**Sigurnost MongoDB baze podataka**

Kristina Antić, 934

Sadržaj

[1. uvod 3](#_Toc40435005)

[1.1. Kontrola pristupa 3](#_Toc40435006)

[1.2. Bezbednost za internet aplikacije 4](#_Toc40435007)

[1.3. Administrator baze podataka 4](#_Toc40435008)

[2. sigurnost mongodb baze podataka 5](#_Toc40435009)

[2.1. Kontrola pristupa bazirana na ulogama 5](#_Toc40435010)

[2.1.1. Upravljanje korisnicima i ulogama 6](#_Toc40435011)

[2.2. Autentifikacija 11](#_Toc40435012)

[2.3. TLS/SSL i enkripcija 13](#_Toc40435013)

[2.3.1. TLS/SSL 13](#_Toc40435014)

[2.3.2. Enkripcija polja na klijentskoj strani 14](#_Toc40435015)

[2.4. Restrikcija na nivou polja 17](#_Toc40435016)

[3. zaključak 20](#_Toc40435017)

[literatura 20](#_Toc40435018)

# uvod

Baza podataka i podaci koji se čuvaju u njoj moraju biti zaštićeni od neovlašćenog i zlonamernog pristupa. Mora se osigurati privatnost i kontrola pristupa podacima, koji ne smeju biti dostupni bilo kojim korisnicima. DBMS ima ugrađene mehanizme zaštite koji se odnose na kontrolu prava pristupa.

Kada se projektuje bezbedna baza podataka postoje tri glavna cilja:

1. ***Tajnost*** – korisnik ne može da pristupa onome što nije namenjeno njemu.
2. ***Integritet*** – korisnici ne mogu da modifikuju ono što im nije dozvoljeno.
3. ***Dostupnost*** – korisnici mogu da vide i modifikuju samo ono što im je dozvoljeno.

Kako bi se ispunili ovi ciljevi potrebno je da postoji ***politika sigurnosti*** koja specificira ko je autorizovan da vrši određene akcije. Takođe, mora da postoji i određeni ***mehanizam sigurnosti*** koji dozvoljava sprovođenje izabrane politike sigurnosti.

## Kontrola pristupa

Baza podataka sadrži veliku količinu informacija i uglavnom nekoliko grupa korisnika. Većem broju korisnika je potreban samo određeni deo podataka iz baze kako bi izvršavali svoje zadatke. Zbog toga DBMS treba da obezbedi mehanizam za kontrolu pristupa podacima. DBMS nudi dva glavna mehanizma za kontrolu pristupa:

* **Discretionary access control** – bazira se na konceptu prava pristupa ili ***privilegija*** za objekte (tabele, pogledi) i mehanizmima dodele i oduzimanja ovih prava korisnicima baze podataka. DBMS čuva informacije o tome ko dobija i gubi privilegije i obezbeđuje da se prihvataju zahtevi samo od korisnika koji imaju odgovarajuće privilegije za posmatrani objekat. Kreator tabele ili nekog drugog objekta automatski dobija sve privilegije nad tim objektom.

Privilegije se često povezuju sa ***ulogama*** korisnika. Mnogi DBMS-ovi podržavaju koncept uloga i privilegija koje im se dodeljuju. Korisnicima se preko uloga dodeljuju odgovarajuće privilegije.

Takođe, kroz ***poglede*** se korisnicima može dati pravo pristupa samo jednom delu podataka. Pogledi se koriste da prikažu potrebne informacije, pri čemu sakrivaju detalje osnovnih relacija. Za kreiranje pogleda potrebne su privilegije za pristup tabelama koje učestvuju u pogledu.

Ovaj mehanizam, iako je generalno dobar, ima neke nedostatke. Nedostaci se odnose na to da neautorizovani korisnici mogu da prevare autorizovane korisnike i na taj način dobiju pristup poverljivim podacima.

* **Mandatory access control** – zasniva se na politikama sigurnosti na nivou celog sistema koje ne može da promeni neki korisnik. Popularni model za mandatory access control je ***Bell-LaPadula model***, koji je opisan preko ***objekata*** (npr. tabele, pogledi, vrste, kolone), ***subjekata*** (npr. korisnici, programi), ***klasa sigurnosti*** i ***dozvola***. Svaki DB ***objekat*** ima dodeljenu ***klasu sigurnosti*** i svaki ***subjekat*** ima dodeljenu ***dozvolu*** za klasu sigurnosti. Pravila koja se zasnivaju na klasama sigurnosti i dozvolama upravljaju ko može da izvrši operacije čitanja/upisa za objekte.

Klase sigurnosti su: ***top secret*** (TS), ***secret*** (S), ***confidential*** (C) i ***unclassified*** (U), pri čemu važi TS > S> C > U . Bell-LaPadula model postavlja dva pravila koja se odnose na čitanje i upis objekata:

1. Subjekat S može da čita objekat O samo ako je class(S) >= class(O) ***(Simple Security Property)***
2. Subjekat S može da upisuje objekat O samo ako je class(S) <= class(O) ***(\*-Property)***

Ova pravila se primenjuju kao dodatak uz sve Discretionary access control mehanizme koji su na snazi.

## Bezbednost za internet aplikacije

Kada se DBMS-u pristupa preko internet aplikacija, potrebni su sofisticiraniji pristupi za autentifikaciju. Potrebno je da obe strane (i korisnici i aplikacije) budu sigurne u to sa kim komuniciraju. Tehnike enkripcije su osnova za modernu autentifikaciju.

