UNIVERZITET U NIŠU

ELEKTRONSKI FAKULTET

Katedra za računarstvo

MongoDB sigurnost

Sistemi za upravljanje bazama podataka

Student:

Andrija Petrović 1387

Mentor:

Doc. dr Aleksandar Stanimirović

Sažetak

Bezbednost je veoma važno pitanje u upravljanju bazom podataka, jer informacije uskladištene u bazi podataka mogu biti veoma vredne i često veoma osetljive. Dakle, podaci u sistemu upravljanja bazom podataka moraju biti zaštićeni od zloupotrebe, neovlašćenog pristupa i ažuriranja. Ovaj rad, baziran je na sigurnosti MongoDB baze podataka, koja pripada NoSQL porodici. Najpre su obrazloženi sigurnosni propusti i problemi, uopšteno kod NoSQL tipova baza podataka, a zatim i detaljno razmatranje sigurnosti MongoDB baze podataka. Predmet razmatranja je *MongoDB v. 5.0.8* verzija baze podataka, koja je prethodno instalirana u vidu servera na *Ubuntu 21.10 (impish)* operativnom sistemu.

Sadržaj

1 Bezbednosni problemi kod NoSQL baza podataka

Napredak u računarstvu u oblaku i distribuiranim web aplikacijama stvorio je potrebu za skladištenjem velike količine podataka u distribuiranim bazama podataka koje pružaju visoku dostupnost i skalabilnost.

Poslednjih godina, sve veći broj kompanija je usvojio različite tipove nerelacionih baza podataka, koji se obično nazivaju NoSQL bazama podataka, i kako se pojavljuju aplikacije kojima služe, one dobijaju veliko interesovanje tržišta. Ovi novi sistemi baza podataka nisu relacioni po definiciji i stoga ne podržavaju punu SQL funkcionalnost. Štaviše, za razliku od relacionih baza podataka, one trguju doslednost i bezbednost za performanse i skalabilnost. Kako se sve osetljiviji podaci čuvaju u NoSQL bazama podataka, bezbednosna pitanja postaju sve veća zabrinutost. Bezbednost je jedan od glavnih problema kod relacionih baza podataka, koji su i nerelacione NoSQL baze podataka nasledile.

Glavni razlog koji leži u osnovi upotrebe NoSQL baza podataka jesu performanse i odgovor u realnom vremenu za veliku količinu podataka koji nisu strukturirani kao RDBMS tabele. Ipak, pitanje bezbednosti nije toliko pokriveno jer su se temelji fokusirali na performanse nauštrb drugih oblasti, a bezbednost je jedna od njih. Ovde moramo da razjasnimo da NoSQL baze podataka ne dele istu arhitekturu jer su te baze podataka dizajnirane da obrađuju zahteve aplikacija na web-u.

Danas se većina NoSQL baza podataka suočava sa skoro istim napadima iz RDBMS ere. Čak i sa pružanjem mnogih rešenja za poboljšanje njihove bezbednosti, napadi su napravili ogromne probleme. Napadi SQL injekcijom i ubrizgavanjem skripte su i dalje prisutni kod NoSQL baza podataka. To je direktno povezano sa zavisnošću od specifične strukture podataka u NoSQL bazama podataka (nestrukturirani podaci) kao što su JSON datoteke i mnoge druge, koje se mogu koristiti za zaobilaženje zaštitnih zidova (firewall) i bezbednosnog filtriranja jednostavnim Java-Script kodom, a sistem će to uzeti u obzir kao da su datoteke sa podacima.

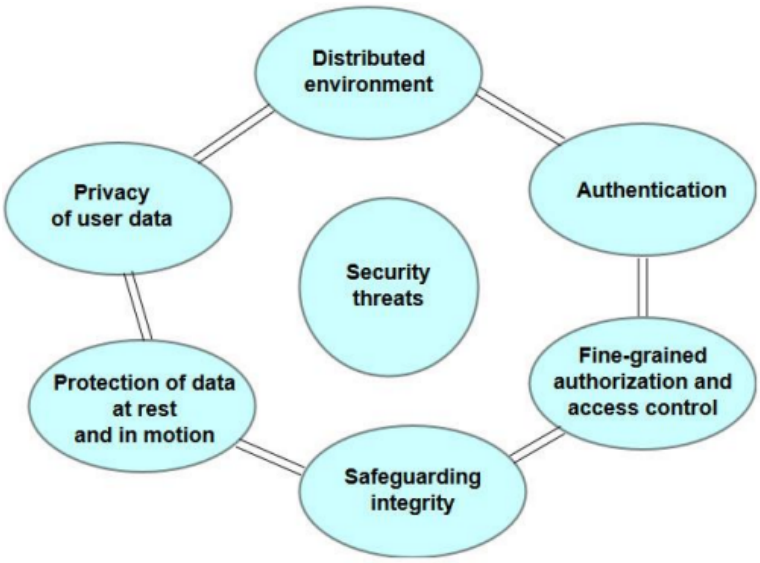
Drugi razlog je distributivno okruženje NoSQL baza podataka koje su razvijene za rukovanje paralelnim računarstvom i deljenjem podataka između čvorova sa velikom verovatnoćom ukradenih podataka. Takođe, to čini da su NoSQL baze podataka podložne velikom području napada, koji komplikuju bezbednosne probleme.

