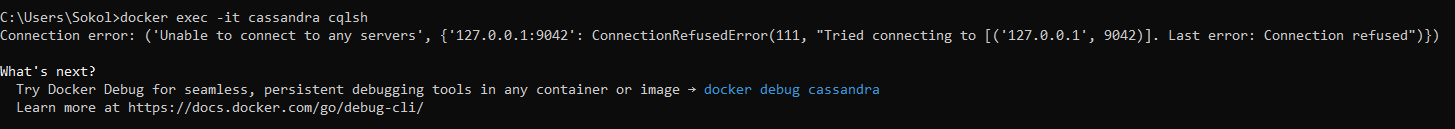
*Slika 5. Omogućena autentifikacija u u cassandra.yaml*

Sada kada je autentifikacija omogućena, mogu se izvršiti sledeći koraci:

1. Prvo, povezati se na server koristeći cqlsh alat putem Docker komande:

docker exec -it cassandra cqlsh

Kada se izvrši ova komanda, može se pojaviti greška slična sledećoj:



*Slika 6. Greška u konekciji zbog nedostatka autentifikacije pre pokušaja kreiranja keyspace-a*

Ova greška se javlja zbog toga što autentifikacija nije uspešno sprovedena pre pokušaja pristupa bazi podataka putem cqlsh alata. To znači da je autentifikacija aktivna, ali nije pruženo korisničko ime i lozinka prilikom pokušaja povezivanja.

1. Da bi se rešio ovaj problem, potrebno je da ponovo pokreneti komandu za povezivanje, ali ovaj put pružiti odgovarajuće kredencijale za autentifikaciju.

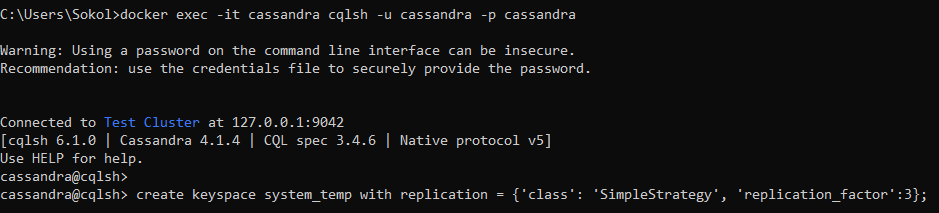
docker exec -it cassandra cqlsh -u cassandra -p cassandra

Nakon što se pruže ispravne kredencijale, trebalo bi da uspešno uspostavite vezu sa Cassandra serverom i pristupite cqlsh interfejsu za rad sa bazom podataka.

Sada se može kreirati keyspace:

create keyspace system\_temp with replication = { 'class' : 'SimpleStrategy', 'replication\_factor' : 3 };

Slika 7 prikazuje uspešnu konekciju.



*Slika 7. Nakon uspešne konekcije koristeći korisničke akreditacije*

1. Zatim, kreirati superkorisnika i običnog korisnika:

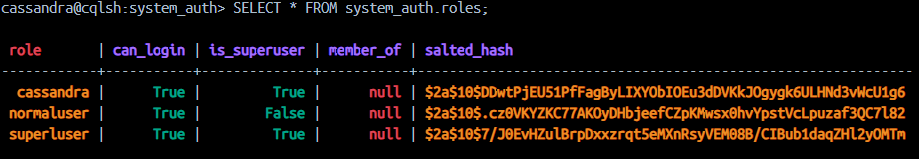
CREATE ROLE normaluser WITH SUPERUSER=false AND LOGIN=true AND

PASSWORD='normal';

CREATE ROLE superuser WITH SUPERUSER=true AND LOGIN=true

AND PASSWORD='super';

Ovo će čuvati informacije o korisnicima u tabeli system\_auth.roles, kao što je prikazano na slici.



*Slika 8. Lista uloga*

## Upravljanje korisničkim dozvolama

Cassandra pruža mehanizam za upravljanje korisničkim dozvolama i pristupnim podacima za autentifikovane korisnike.

Prvo, pokušajmo da se autentifikujemo sa korisnikom "normaluser" i upravljamo njegovim dozvolama pomoću AllowAllAuthorizer-a:

1. Povežimo se sa cqlsh.

cqlsh -u normaluser -p normal

1. Izlistajmo sve dozvole korisnika "normaluser".

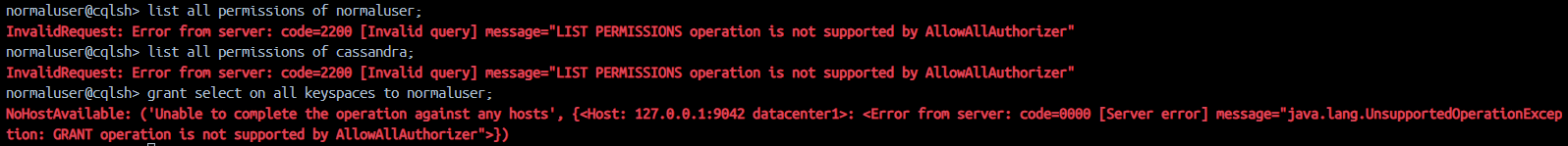
list all permissions of normaluser;

list all permissions of cassandra;

1. Dodelimo dozvole korisniku "normaluser".

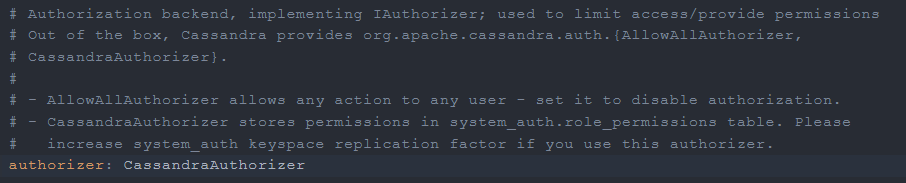
grant select on all keyspaces to normaluser;

Ovo rezultira greškom autorizacije prikazanom na slici 9. Stoga je potrebno omogućiti autorizator u fajlu cassandra.yaml.