Osnovna ideja koja stoji iza enkripcije je primena nekog algoritma za enkripciju nad podacima, korišćenjem posebnog ključa za enkripciju. Izlaz iz algoritma su ekriptovani (šifrovani) podaci. Postoje i algoritmi za dekripciju (dešifrovanje) koji uzimaju šifrovane podatke i korišćenjem ključa za dešifrovanje vraćaju originalne podatke. Algoritmi za enkripciju i dekripciju su javno dostupni, dok su ključevi (jedan ili oba) tajni.

DBMS može da koristi enkripciju za čuvanje i prenos podataka kako bi osigurao da, čak i ako dođe do neautorizovanog pristupa podacima, oni nisu od koristi.

## Administrator baze podataka

Administrator baze podataka je odgovoran za kompletnu sigurnost. Administrator ima poseban nalog, koji se zove ***sistemski nalog***. Obaveze administratora su sledeće:

1. Održavanje istorije pristupa korisnika bazi podataka (***audit trail***).
2. ***Definisanje šema*** – administrator definiše šemu koja sadrži strukturu podataka u aplikaciji. Administrator određuje i koji će se podaci čuvati u sistemu i kako su organizovani.
3. ***Komunikacija sa korisnicima*** – administrator treba da bude u stalnoj interakciji sa korisnicima da bi razumeo podatke koji su u sistemu i njihovo korišćenje.
4. ***Definisanje provera sigurnosti i integriteta*** – administrator vodi računa o restrikcijama pristupa i definiše sigurnosne provere. Provera integriteta podataka je takođe zaduženje administratora.
5. ***Definisanje backup/recovery procedura*** – administrator specificira za koje podatke treba uraditi backup, periodično izvršava backup, definiše periode i medijume za čuvanje podataka.
6. ***Monitoring performansi*** – administrator konstantno nadgleda performanse upita i preduzima mere za njihovu optimizaciju, kako u bazi podataka, tako i u aplikacijama.

# sigurnost mongodb baze podataka

MongoDB obezbeđuje različite mogućnosti za kontrolu sigurnosti baze podataka, kao što su autentifikacija, kontrola pristupa, enkripcija, itd.

Omogućavanjem kontrole pristupa na MongoDB instanci se primenjuje autentifikacija, odnosno zahteva se identifikacija korisnika. Kada se pristupa MongoDB instanci koja ima omogućenu kontrolu pristupa, korisnik može da izvršava samo one akcije koje su omogućene njegovom ulogom. U nastavku je dat detaljan opis metoda za kontrolu pristupa, autentifikaciju i enkripciju.

## Kontrola pristupa bazirana na ulogama

MongoDB koristi kontrolu pristupa baziranu na ulogama kako bi upravljao pristupom MongoDB sistemu. MongoDB po default-u ne omogućava kontrolu pristupa. Autentifikacija mora eksplicitno da se omogući. Detalji o omogućavanju kontrole pristupa i autentifikacije dati su u delu **2.2 Autentifikacija**.

Kreiraju se ***korisnici*** i njima se dodeljuje jedna ili više ***uloga*** kako bi se odredilo kojim resursima mogu da pristupaju i koje operacije mogu da izvršavaju. Postoje ugrađene i korisnički definisane uloge. Informacije o korisnicima i ulogama se čuvaju u posebnoj bazi podataka, ***admin***, odnosno u kolekcijama ***system.users*** i ***system.roles***. Ovim kolekcijama se ne može direktno pristupati, već se za to koriste komande za upravljanje korisnicima i ulogama.

Ulogama se dodeljuju ***privilegije*** za izvršavanje određenih operacija nad resursima. Resurs može da bude baza podataka, kolekcija, skup kolekcija ili klaster. Svaka privilegija je ili eksplicitno specificirana u okviru uloge ili nasleđene od neke druge uloge ili oba. Uloga može da sadrži jednu ili više postojećih uloga u svojoj definiciji i u tom slučaju uloga nasleđuje sve privilegije tih uloga.

Uloge se dodeljuju prilikom kreiranja korisnika. Takođe je moguće ažuriranje postojećih korisnika kako bi im se dodale ili oduzele privilegije. Korisniku se može dodeliti više uloga. Prvi korisnik koji se kreira u bazi podataka treba da bude administrator koji ima privilegije da upravlja ostalim korisnicima. U nastavku će biti opisane postojeće uloge u MongoDB bazi, kao i načini upravljanja korisnicima i ulogama.

MongoDB ima skup ugrađenih uloga, ali je moguće kreirati i korisnički definisane uloge. Svaka baza podataka sadrži sledeće ***klijentske*** uloge:

* ***read*** – obezbeđuje mogućnost čitanja podataka iz svih nesistemskih kolekcija i system.js kolekcije. Ova uloga omogućava pristup akcijama pretrage podataka, pregleda statistike kolekcije i baze, pregleda indeksa i kolekcija, heširanja kolekcije.
* ***readWrite*** – omogućava sve privilegije koje ima i read uloga, uz mogućnost modifikacije podataka iz nesistemskih i system.js kolekcija. Dodatne akcije obuhvataju kreiranje i brisanje kolekcija i indeksa, dodavanje, brisanje i modifikaciju dokumenata, kao i promenu imena kolekciji.