Generalno, možemo nabrojati bezbednosne probleme NoSQL baza podataka na sledeći način:

1. nedostatak šifrovanja datoteka sa podacima
2. ne obezbeđuju dovoljnu autentifikaciju u mrežama klijent-server
3. jednostavna autorizacija
4. ranjivost SQL injekcije
5. filtriranje na ulazu krajnje tačke
6. šifrovanje atributa
7. metodologija odnosa atributa
8. granularni kontroler pristupa
9. nesigurno izračunavanje
10. očuvanje data mining-a i analitike

2 Bezbednosne pretnje kod NoSQL baza podataka

Glavne pretnje kod NoSQL baza podataka obično se dešavaju u middleware-u. Kako bismo obrazložili te pretnje, moramo ih najore klasifikovati. Slika 1 prikazuje bezbednosne pretnje kod NoSQL baza podataka.



Slika 1. Bezbednosne pretnje kod NoSQL baza podataka

2.1 Distribuirana okolina

Većina baza podataka se distribuira putem mreže kako bi se korisniku omogućila dostupnost na daljinu ali i lokalno. Sinhronizaciju pisanja i ažuriranje sadržaja baza podataka kontroliše centralna baza podataka. U distribuiranom okruženju postoji mnogo čvorova raspoređenih po mreži i oni rade paralelno što daje više prostora za napadače. Takođe, održavanje je ovde skupo u smislu da postoji veća šansa da se pojavi greška, a samim tim i mogućnost krađe podataka. Sve to je usko povezano sa činjenicom da ne postoje centralni sistemi upravljanja.

2.2 Autentifikacija

Autentifikacija kod NoSQL baza podataka je obezbeđena na lokalnom čvoru za razliku od servera, što čini NoSQL baze podataka široko izloženim različitim napadima koji dovode do gubitka podataka. Kao primer ove vrste pretnji, imamo Kerberos NoSQL bazu podataka, koja daje autentifikaciju klijentskim čvorovima, ali napadači mogu dobiti neovlašćeni pristup dupliranjem Kerberos tiketa. Kerberos karta je mehanizam za autentifikaciju u mreži koji je razvio MIT. Radi dok korisnik šalje zahtev za prijavu tražeći kartu od Centra za distribuciju ključeva (KDC). Kao potvrdu, KDC će poslati Ticket-Granting Ticket (TGT) za korisnika i kartu šifrovanu korisničkom lozinkom. Korisnik će dobiti TGT, a ako ga korisnik dešifruje sa pravom lozinkom, onda će tiket omogućiti korisniku da se prijavi u

sistem. To je odličan način za autentifikaciju, ali je ipak kritičan, a to je zbog razloga što ga jedna strana nosi, a napadači mogu da probiju mrežnu zaštitu, otmu akreditive i koriste ih za kretanje kroz mrežu, uzimajući dodatne akreditive i eskalirajući privilegije na putu za postizanje svojih ciljeva.

2.3 Zaštita integriteta

Zaštita integriteta je deklaracija zahteva u kojoj baza podataka ne daje neovlašćenu dostupnost korisniku da menja sadržaj podataka, a te operacije mogu biti brisanje, ažuriranje i umetanje. U NoSQL bazama podataka nedostaje i integritet i poverljivost. Sve je to povezano sa razlogom što nemamo šemu ili tabele za određivanje dozvole i ažuriranja na toj tabeli ili redu. U NoSQL bazama podataka možemo se suočiti sa situacijom kada mnogi korisnici modifikuju podatke, a to će dati različite kopije istih podataka. Takođe, to čini transakciju u NoSQL bazama podataka teškom. Zbog toga finansijski sektori ne zavise u potpunosti od NoSQL baza podataka. Nedostaje mu centralna kontrola za takvu vrstu transakcija.