*Slika 9. Upravljanje dozvolama s AllowAllAuthorizer-om nije dozvoljeno*

Sada ćemo da omogućimo konfiguraciju autorizatora u datoteci cassandra.yaml i ponovo pokrenemo server. Na slici 10 možete videti kako omogućiti CassandraAuthorizer u cassandra.yaml datoteci.



*Slika 10. Omogućavanje CassandraAuthorizer u cassandra.yaml*

Sada, pokušajmo da kreiramo keyspace sa normaluser-om:

CREATE KEYSPACE testkeyspace WITH replication = { 'class' : 'SimpleStrategy',

'replication\_factor' : 3};

Opet dobijamo grešku (Slika 11) jer nije dodeljena dozvola za kreiranje korisniku normaluser.



*Slika 11. Greška se javlja zato što korisnik normaluser nema dozvolu da kreira keyspace.*

Takođe, normaluser ne može da dodeli dozvole sebi. Samo superkorisnik to može.

Zato, sada ćemo se ulogovati kao superkorisnik:

cqlsh -u superuser -p super

Sada kreirajmo keyspace i dodelimo dozvole korisniku normaluser:

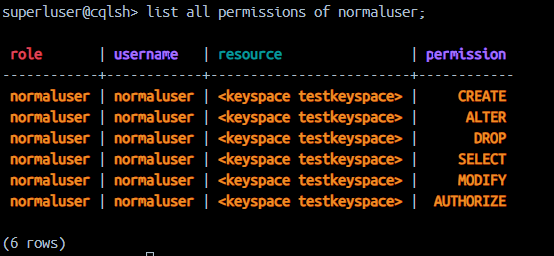
create keyspace testkeyspace with replication = { 'class' : 'SimpleStrategy' , 'replication\_factor' : 3};

grant all on keyspace testkeyspace to normaluser;

Nakon izdavanja dozvola korisniku normaluser, server će sačuvati njegove dozvole u tabeli dozvola pod sistemskim ključem system\_auth.

Takođe, možemo da vidimo sve dodeljene dozvole korisniku normaluser (slika 12).

list all permissions of normaluser;



*Slika 12. Trenutne dozvole normaluser-a*

Sada ćemo da kreiramo još jedan ključni prostor i tabelu privilegija u njemu pod imenom privileges:

create keyspace anotherkeyspace with replication = { 'class' : 'SimpleStrategy'

,'replication\_factor' : 3};

create table anotherkeyspace.privileges (user\_id text PRIMARY KEY,

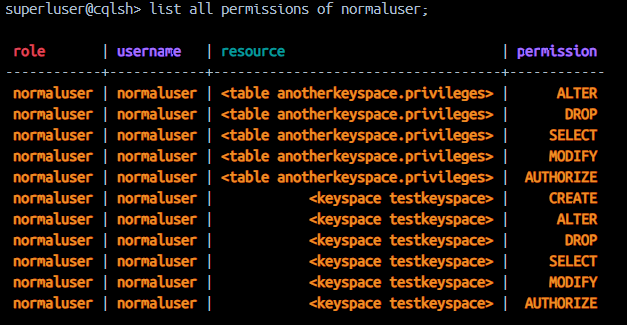
read\_write boolean);

Dodelimo sve dozvole na tabelu privileges korisniku normaluser:

grant all on table anotherkeyspace.privileges to normaluser;

Sada ćemo ponovo da izlistamo sve dozvole korisnika normaluser (slika 13).

list all permissions of normaluser;



*Slika 13. Nova lista dozvola normaluser-a*

Zatim, dajmo sve dozvole na svim ključnim prostorima korisniku superuser:

grant select on all keyspaces to 'superuser';

I sada dajmo dozvolu za select korisniku superuser na tabeli privilegija u ključnom prostoru:

grant select on anotherkeyspace.privileges to 'superuser';

Pogledajmo sve dozvole dodeljene na tabeli privilegija (slika 14).

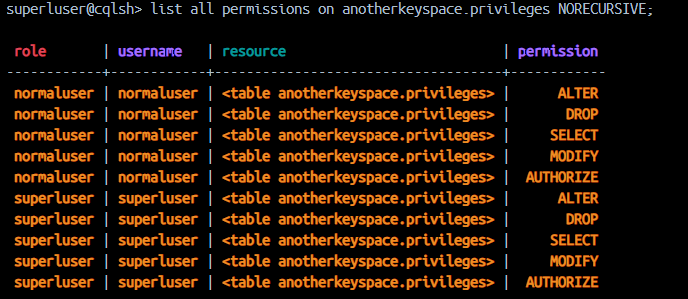
list all permissions on anotherkeyspace.privileges;



*Slika 14. Dozvole na tabeli privileges i njenim nadređenim resursima*

Takođe možemo prikazati dozvole specifične za resurs koristeći NORECURSIVE specifikator (slika 15). On se koristi da prikaže samo direktne dozvole koje su dodeljene resursu bez uključivanja dozvola koje su nasledne kroz hijerarhiju uloga ili drugih resursa.

list all permissions on anotherkeyspace.privileges NORECURSIVE;



*Slika 15. Lista dozvola dodeljenih svim korisnicima na tabeli privileges*

Pokušajmo se povezati koristeći normalnog korisnika i pokušajmo videti sve dozvole dodeljene tabeli privileges.

list all permissions on anotherkeyspace.privileges;

Opet dobijamo grešku "nije autorizovan" (Slika 16). Normalnom korisniku su dodeljene sve dozvole na tabeli privileges, ali ne može videti dozvole drugih korisnika. Međutim, normalni korisnik može izvršiti druge dozvoljene operacije koje su prethodno pomenute.