Od ***administratorskih*** uloga sadrži sledeće uloge:

* ***dbAdmin*** – omogućava izvršenje administrativnih zadataka kao što su zadaci koji se odnose na šemu baze podataka, indeksiranje i prikupljanje statističkih podataka. Ova uloga ne dodeljuje privilegije za upravljanje korisnicima i ulogama. Od dodatnih akcija u odnosu na readWrite klijentsku ulogu obuhvata menjanje opcija kolekcije (dodavanje opcija, modifikacija definicija pogleda), validaciju podataka i indeksa, pregled informacija o skladištu, pregled i brisanje keširanih planova izvršenja upita, kompakcija podataka i indeksa (oslobađanje nepotrebnog prostora na disku).
* ***dbOwner*** – vlasnik baze podataka može da izvršava bilo koju administrativnu akciju nad bazom podataka. Ova uloga kombinuje privilegije dodeljene readWrite, dbAdmin i userAdmin ulogama.
* ***userAdmin*** – omogućava kreiranje i modifikaciju uloga i korisnika za određenu bazu podataka. Aktivnosti koje se mogu izvršavati sa ovom ulogom su kreiranje i brisanje korisnika i uloga, dodeljivanje i ukidanje uloga, promena šifre bilo kom korisniku, pregled uloga i korisnika.

Prethodno navedene uloge mogu da se koriste za bilo koju bazu podataka. Za admin bazu podataka postoje još neke dodatne uloge, koje su opisane u nastavku:

* ***backup*** – obezbeđuje minimalne privilegije za backup podataka.
* ***restore*** – obezbeđuje privilegije za vraćanje podataka iz backup-a.
* ***readAnyDatabase*** – omogućava iste privilegije kao read uloga nad svim bazama podataka osim local i config.
* ***readWriteAnyDatabase*** – omogućava iste privilegije kao readWrite uloga nad svim bazama podataka osim local i config.
* ***userAdminAnyDatabase*** – omogućava iste privilegije kao userAdmin uloga nad svim bazama podataka osim local i config. Ova uloga ima i neke dodatne privilegije nad system.users i system.roles kolekcijama, koje se odnose na pretragu podataka, pregled statistike kolekcije, heširanje, pregled keširanih planova za izvršenje upita, kao i kreiranje i brisanje indeksa.
* ***dbAdminAnyDatabase*** – omogućava iste privilegije kao dbAdmin nad svim bazama podataka osim local i config.

Neke uloge obezbeđuju mogućnost da se bilo kom korisniku dodeli bilo koja uloga za bilo koju bazu podataka. To su *dbOwner* i *userAdmin* uloge koje se odnose na admin bazu podataka i *userAdminAnyDatabase* uloga. Postoji i ***root*** uloga koja omogućava pristup svim operacijama i resursima uloga readWriteAnyDatabase, dbAdminAnyDatabase, userAdminAnyDatabase, restore, backup i clusterAdmin.

Postoje i uloge koje su specifične za MongoDB klaster i odnose se na upravljanje klasterom.

Osim ovih uloga, moguće je definisati i nove uloge. U nastavku će biti detaljnije objašnjeno kreiranje novih uloga.

## Upravljanje korisnicima i ulogama

Da bi se upravljalo korisnicima i ulogama, neophodno je prvo omogućiti kontrolu pristupa, što je detaljnije opisano u delu **2.2 Autentifikacija**. Kreiranje korisnika se vrši korišćenjem ***db.createUser()*** metode. U funkciji je potrebno specificirati naziv korisnika (***user***), lozinku (***pwd***), kao i uloge koje su mu dodeljene i na koje baze podataka se one odnose (***roles***). Na slici 2.1.1.1 dat je primer kreiranja korisnika u *test* bazi podataka. Baza podataka gde se kreira korisnik se zove ***baza podataka za autentifikaciju***. Autentifikacija se vrši za tu bazu podataka, ali korisnik može da ima uloge koje se odnose na druge baze podataka.

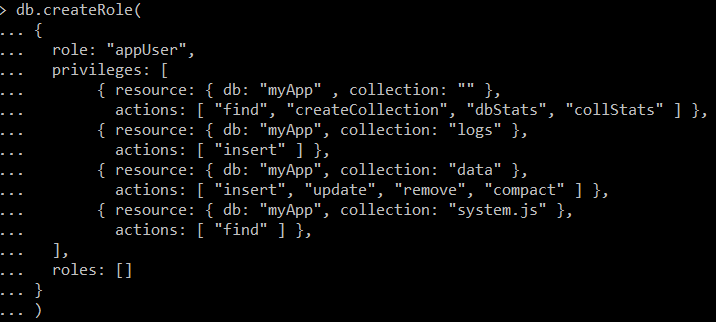


Slika 2.1.1.1 Kreiranje korisnika

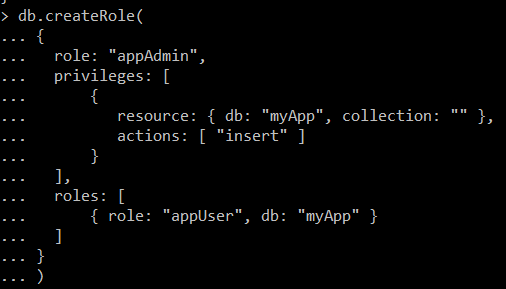
Ukoliko se uloge specificiraju za bazu podataka gde je kreiran korisnik, dovoljno je samo nabrojati imena uloga, bez eksplicitnog navođenja baze podataka na koju se odnose. ***PasswordPrompt()*** se koristi da bi se izbegla specifikacija lozinke, već se prilikom autentifikacije dobija lozinka.