2.4 Granularna autorizacija i kontrola pristupa

Da bismo zaštitili podatke, prvo moramo da potvrdimo njihovu ispravnu upotrebu. Kada korisnik prođe autorizaciju, on će postati deo sistema baze podataka. Međutim, kod NoSQL baza podataka situacija je drugačija jer nedostaje šema i ne može dati ovlašćenje nad tabelama dela podataka. Postoji sigurnost na nivou objekata što znači da nemamo granularitet kod NoSQL baza podataka. Kontrola pristupa bazama podataka zavisi od klasifikacije autorizacije baze podataka. Kontrola pristupa i obračun diskova i memorije se obavljaju na nivou porodice kolona. Ključevi kolona su grupisani u skupove koji se nazivaju porodice kolona, koji čine osnovnu jedinicu kontrole pristupa. Svi podaci uskladišteni u porodici kolona obično su istog tipa. Porodica kolona mora biti kreirana pre nego što podaci mogu da se skladište pod bilo kojim ključem kolone u toj porodici. Nakon što je porodica kreirana, može se koristiti bilo koji ključ kolone unutar porodice. Međutim, kontrola pristupa se razlikuje od jedne baze podataka do druge u zavisnosti od nivoa bezbednosti koji je promovisan korisnicima.

2.5 Zaštita podataka u mirovanju i u tranzitu

Kada završimo rad sa podacima, a zatim ih uskladištimo na jedinicu za skladištenje, ove podatke nazivamo podacima u stanju mirovanja. Kada komuniciramo sa podacima u tranzitu, postoji promena prelaza. Ona je klasifikovana u dva tipa: komunikacija između čvorova i komunikacija klijent-čvor. Većina NoSQL baza podataka ne služi sa metodom zaštite podataka u mirovanju. Neke NoSQL baze podataka pružaju delimičan nivo zaštite podataka u mirovanju i podataka u pokretu.

U MongoDB-u imamo dva metoda šifrovanja za podatke u mirovanju. Kada su podaci u pokretu, MongoDB ne pruža nikakvu komunikaciju klijent-čvor. MongoDB ne pruža nikakvu komunikaciju među čvorovima.

2.6 Privatnost korisničkih podataka

Privatnost je jedan od glavnih izazova sa kojima se suočavaju NoSQL baze podataka, zbog njihove široke primene kod web-baziranih aplikaija. Dok NoSQL baze podataka rukuju značajnom količinom podataka sa brzim odgovorima, privatnost je najaktivnija pretnja u oblasti velikih podataka. Svaki NoSQL sistem baze podataka može da se nosi sa mnoštvom čvorova u vidu korisnika, ali problem nastaje kada se podaci injektuju od strane napadača u sistem, a mogu se proširiti po celom sistemu zbog činjenice da ne postoji centralno upravljanje.

Da bi ublažili te probleme, programeri pokušavaju da ih prevaziđu, obezbeđujući sigurniji mehanizam i metode za zaštitu podataka. Kriptovanje podataka je prva odbrambena linija protiv napadača, koja podrazumeva korišćenje ustaljenih algoritama za zaštitu podataka. Drugi izazov je analiza ekstrakcije podataka. Analiza podataka je dobra za pružanje više informacija o korisnicima kako bi se pomoglo u davanju pravog odgovora na korisnički zahtev. U isto vreme, međutim, postoji rizik da sam analizator povredi privatnost. Poslednji primer je podeljena kontrola pristupa velikim podacima. Kada ne postoji glavni kontroler za protok korisnika kroz mrežu, postoji velika verovatnoća da će napadači pristupiti tim mrežama kao običan korisnik.

3 MongoDB bezbednost

Ovo poglavlje razmatra glavne bezbednosne karakteristike koje se odnose na dizajn MongoDB klastera, uključujući način da se obezbedi sistem na infrastrukturnom nivou i da se šifruju podaci u tranzitu i mirovanju, kao i način da se zaštiti od napada putem koda aplikacije i interfejsa. Konačno, razmotrićemo kako da revidiramo radnje klastera i da sakrijemo evidencije da bismo omogućili analizu performansi bez curenja korisničkih podataka iz baze podataka.