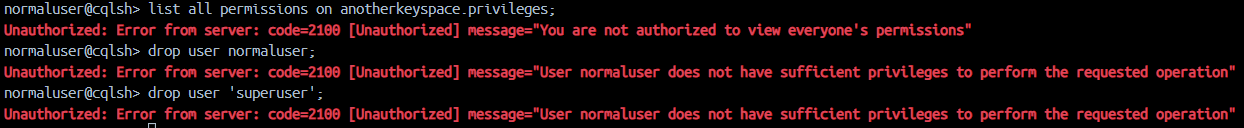
*Slika 16. Nema dozvola dato normalnom korisniku na anotherkeyspace*

Pokušajmo takođe da normalni korisnik obriše sam sebe i superkorisnika (slika 17).

drop user normaluser;

drop user 'superuser';

Rezultat, prikazan na slici 17, pokazuje da normalni korisnik ne može obrisati, opozvati ili pregledati dozvole drugih korisnika. Ali superkorisnik može.

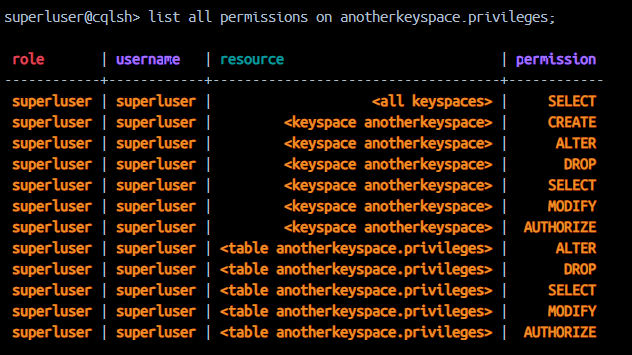


*Slika 17. Brisanje korisnika nije dozvoljeno za korisnike koji nisu superkorisnici*

Dakle, povežimo se sa superkorisnikom i pokušajmo opozvati dozvole koje su bile date određenom korisniku (normaluser, u ovom slučaju):

revoke all permissions on anotherkeyspace.privileges from normaluser;

list all permissions on anotherkeyspace.privileges;



*Slika 18. Dpzvp;e superusera u tabeli privilegija*

Iz ove serije koraka zaključujemo:

* Korisnik koji nije superuser ne može upravljati, pristupati niti pregledati dozvole drugih korisnika.
* Dozvole se mogu dati na ključni prostor i određenu tabelu/kolonu.
* Superkorisnik može upravljati ili pristupiti dozvolama drugih korisnika.

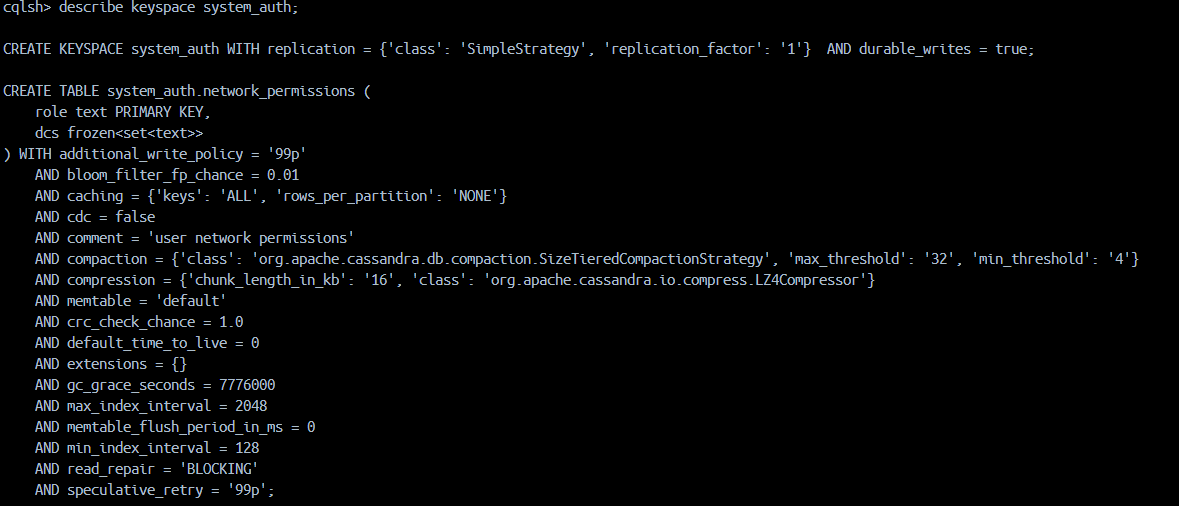
## Pristupanje system\_auth koristeći AllowAllAuthorizer

U Cassandri, podrazumevano je da je autentikacija isključena, a koristi se AllowAllAuthenticator kao autentikator. U prethodnom delu smo istraživali različite tehnike autorizacije koristeći PasswordAuthenticator. U ovom delu ćemo istražiti da li je moguće upravljati dozvolama korisnika kada je AllowAllAuthorizer konfigurisan. Ovo će nam pomoći da shvatimo šta neće raditi sa AllowAllAuthorizer-om, i to samo u eksperimentalne svrhe.

Ključni prostor system\_auth sadrži korisničke akreditive i detalje dozvola. U ovoj proceduri, pokušaćemo da pristupimo system\_auth ključnom prostoru i upravljamo korisničkim akreditivima bez autentikacije.