Za kreiranje uloga koristi se ***db.createRole()*** metoda, ge se specificira naziv uloge (***role***), privilegije (***privileges***), kao i uloge koje se nasleđuju (***roles***). Privilegije se sastoje od specifikacije resursa i dozvoljenih ***akcija*** nad njima. Resurs predstavlja bazu podataka i kolekciju na koju se privilegije odnose. Prilikom kreiranja uloga čiji je scope (baza podataka gde je kreirana uloga) baza koja nije admin, prilikom specifikacije resursa, oni moraju da se nalaze u istoj bazi kao uloga. Uloge kreirane u okviru admin baze mogu da specificiraju resurse koji se odnose na druge baze podataka. Lista svih akcija se može naći na <https://docs.mongodb.com/manual/reference/privilege-actions/>. Takođe, prilikom nasleđivanja uloga, ukoliko scope kreirane uloge nije admin baza, ona može da nasleđuje samo druge uloge nad tom bazom podataka. Uloga kreirana nad admin bazom može da nasleđuje uloge kreirane nad drugim bazama, kao i nad admin bazom podataka.

MongoDB koristi kombinaciju imena baze u kojoj se kreira uloga i imena uloge kako bi jedinstveno definisao svaku ulogu. Ulogu može kreirati i dodeliti samo korisnik koji ima privilegije za to. Ugrađene uloge *userAdmin*, *dbOwner* i *userAdminAnyDatabase* imaju ove privilegije. Na slici 2.1.1.2 je primer kreiranja uloge koja ne nasleđuje druge uloge i ima privilegije nad nekim kolekcijama *myApp* baze podataka, a na slici 2.1.1.3 je primer kreiranja uloge koja nasleđuje prethodno kreiranu ulogu i ima još neke dodatne privilegije.



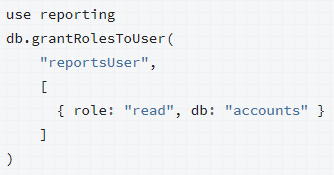
Slika 2.1.1.2 Kreiranje uloge koja ne nasleđuje druge uloge



Slika 2.1.1.3 Kreiranje uloge koja nasleđuje drugu ulogu

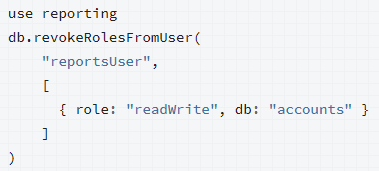
Ukoliko je prilikom specifikacije resursa polje *collection* prazan string, navedene akcije se odnose na sve kolekcije te baze, ne uključujući sistemske kolekcije. Sistemske kolekcije moraju eksplicitno da se navedu. Ukoliko je polje *db* prazan string , resursi su sve kolekcije sa specificiranim nazivom u svim bazama podataka. Ovo je moguće samo za uloge čiji je scope admin baza. Ukoliko su i *collections* i *db* polje prazni stringovi, resursi su sve kolekcije, ne uključujući sistemske kolekcije, svih baza podataka. I ovo je moguće samo za uloge čiji je scope admin baza podataka.

Metoda ***db.grantRolesToUser()*** dodeljuje korisniku ulogu. Kao parametri se navode naziv korisnika i lista uloga koje mu se dodeljuju. Za ulogu koja se odnosi na bazu gde se izvršava metoda, dovoljno je samo navesti ime uloge. Za ulogu koja se odnosi na neku drugu bazu, mora se specificirati par {*role*:naziv uloge, *db*:baza na koju se odnosi uloga}. Na slici 2.1.1.4 dat je primer dodeljivanja *read* uloge nad *accounts* bazom podataka korisniku *reportsUser*.



Slika 2.1.1.4 Dodeljivanje uloge korisniku

Metoda ***db.revokeRolesFromUser()*** ukida korisniku ulogu. Parametri su isti kao i kod *grantRolesToUser()* metode. Na slici 2.1.1.5 je dat primer ukidanja *readWrite* uloge nad *accounts* bazom podataka korisniku *reportsUser*.



Slika 2.1.1.5 Ukidanje uloge korisniku

Da bi mogla da se promeni lozinka nekom korisniku, potrebno je da onaj ko menja ima mogućnost izvršavanja ***changeAnyPassword*** akcije. Lozinka se menja korišćenjem ***db.changeAnyPassword()*** metode. Parametri su naziv korisnika i nova lozinka. Primer promene lozinke *reporting* korisniku je dat na slici 2.1.1.6.



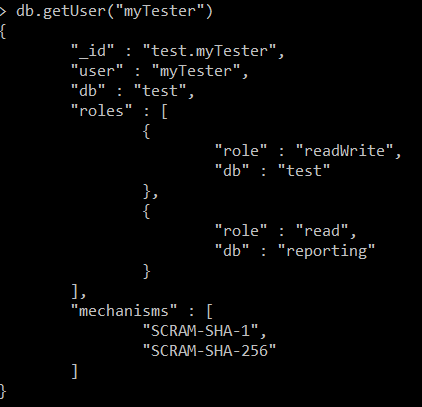
Slika 2.1.1.6 Promena lozinke korisniku

Korisnik može da promeni i svoju lozinku i custom podatke. Custom podaci čuvaju opcione informacije o korisnicima. Da bi mogao da menja svoju lozinku i custom podatke, korisnik mora da ima privilegije koje omogućavaju izvršenje akcija ***changeOwnPassword*** i ***changeOwnCustomData***. Za promenu lozinke i custom podataka koristi se ***db.updateUser()*** metoda koja izvršava prethodno navedene akcije. Parametri ove funkcije su naziv korisnika i podaci koji se menjaju. Upotreba ove funkcije je ilustrovana na slici 2.1.1.7.