3.1 Uvod

Softver baze podataka MongoDB je bezbedan i poštuje sigurnosne standarde. Jedino što treba znati, jeste kako da podesite, konfigurišete i radite sa MongoDB instalacijom. MongoDB je bezbedan za korišćenje -- ako znate na šta da obratite pažnju i kako da ga konfigurišete. Pre svega, kako ljudi greše sa MongoDB bezbednošću? Postoji nekoliko ključnih oblasti koje "sapliću" korisnike kada je u pitanju MongoDB bezbednost:

- Korišćenje podrazumevanih portova
- Ne omogućavanje autentifikacije odmah (najveći problem!)
- Kada koristite autentifikaciju, davanje širokog pristup svima
- Ne koristite LDAP za prisilno rotiranje lozinki
- Ne forsira upotrebu SSL-a u bazi podataka
- Ne ograničavajući pristup bazi podataka poznatim mrežnim uređajima (hostovi aplikacija, balanseri opterećenja itd.)
- Ne ograničavajući koja mreža sluša (ovo više ne utiče na novije verzije)

3.2 Lokalni pristup

Prvi korak ka obezbeđivanju primene MongoDB-a je da obezbedite ograničen pristup glavnoj mašini, samo na korisnike od poverenja. Pošto su svi podaci na kraju uskladišteni u sistemu datoteka, zlonamerni korisnik sistema bi mogao da pokvari, ukrade, drži kao taoce ili uništi sve podatke uskladištene na lokalnom disku. Svaki operativni sistem ima svoje metode za ograničavanje prijavljivanja na mašinu. Obično bi svaki korisnik imao sopstvene akreditive za autentifikaciju (lozinku ili SSH ključ) i bio bi dodeljen grupi. *Mongod* proces i njegove binarne datoteke i datoteke sa podacima obično se pokreću kao *mongod:mongod* što je podrazumevano kada je MongoDB instaliran na *Linux* mašini. Uvek bi trebalo slediti princip najmanje privilegija, na primer, tako što ćete onemogućiti *root SSH* pristup mašini, ograničiti *sudo* privilegije i evidentirati sve pokušaje povezivanja kako uspešne tako i neuspešne za kasniju reviziju.

3.3 Firewall sa IP-tabelama

Većina Linux verzija uključuje unapred instaliran sistem zaštitnog zida (Firewall). Najčešći ovakav sistem su IP-tabele. Ovo je veoma složen i moćan sistem zaštitnog zida koji može biti teško pravilno konfigurisati. *Red Hat Linux* uključuje neke pomoćne alate kako u grafičkom korisničkom interfejsu tako i u čarobnjaku koji pomaže u generisanju liste pravila za *IP-tabele*. U stvari, možete ručno da konfigurišete svoj zaštitni zid koristeći skup blokova da biste izgradili pravila koja će dozvoliti veze samo sa drugim članovima MongoDB klastera i sa bilo kojim serverima aplikacija, mašinama za programere ili čak sa hostovima krajnjih korisnika.

Da biste dozvolili dolazne veze sa pouzdanih hostova, potrebno je dodati unose u konfiguracione datoteke */etc/sysconfig/iptables* i */etc/sysconfig/ip6tables*:

```
-A INPUT -s 12.34.56.78 -j ACCEPT

-A OUTPUT -d 12.34.56.78 -j ACCEPT
```

Kod 1. Firewalls with iptables

Ovo će omogućiti bilo koji saobraćaj (TCP ili UDP) na bilo kom portu od i do ovog pouzdanog hosta. Za dodatnu sigurnost, takođe možete ograničiti samo na TCP saobraćaj na portu koji koristi MongoDB čvor.

Pored toga, možete ograničiti odlazni saobraćaj samo na veze koje su uspostavljene sa udaljenog hosta:

```
-A INPUT -s 12.34.56.78 -p tcp --destination-port 27017 -m state --state
NEW,ESTABLISHED -j ACCEPT

-A OUTPUT -d 12.34.56.78 -p tcp --source-port 27017 -m state --state ESTABLISHED -j
ACCEPT
```

Kod 2. Limit outgoing traffic

Ovaj konfiguracioni par će vam trebati za svaki host u vašoj implementaciji, uključujući konfiguracione servere, *mongos* instance i bilo koji host koji pokreće aplikaciju koja treba da se poveže sa bazom podataka.

Takođe možete koristiti *firevalld* kao alternativu za *IP-tabele*, kod novijih Linux sistema. Oba koriste okvir *netfilter* za stvarnu obradu paketa, ali *firevalld* može promeniti konfiguracije na već otvorenim vezama. Za rešavanje problema sa vezom na Linux-u, *ss -plnt* može biti koristan za monitoring statistike soketa. Na Windows-u, možete koristiti *netsh* da

otvorite sve potrebne portove između svih komponenti (svih MongoDB čvorova, servera za šard i konfiguraciju i aplikacija). Dokumentacija pruža korak po korak primere za vašu verziju MongoDB-a.