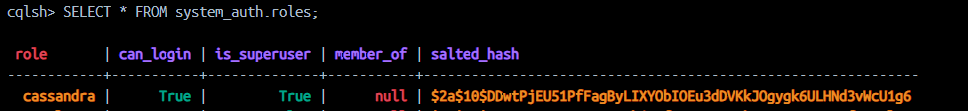
Prvo ćemo opisati keyspace system\_auth (Slika 19).

describe keyspace system\_auth;



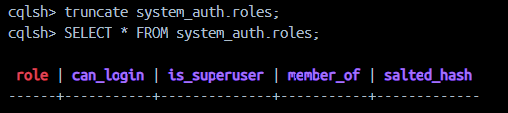
*Slika 19. Opisan keyspace system\_auth*

Sada ćemo da istražimo tabelu korisnika kao što je prikazano na slici 19.



*Slika 20. Podrazumevani korisnik*

Za početak, hajde da pokušamo da obrišemo sve podatke iz tabele korisnika (Slika 21):



*Slika 21. Izvrsenje naredbe truncate*

Uspešno smo izbrisali sadržaj tabele korisnika, ali sada da vidimo da li možemo da je obrišemo.

drop table system\_auth.roles;

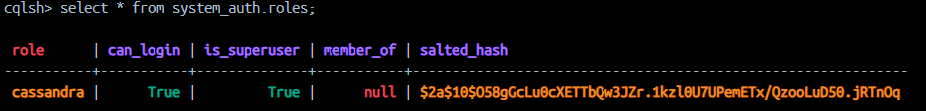
Međutim, slika 22 pokazuje da to nije moguće jer je ova tabela sistematska i zaštićena od strane Cassandrea za spoljne korisnike.



*Slika 22. Brisanje tabele system\_auth.roles nije dozvoljeno*

Zatim, restartujte Cassandra server da biste proverili da li je tabela korisnika prazna (Slika 23):

select \* from system\_auth.roles;



*Slika 23. Podrazumevani korisnik cassandra je unapred popunjen*

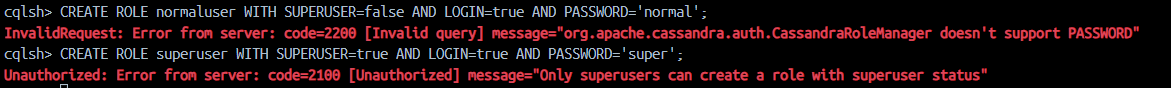
Izmene koje se vrše u ključnom prostoru system\_auth kada je omogućen AllowAllAuthorizer neće biti prihvaćene od strane Cassandrea. Zbog toga promene koje su napravljene u koraku 3 neće imati efekta, i Cassandra će zadržati podrazumevane postavke kao što je prikazano u koraku 2.

Sada ćemo pokušati da kreiramo korisnika:

create role normaluser with password 'normal';

create role normaluser;

Međutim, kreiranje korisnika sa AllowAllAuthenticatorom nije dozvoljeno u Cassandri, kako se može videti na slici 24. Ipak, kao što smo ranije videli, ovo je moguće koristeći PasswordAuthenticator.



*Slika 24. Kreiranje korisnika sa AllowAllAuthenticatorom*

Možemo promeniti keyspace system\_auth za replikacionu strategiju:

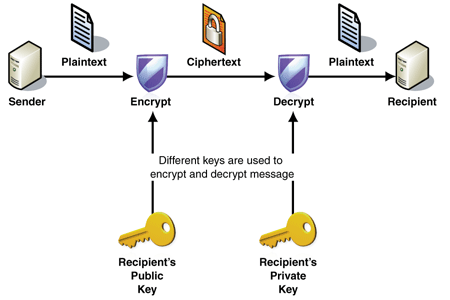
alter keyspace system\_auth with replication = {'class':'SimpleStrategy', 'replication\_factor':3};

Zaključujemo da možemo promeniti keyspace system\_auth, ali ne možemo kreirati korisnika sa AllowAllAuthenticator-om (kada je autentifikacija onemogućena).

# SSL Enkripcija u Cassandri

Sigurnosni sloj (SSL) je kriptografski protokol koji se koristi za obezbeđivanje komunikacije između računara. SSL šifruje podatke tokom prenosa kako bi se sprečili slučajni ili namerni pokušaji čitanja podataka.

Ukratko, SSL funkcioniše na sledeći način: Dva entiteta, bilo softverska ili hardverska, komuniciraju međusobno. Ovi entiteti mogu biti klijent i čvor ili parovi čvorova u klasteru. Da bi uspostavili poverenje, moraju razmeniti informacije. Svaki entitet mora imati generisani ključ koji se sastoji od privatnog ključa koji samo taj entitet čuva i javnog ključa koji se razmenjuje sa drugim entitetima. Kada klijent želi da se poveže sa serverom, on zahteva sigurnu vezu i server šalje sertifikat koji sadrži njegov javni ključ. Klijent proverava validnost sertifikata razmenom informacija sa serverom, koje server potvrđuje svojim privatnim ključem. Ako se zahteva dvostruka validacija, ovaj proces se sprovodi u oba smera. Privatni ključevi i sertifikati čuvaju se u keystore-u, a javni ključevi se čuvaju u truststore-u. Za sisteme koji koriste Certificate Authority (CA), truststore može čuvati sertifikate potpisane od strane CA za verifikaciju. Obe datoteke, keystore i truststore, imaju dodeljene lozinke nazvane keypass i storepass.