Slika 2.1.1.7 Promena svoje lozinke i custom podataka

Moguće je videti sve informacije o nekom korisniku. Svaki korisnik može da vidi svoje infromacije. Da bi mogle da se vide infromacije drugih korisnika, potrebno je imati privilegiju koja uključuje mogućnost izvršenja ***viewUser*** akcije nad bazom podataka gde je kreiran korisnik. Pregled informacija o korisniku se dobija izvršavanjem ***db.getUser()*** metode, kojoj se kao parametar prosleđuje naziv korisnika. Informacije koje se dobijaju izvršavanjem ove metode obuhvataju id i naziv korisnika, bazu podataka u kojoj je kreiran korisnik, uloge dodeljene tom korisniku i podržane mehanizme za autentifikaciju. Na slici 2.1.1.8 prikazane su informacije koje se dobijaju izvršavanjem ove metode. Za pregled informacija o svim korisnicima kreiranim u određenoj bazi podataka koristi se ***db.getUsers()*** metoda.



Slika 2.1.1.8 Pregled informacija o korisniku

Za pregled informacija o ulogama, potrebno je ili da bude omogućeno izvršenje ***viewRole*** akcije nad bazom gde je kreirana uloga ili da je uloga eksplicitno dodeljena korisniku. Informacije o ulozi se mogu dobiti pozivom ***db.getRole()*** metode, kojoj se kao parametri prosleđuju naziv uloge i dodatne opcije (npr. showPrivileges za pregled svih privilegija koje uloga ima). Na slici 2.1.1.9 prikazani su rezultati izvršenja ove naredbe. Mogu se videti osnovne informacije o ulozi, kao što je naziv, baza u kojoj je kreirana uloga, da li je uloga ugrađena ili korisnički definisana, da li nasleđuje neku ulogu, kao i niz privilegija (***privileges***) i niz nasleđenih privilegija (***inheritedPrivileges***). Pošto *appUser* uloga ne nasleđuje ni jednu ulogu, ova dva niza privilegija su ista. Za pregled informacija o svim ulogama kreiranim u određenoj bazi podataka koristi se funkcija ***db.getRoles()***.



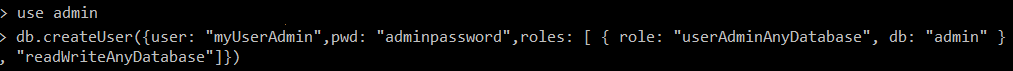
Slika 2.1.1.9 Pregled informacija o ulozi

## Autentifikacija

Autentifikacija je proces verifikacije identiteta klijenta. Autentifikacija je neophodna da bi se utvrdilo kojim resursima klijenti smeju da pristupaju i koje operacije smeju da izvršavaju.

Da bi se izvršila autentifikacija korisnika, potrebno je obezebditi username, password i bazu podataka za autentifikaciju. Pre toga je potrebno kreirati korisnike i dodeliti im odgovarajuće uloge. Više o načinu kreiranja korisnika i uloga se može pročitati u prethodnom delu **2.1 Kontrola pristupa bazirana na ulogama**. U nastavku će biti detaljnije opisan način omogućavanja kontrole pristupa.

Prvi korak je kreiranje korisnika sa ulogom administratora, kako bi moglo da se upravlja korisnicima i ulogama. Na slici 2.2.1 dat je primer kreiranja korisnika sa ulogom administratora u *admin* bazi podataka.



Slika 2.2.1 Kreiranje administratora

Zatim je potrebno restartovati MongoDB instancu sa kontrolom pristupa. Prilikom startovanja instance dodaje se ***--auth*** opcija. Na slici 2.2.2 je prikazan način pokretanja MongoDB instance sa kontrolom pristupa.



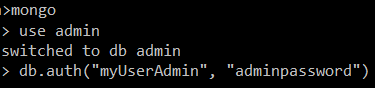
Slika 2.2.2 Pokretanje instance sa kontrolom pristupa

Poslednji korak je autentifikacija kao određeni korisnik (u ovom slučaju administrator). Autentifikacija korišćenjem mongo shell-a se može izvršiti na dva načina:

1. Korišćenjem mongo command-line opcija za autentifikaciju (-u (username), -p (password) i --authenticationDatabase) prilikom konektovanja na mongo instancu.



1. Konektovanjem prvo na mongo instancu, a zatim izvršavanjem ***db.auth()*** metoda u bazi podataka za autentifikaciju. Metodi se kao parametri prosleđuju naziv korisnika i lozinka.



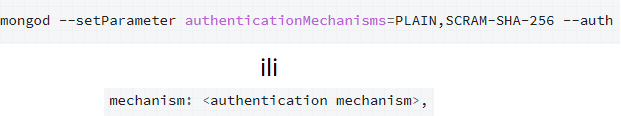
Nakon toga je moguće izvršavati akcije omogućene dodeljenim ulogama.

MongoDB podržava više mehanizama za autentifikaciju koje klijenti mogu da koriste kako bi potvrdili svoj identitet. MongoDB podržava sledeće mehanizme:

* ***SCRAM*** (default) – mehanizam za autentifikaciju korisnika korišćenjem lozinke. MongoDB podržava SCRAM-SHA-1 i SCRAM-SHA-256 mehanizme. Ovi mehanizmi koriste različite heš funkcije.
* ***x.509 Certificate Authentication*** – zahteva bezbednu TLS/SSL konekciju. Detaljnije o TLS/SSL može se naći u narednom delu **2.3.1 TLS/SSL.** Da bi se izvršila autentifaikacija, klijenti koriste x.509 sertifikate umesto korisničkih imena i lozinki. Svaki korisnik mora da ima jedinstveni sertifikat.