*Slika 25. Primer SSL enkripcije*

Apache Cassandra pruža sledeće SSL funkcije šifrovanja:

* Šifrovana komunikacija između čvorova - Internodna enkripcija se koristi za obezbeđivanje podataka koji se prenose između čvorova u klasteru.
* Šifrovana komunikacija između klijenta i čvora - Enkripcija između klijenta i čvora se koristi za obezbeđivanje podataka koji se prenose između klijentskih programa, kao što su cqlsh, DevCenter ili nodetool, i čvorova u klasteru.

## Priprema sertifikata servera za SSL enkripciju

U poslednjih nekoliko godina, društvene mreže i internet aplikacije omogućile su dostupnost i deljenje podataka širom sveta. SSL protokoli se koriste za slanje enkriptovanih podataka preko interneta sa sigurnom komunikacijom.

Za komunikaciju između klijenata i servera ili između čvorova preko SSL-a, potrebno je pripremiti sertifikate servera. Keystore datoteka sadrži ključeve servera, dok Truststore sadrži poverene SSL sertifikate za sve klijente ili čvorove.

U nastavku su detaljno opisani koraci za konfiguraciju sigurnosnih sertifikata:

1. Inicijalni korak u postavljanju sigurnosne infrastrukture je preuzimanje odgovarajućih JCE paketa za podršku Java verzije koja se koristi. Ovi paketi omogućavaju napredne sigurnosne funkcionalnosti unutar Java aplikacija.

<http://www.oracle.com/technetwork/java/javase/downloads/jce-6-download-429243.html> (Java 6)

<http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html> (Java 7)

1. Nakon preuzimanja, kopirajte datoteke local\_policy.jar i US\_export\_policy.jar u direktorijum $JAVA\_HOME/jre/lib/security, gde je $JAVA\_HOME direktorijum u kojem je instaliran JDK.
2. Ključni korak je generisanje keystore-a koji sadrži privatne ključeve potrebne za identifikaciju servera. Korišćenjem alata poput keytool, generišu se keystore datoteke sa odgovarajućim sigurnosnim parametrima.

keytool -genkey -alias servernode -keystore /home/impadmin/keys/server/server.jks

-storepass server -keypass server

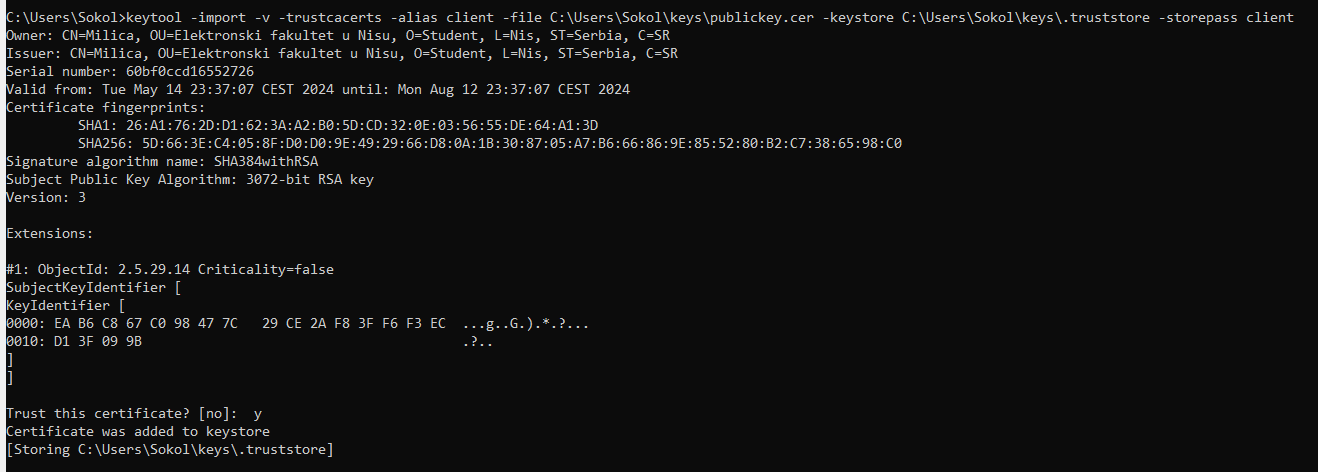
1. Nakon generisanja keystore-a, javni deo ključa, neophodan za verifikaciju servera od strane klijenata, izvozi se u posebnu certifikatnu datoteku.

keytool -export -alias servernode -file /home/impadmin/keys/publickey.cer -keystore /home/impadmin/keys/server.jks -storepass server

1. Konačni korak je uvoz izvezenog javnog ključa u truststore servera, omogućavajući serveru da proveri autentičnost klijenata tokom sigurnosne komunikacije.

keytool -import -v -trustcacerts -alias client -file /home/impadmin/keys/publickey.cer -keystore /home/impadmin/keys/.truststore -storepass client

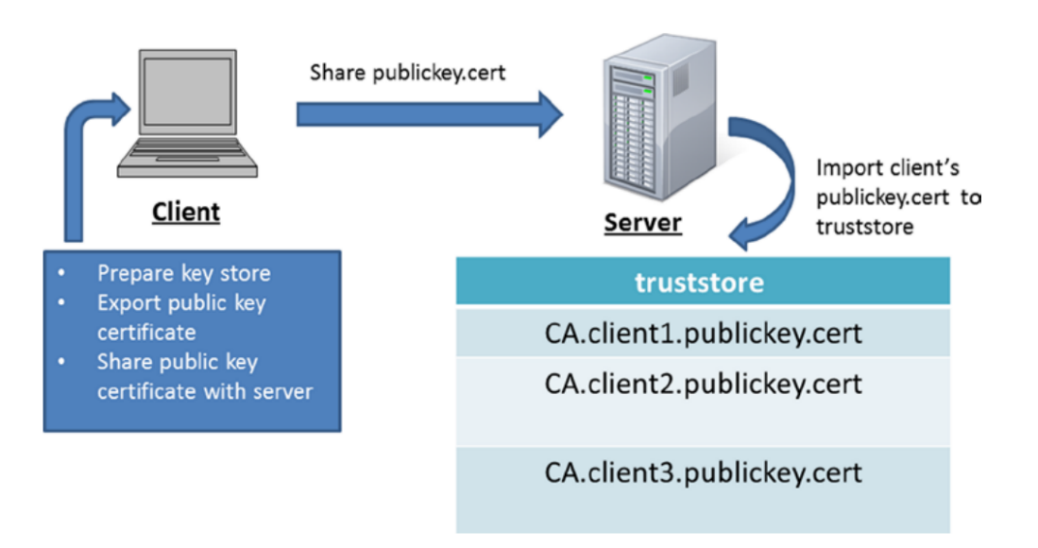
Slika 26. prikazuje da je sertifikat uspešno dodat.

**

*Slika 26. Sertifikat klijenta uspešno dodat u povereno skladište servera*

Ovim smo pripremili sertifikate servera. U nastavku koristićemo truststore da pristupimo serveru sa pouzdanim sertifikatima. Za komunikaciju među čvorovima, moramo ponoviti gorenavedeni postupak na svakom čvoru i kopirati javni sertifikat svakog čvora u truststore svakog čvora.

Slika 27 prikazuje grafički prikaz pripreme i uvoza sertifikata klijenta sa truststore-om.



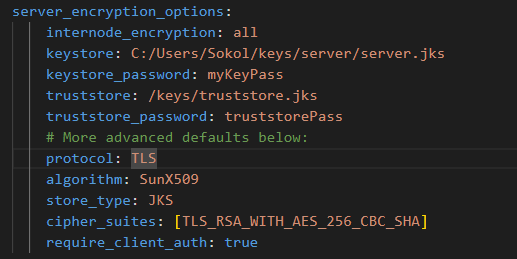
*Slika 27. Proces pripreme i deljenja javnog ključa klijenta sa serverom*

Postupak pripreme i deljenja sertifikata među čvorovima Cassandre je isti, i svaki čvor mora izvesti i podeliti javne ključeve sa svim ostalim čvorovima u klasteru Cassandre. Kada je proces pripreme sertifikata završen, trebate omogućiti i povezati komandne linije klijenata sa SSL enkripcijom.

## Enkripcija između čvorova

Enkripcija između čvorova štiti podatke koji se prenose između čvorova u klasteru, uključujući gossip komunikacije, koristeći SSL (Secure Sockets Layer).

Najpre je potrebno omogućiti server\_encryption\_options na svakom čvoru. Izmeniti cassandra.yaml fajl sa sledećim podešavanjima:



*Slika 27. Izmene u cassandra.yaml fajlu*

Ovaj fajl koristi sertifikate pripremljene za razvoj.

Enkripcija između čvorova može biti postavljena na četiri različita načina:

* all - Sav saobraćaj je enkriptovan.
* none - Nijedan saobraćaj nije enkriptovan.
* dc - Saobraćaj između datacentara je enkriptovan.
* rack - Saobraćaj između rackova je enkriptovan.

Potrebno je ostaviti odgovarajuće putanje do keystore i truststore fajlova. Takođe, postaviti lozinke na lozinke koje su postavljene tokom generisanja keystore i truststore. Ako je željena dvosmerna autentifikacija sertifikata, postaviti require\_client\_auth na true.

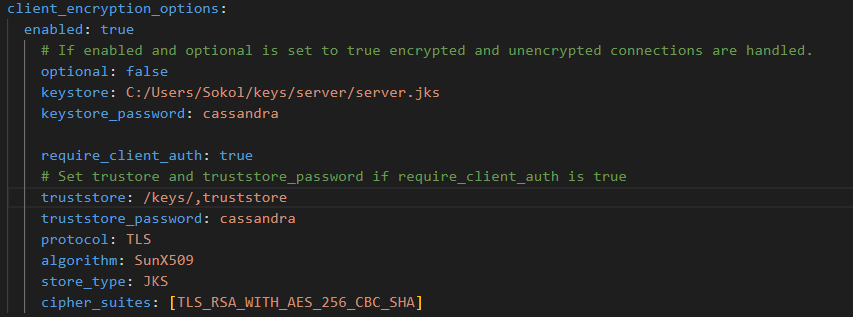
Restartovati cassandru da bi promene postale efektivne. Proveriti logove da otkrijete da li je SSL enkripcija pokrenuta.

## Enkripcija između klijenta i čvora

Enkripcija između klijenta i čvora (client-to-node encryption) štiti podatke koji se prenose sa klijentskih mašina do baze podataka u klasteru koristeći SSL (Secure Sockets Layer). Ova enkripcija uspostavlja siguran kanal između klijenta i koordinacionog čvora u klasteru.

Da bi se omogućila enkripcija između klijenta i čvora, potrebno je postaviti opcije client\_encryption\_options u cassandra.yaml fajlu na svakom čvoru.

U cassandra.yaml fajlu dodati sledeće:



*Slika 28, Izmene u cassandra.yaml fajlu*

Postaviti odgovarajuće putanje do keystore i truststore fajlova i lozinke koje su korišćene pri njihovom generisanju. Ako želite dvosmernu autentifikaciju sertifikata, postaviti require\_client\_auth na true. Ovo omogućava alatima da se povežu na udaljeni čvor. Za lokalni pristup pomoću cqlsh na lokalnom čvoru sa SSL enkripcijom, require\_client\_auth može biti postavljeno na false.