Uz prethodno navedene mehanizme, MongoDB Enterprise podržava još neke dodatne mehanizme – LDAP proxy authentication i Kerberos authentication.

Autentifikacioni mehanizam se može specificirati korišćenjem ***authenticationMehanisms*** parametra prilikom pokretanja mongo instance ili preko ***mechanism*** parametra u okviru db.auth() metode. Načini specificiranja mehanizma za autentifikaciju su ilustrovani na slici 2.2.3.



Slika 2.2.3 Specificiranje mehanizma za autentifikaciju

Pored verifikacije identiteta klijenta, MongoDB može da zahteva od članova replica set-ova i klastera da potvrde svoje članstvo odgovarajućem replica set-u ili klasteru (internal authentication).

Kada su u pitanju klasteri, obično se vrši autentifikacija klijenata na nivou klastera. Međutim, ponekad je potrebno izvršiti autentifikaciju u okviru određenog računara u klasteru.

## TLS/SSL i enkripcija

## TLS/SSL

MongoDB podržava TLS/SSL (Transport Layer Security/Secure Sockets Layer) kako bi izvršio enkripciju svog mrežnog sobraćaja koji generiše MnogoDB. TLS/SSL osigurava da je saobraćaj čitljiv samo klijentu kome je namenjen.

MongoDB podržava Forward Secrecy, koji kreira kratkotrajni ključ sesije koji je zaštićen privatnim ključem servera. Korišćenje kratkotrajnog ključa osigurava da, čak iako se privatni ključ servera otkrije, pomoću njega ne može da se dešifruju prethodne sesije. Minimalna dužina ključa mora biti 128 bitova.

Da bi TLS/SSL mogao da se koristi, potreban je TLS/SSL sertifikat. Sertifikat može da se dobije od nekog ko je nadležan za to, ali se može koristiti i self-signed sertifikat. Ukoliko se koristi self-signed sertifikat, komunikacioni kanal će biti enkrtiptovan, ali neće biti validacije identiteta serverske strane. Ukoliko je sertifikat potpisan od strane nekog nadležnog organa, vršiće se potvrda identiteta servera.

Da bi se MongoDB instanca konfigurisala da podržava TLS/SSL, potrebno je u konfiguracionom fajlu navesti putanju do fajla sertifikata (***certificateKeyFile*** polje za TLS ili ***PEMKeyFile*** za SSL), a zatim, prilikom startovanja instance, navesti putanju do konfiguracionog fajla, kao što je prikazano na sledećoj slici.



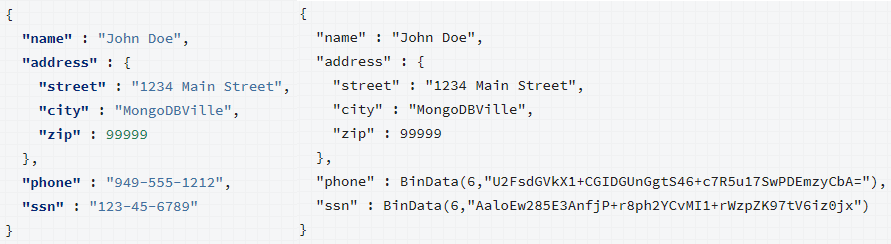
Na ovaj način mongo instanca prezentuje svoj sertifikat klijentu, ali ne vrši verifikaciju identiteta klijenta. Da bi se izvršila validacija sertifikata klijenta, potrebno je u konfiguracionom fajlu dodati i putanju do fajla koji se koristi za verifikaciju sertifikata klijenta (***CAFile*** polje).

Klijenti moraju da imaju podršku za TLS/SSL da bi se konektovali na instancu koja zahteva TLS/SSL konekcije. Prilikom konekcije potrebno je dodati ***--tls*** ili ***--ssl*** opciju, specificirati gde je pokrenuta mongo instanca u okviru ***--host*** opcije, navesti putanju do fajla koji se koristi za verifikaciju sertifikata mongo instance u okviru opcije ***--tlsCAFile*** (***--sslCAFile*** ukoliko se radi sa ssl-om), i, ukoliko instanca zahteva verifikaciju sertifikata klijenta, navesti putanju do sertifikata u okviru ***--tlsCertificateKeyFile*** opcije ***(--sslPEMKeyFile*** ukoliko se radi sa ssl-om), kao što je ilustrovano na narednoj slici.



## Enkripcija polja na klijentskoj strani

MongoDB kompatibilni drajveri omogućavaju enkripciju polja na klijentskoj strani. Aplikacije mogu da izvrše ekripciju polja u dokumentima pre slanja podataka serveru. Samo aplikacije sa pristupom odgovarajućem ključu za enkripciju mogu da dekriptuju i pročitaju zaštićene podatke. Brisanjem ključa za enkripciju svi enkriptovani podaci koji koriste taj ključ postaju trajno nedostupni. Korišćenjem enkripcije polja na klijentskoj strani aplikacija može da izvrši enkripciju i na taj način zaštiti osetljive podatke. Enkriptovana polja se čuvaju kao binarni podaci (***BinData***) sa podtipom 6. Na slici 2.3.2.1 prikazan je izgled dokumenta pre i nakon enkripcije određenih polja.

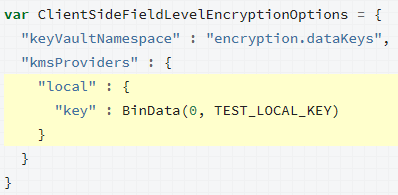


Slika 2.3.2.1 Dokument pre i nakon enkripcije određenih polja

Prvo je potrebno omogućiti enkripciju polja na klijentskoj strani. To se postiže specificiranjem određenih opcija prilikom instanciranja konekcije ka bazi podataka. Potrebno je specificirati:

* ***keyVaultNamespace*** – key vault kolekcija, odnosno kolekcija u kojoj će se čuvati svi ključevi koji se koriste za enkripciju.
* ***kmsProvider*** – Key Managment Service (KMS) koji se koristi za upravljanje CMK-om (Customer Master Key). CMK se koristi za enkripciju i dekripciju ključeva.

Na slici 2.3.2.2 dat je primer korišćenja ovih opcija.



Slika 2.3.2.2 Omogućavanje enkripcije polja na klijentskoj strani

MongoDB podržava dva načina enkripcije polja na klijentskoj strani:

1. ***Eksplicitna enkripcija polja*** – eksplicitna enkripcija i dekripcija polja određenim ključem i algoritmom za enkripciju. Aplikacije moraju da modifikuju kod za čitanje i upis kako bi uključile logiku za enkripciju i dekripciju. Za eksplicitnu enkripciju i dekripciju se koriste sledeće metode:

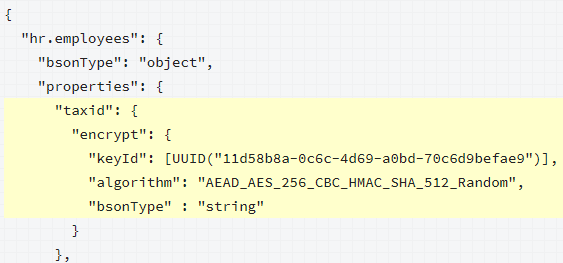
* ***getClientEncryption()*** – vraća ClientEncryption objekat koji se koristi za enkripciju i dekripciju
* ***ClientEncryption.encrypt()*** – vrši enkripciju, kao argumenti se prosleđuju id ključa za enkripciju, vrednost koja se enkriptuje, kao i algoritam za enkripciju
* ***ClientEncryption.decrypt()*** – vrši dekripciju, kao argument se prosleđuje vrednost koju treba dekriptovati

Na sledećoj slici je dat primer enkripcije određenog polja dokumenta prilikom dodavanja u bazu:



1. ***Automatska enkripcija polja*** – opcija dostupna samo u enterprise verziji. Aplikacije moraju da kreiraju objekat za konekciju sa bazom sa odgovarajućim podešavanjima za automatsku enkripciju. Uz opcije navedene na početku ovog dela, potrebno je specificirati koja se polja iz koje kolekcije enkriptuju i na koji način. To se postiže korišćenjem ***schemaMap*** opcije i sledećih ključnih reči:

* ***encrypt*** – specificira opcije za enkripciju određenog polja, ključ i algoritam za enkripciju, kao i tip polja koje se enkriptuje.



* ***encryptMetadata*** – specificira opcije za enkripciju koje se nasleđuju. Definišu se algoritam i ključ za enkripciju. Definisane opcije mogu da nasleđuju polja definisana u okviru *properties*, pri čemu se u okviru *encrypt* opcije za svako polje mora navesti tip tog polja. Ne moraju sva polja da nasleđuju ove opcije, moguće je specificirati drugačije opcije za enkripciju određenih polja.



Kod za čitanje i upis nije potrebno modifikovati.

MongoDB podržava samo AEAD AES-256-CBC algoritam za enkripciju. Ovaj algoritam se može koristiti u dve varijante:

1. ***Deterministic*** – deterministički algoritam za enkripciju osigurava da se određena ulazna vrednost enkriptuje u istu izlaznu vrednost svaki put kada se izvrši algoritam. Deterministička enkripcija podržava operacije čitanja.
2. ***Random*** – randomizirani algoritam za enkripciju osigurava da se određena ulazna vrednost enkriptuje u različitu izlaznu vrednost svaki put kada se izvrši algoritam. Iako randomizirana enkripcija obezbeđuje bolju garanciju za poverljivost podataka, ona ne podržava operacije čitanja koje rade sa enkriptovanim poljem kako bi izvršile upit. Randomizirana enkripcija podržava i enkripciju celih objekata ili nizova.

U okviru metapodataka binarnog tipa podataka BinData, koji se koristi za čuvanje enkriptovanih podataka, čuva se i id ključa korišćenog za enkripciju, kao i korišćeni algoritam. Ovi podaci se koriste kako bi se izvršila automatska dekripcija podataka ovog tipa. Najpre se iz metapodataka uzimaju podaci o korišćenom ključu i algoritmu. Zatim se u key vault kolekciji traži ključ za odgovarajućim identifikatorom. Ukoliko takav ključ ne postoji podaci se vraćaju u binarnom obliku, a ukoliko postoji, vrši se dekripcija podataka.

MongoDB server podržava korišćenje validacije šeme kako bi se osigurala enkripcija specificiranih polja u kolekciji. Za specifikaciju polja koja zahtevaju enkripciju koriste se ključne reči kao za automatsku enkripciju zajedno sa ***$jsonSchema*** validacionim objektom. Server će odbiti bilo koje operacije upisa u okviru te kolekcije gde specificirana polja nisu tipa BinData. Na slici 2.3.2.3 je dat primer korišćenja validacije šeme. ***collMod*** komanda se koristi da bi se modifikovala employees kolekcija tako da sadrži validator. $jsonSchema validacioni objekat uključuje polja koja moraju biti enkriptovana.