Ako su potrebni i enkriptovani i neenkriptovani saobraćaj, omogućiti dodatno podešavanje u cassandra.yaml fajlu:

native\_transport\_port\_ssl: 9142

Da bi promene u cassandra.yaml fajlu stupile na snagu, potrebno je restartovati Cassandra čvor. Nakon restarta, proveriti logove da biste se uverili da je SSL enkripcija uspešno pokrenuta.

## Povezivanje sa SSL enkripcijom

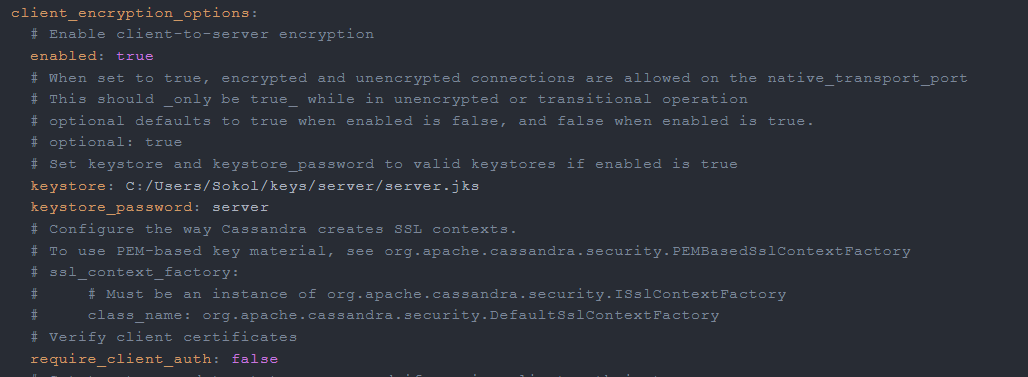
Sada ćemo istražiti povezivanje sa Cassandra-cli komandnim klijentom sa uključenom SSL enkripcijom.

### Povezivanje putem Cassandra-cli

Sada ćemo omogućiti enkripciju između klijenata i čvorova i povezati se sa serverom koristeći Cassandra-cli.

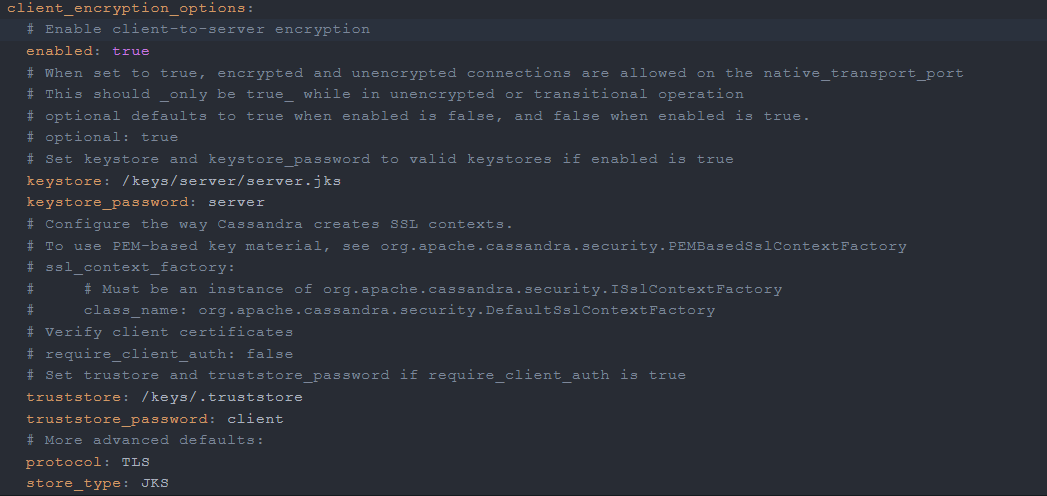
Korak 1: Konfigurisanje servera za enkripciju između klijenata i čvorova

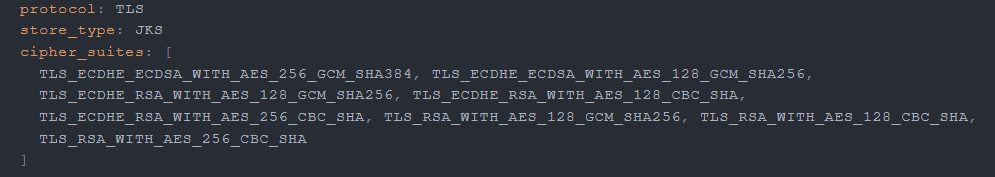
Na slici 29 je prikazan primer konfigurisanja cassandra.yaml datoteke.



*Slika 29. Konfiguraciju SSL enkripcije za klijentsko-server povezivanje*

Sledeća konfiguracija prikazuje potrebne izmene u fajlu cassandra.yaml za postavljanje putanja do keystore i truststore datoteka:





*Slika 30. Izmene u fajlu cassandra.yaml za postavljanje putanja do keystore i truststore datoteke*

Konfiguracija cipher\_suites sadrži sve važeće protokole i preporučuje se da je ne menjate prilikom konfiguracije aplikacije. Ako želite da omogućite autentifikaciju klijentskih sertifikata, postavite require\_client\_auth na true.

Pokrenite Cassandra server i povežite se pomoću cassandra-cli:

$CASSANDRA\_HOME/bin/cassandra-cli -h 127.0.0.1 -p 9160 -ts

/home/impadmin/keys/.truststore

-tf org.apache.cassandra.cli.transport.

SSLTransportFactory -tspw client

U ovom primeru ts označava putanju do truststore fajla, tf označava transport factory, tspw označava lozinku za truststore.

Sa omogućenim SSL-om, možemo se povezati sa Thrift klijentom koristeći konfiguraciju transport factory i truststore.

### Povezivanje putem cqlsh

Za povezivanje cqlsh-a sa omogućenim šifrovanjem, potrebno je kreirati .cqlshrc datoteku u korisničkom direktorijumu. Takođe, možemo se povezati sa cqlsh-om sa require\_client\_auth=false.

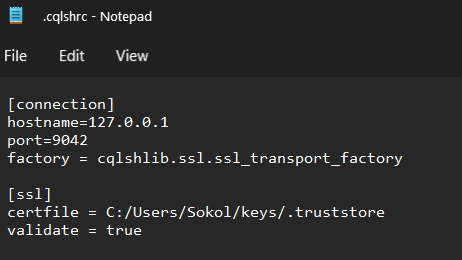
Kreirajmo .cqlshrc datoteku u korisničkom direktorijumu za SSL-specifične konfiguracije.

factory = cqlshlib.ssl.ssl\_transport\_factory

// putanja do fajla poverenja (truststore)

certfile = /home/impadmin/source/keys/.truststore

Slika 31 prikazuje promene konfiguracija u .cqlshrc datoteci.

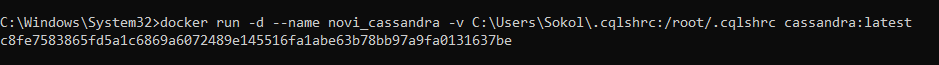


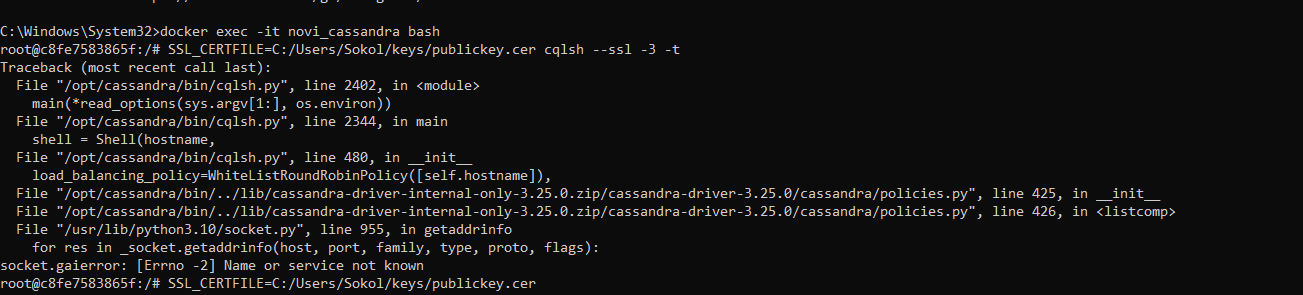
*Slika 31. Promene konfiguracija u .cqlshrc datoteci*

Pokrenimo cqlsh da bismo se povezali sa Cassandra serverom:

SSL\_CERTFILE=/home/impadmin/source/keys/publickey.cer

SSL\_VALIDATE=$CASSANDRA\_HOME/bin/cqlsh -3 -t cqlshlib.ssl.ssl\_transport\_factory





*Slika 32. Pokretanje cqlsh*

# Zaključak

U ovom seminarskom radu detaljno su proučeni mehanizmi za zaštitu u Apache Cassandra bazi podataka.

Apache Cassandra nudi metode autentifikacije i autorizacije kako bi se osigurala kontrola pristupa. Korišćenjem PasswordAuthenticator-a, Cassandra proverava korisnike na osnovu njihovih imena i heširanih lozinki, koje su sačuvane u keyspace-u system\_auth. Autorizacija omogućava upravljanje korisničkim ulogama i privilegijama na nivou šeme, tabela i kolona, što obezbeđuje da samo ovlašćeni korisnici imaju pristup određenim podacima.

Keyspace-ovi system i system\_auth igraju ključnu ulogu u bezbednosnoj arhitekturi Cassandre. system\_auth keyspace čuva kritične informacije o korisničkim akreditacijama i dozvolama, dok system keyspace održava važne podatke o strukturi baze podataka. Iako je system keyspace neizmenjiv za spoljne korisnike, pruža neophodne informacije za internu upravu i bezbednost baze.

Cassandra omogućava kontrolu pristupa i enkripciju podataka, što dodatno pojačava sigurnost baze podataka. Implementacija enkripcije između klijenta i servera, kao i između čvorova, osigurava da se podaci prenose na siguran način, čuvajući ih od neovlašćenog pristupa tokom transfera.

Kombinacija ovih bezbednosnih mehanizama čini Apache Cassandru pogodnom za upotrebu u okruženjima koja zahtevaju visoku sigurnost podataka, poput finansijskih institucija i državnih organizacija. Implementacijom pravilnih strategija autentifikacije, autorizacije i enkripcije, organizacije mogu efikasno zaštititi svoje podatke i osigurati integritet i poverljivost svojih informacija.

# Literatura:

[1] [Security | Apache Cassandra Documentation](https://cassandra.apache.org/doc/stable/cassandra/operating/security.html#operation-roles)

[2] [Security | Apache Cassandra 3.0 (datastax.com)](https://docs.datastax.com/en/cassandra-oss/3.0/cassandra/configuration/secureTOC.html)

[3] [How To Implement Security In Cassandra (marketsplash.com)](https://marketsplash.com/how-to-implement-security-in-cassandra/)

[4] Beginning Apache Cassandra Development, Vivek Mishra

[5] Learning Apache Cassandra, Mat Brown

[6] [Securing Cassandra - GeeksforGeeks](https://www.geeksforgeeks.org/securing-cassandra/)