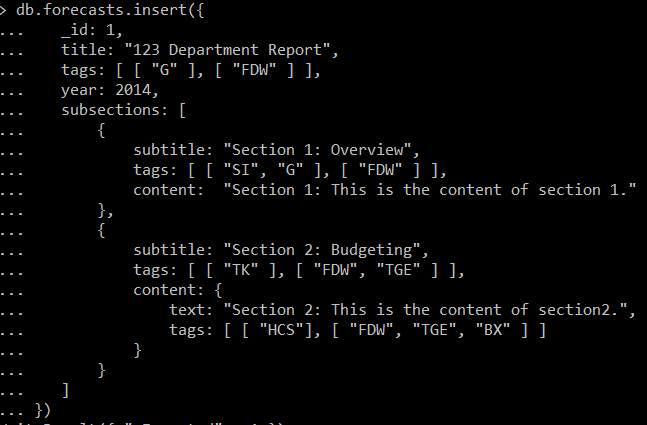
Slika 2.3.2.3 Validacija šeme

## Restrikcija na nivou polja

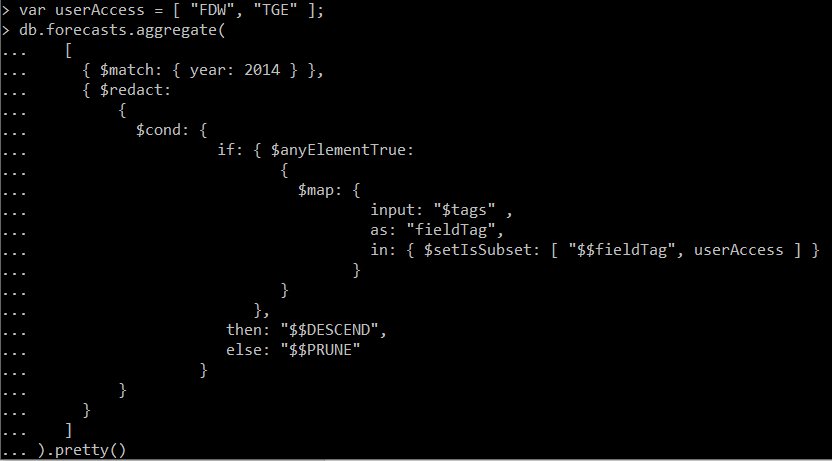
MongoDB podržava restrikciju sadržaja dokumenata na osnovu informacija koje se nalaze u samim dokumentima. Za to se koristi ***$redact*** pipeline operator.

Podaci koji predstavljaju kriterijum za pristup se čuvaju kao polja u dokumentima i ugnježdenim dokumentima. Zatim se koristi $redact operator u okviru ***db.collection.aggregate()*** metoda kako bi se ograničio sadržaj koji se prikazuje korisniku.

Na primer, *forecasts* kolekcija sadrži dokumente oblika prikazanog na slici 2.4.1, gde polje *tags* određuje nivoe pristupa koji su potrebni da bi se videli podaci. Ako korisnik ima pristup samo podacima sa tagovima “FDW” i “TGE”, upit kojim bi se tražili svi dokumenti iz 2014 godine, a koji uključuje i $redact fazu, prikazan je na slici 2.4.2. $redact operator uzima kao argument izraz i, na osnovu vrednosti izraza, vraća kao rezultat ***$$DESCEND*** (vraćaju se samo polja na trenutnom nivou dokumenta, ne uključujući ugnježdene dokumente), ***$$PRUNE*** (isključuju se sva polja na trenutnom nivou dokumenta/ugnježdenog dokumenta) ili ***$$KEEP*** (vraćaju se sva polja na trenutnom nivou dokumenta/ugnježdenog dokumenta). Rezultati ovog upita su prikazani na slici 2.4.3.



Slika 2.4.1 Izgled dokumenata forecasts kolekcije



Slika 2.4.2 Upit sa $redact operatorom



Slika 2.4.3 Rezultati upita sa $redact operatorom

# zaključak

Zaštita baze i podataka koji se čuvaju u njoj je jako bitna. MongoDB nudi različite načine za obezbeđivanje sigurnosti baze podataka.

Autentifikacija i kreiranje različitih korisnika baze kojima se dodeljuju uloge koje obezbeđuju odgovarajuće privilegije omogućavaju zaštitu podataka od neovlašćenog pristupa i zloupotrebe. Takođe, enkripcijom podataka se povećava nivo zaštite podataka. Čak i ako dođe do neovlašćenog pristupa podacima, ukoliko su oni enkriptovani, bez odgovarajućeg ključa neće biti od koristi jer se ne mogu pročitati. Još jedan način zaštite baze podataka koji nudi MongoDB je ograničavanje sadržaja samog dokumenta koji korisnik može da vidi. Na osnovu kriterijuma za pristup se određuje šta korisnik može da vidi, a šta ne.

Zbog svih navedenih mogućnosti za kontrolu sigurnosti baze podataka, MongoDB predstavlja dobro rešenje za čuvanje podataka bez rizika od zloupotrebe i neovlašćenog pristupa.

# literatura

1. Prezentacije sa sajta predmeta
2. Database Management Systems, R. Ramakrishnan and J. Gehrke, McGraw-Hill; 3rd edition, 2002
3. Dokumentacija MongoDB baze podataka

<https://docs.mongodb.com/manual/security/>