3.4 BindIP

Podrazumevano, najnovije verzije MongoDB-a podrazumevaju konfiguracioni parametar bindIp na 127.0.0.1. To znači da *mongod* proces sluša samo dolazne zahteve sa interfejsa lokalnog hosta i ignoriše sve spoljne mrežne interfejse. Ovo je dobro za testiranje na mašini za razvoj ili pokretanje samostalnog MongoDB-a na istoj instanci servera.

Da biste prihvatili zahtev od eksternog klijenta, trebalo bi da navedete i IP adresu interfejsa, odnosno IP adresu koju je hostu dodelila mreža koja se koristi za rutiranje spoljnog saobraćaja podešavanjem:

```
net:
   bindIp: 12.34.56.78
```

Kod 3. BindIp

Ako iz nekog razloga vašem hostu budu dodeljene dinamičke IP adrese koje se menjaju tokom vremena, možete dozvoliti MongoDB da se veže za sve interfejse dodavanjem:

```
net:
   bindIp: 0.0.0.0
```

Kod 4. bindIpAll

ili

```
net:
   bindIpAll: true
```

Kod 5. bindIpAll

3.5 Kustomizacija portova

Još jedno malo bezbednosno poboljšanje je pokretanje MongoDB instance na nestandardnim portovima. U podeljenom klasteru, podrazumevano, čvorovi koji nose podatke će raditi na portu *27018*, a *mongose* na *27017*.

Premeštanjem ovih procesa na različite portove, malver ili skeneri portova mogu biti sprečeni da otkriju pokrenute MongoDB instance. Oni se mogu konfigurisati u konfiguracionoj datoteci sa:

```
net:
   port:37001
```

Kod 6. Kustomizacija portova

3.6 Sistem datoteka

Na većini Linux instalacija, MongoDB je pokrenut od strane korisnika *mongod* u grupi *mongod*. Stoga, kreira sve datoteke sa podacima i log-ovima kao *mongod* korisnik. Putanja datoteka sa podacima i datoteka evidencije takođe treba da budu u vlasništvu korisnika i grupe, a svi ostali korisnici ne bi trebalo da imaju čak ni pristup za čitanje. Trenutne dozvole se mogu proveriti pokretanjem *ls -lh*, obično na putanji */var/lib/mongodb*. Ako kreirate ključnu datoteku da biste zaštitili server, ona takođe treba da bude u vlasništvu *mongod*-a, a da nema čak ni pristup za čitanje po grupi. Ovo se može postići pokretanjem: *chmod 600 /path/to/keyfile*.

3.7 Autentikacija

Iako je bezbednost na nivou mreže ključni faktor u obezbeđivanju bilo koje baze podataka, ona čini samo jedan deo svakog bezbednog okruženja. Veze iz poznatih izvora i dalje moraju biti ograničene kako bi se sprečilo da kompromitovani hostovi pristupaju podacima sa daljine. MongoDB podržava mnoštvo različitih metoda autentikacije čime zahteva od dolaznih veza da se identifikuju pre izvođenja bilo kakvih operacija baze podataka.

3.7.1 Lozinke naspram ključ-datoteka

Pravljenje korisnika sa kontrolom pristupa zasnovanom na pravilima (RBAC) u MongoDB implementacijama je veoma preporučljivo i trebalo bi da kreirate najmanje jedan korisnički nalog čim počnete da skladištite bilo kakve stvarne podatke u svojoj bazi podataka.

Kao i kod većine baza podataka, možete kreirati korisničko ime i lozinku kojima se može odobriti pristup većem broju baza podataka ili kolekcija. Korisnici mogu biti ograničeni na određene uloge kao što je „samo za čitanje“ ili ograničeni da bi mogli da konfigurišu skup replika, ali uopšte ne mogu da vide nikakve podatke. Uvek treba slediti princip „najmanje privilegije“ i dodeliti samo uloge koje su ovom konkretnom korisniku potrebne da dovrši svoje slučajeve korišćenja.

Autentifikacija se može omogućiti dodavanjem sledećih redova u konfiguracioni fajl i zatim ponovnim pokretanjem MongoDB čvora:

```
security:
   authorization: enabled
```

Kod 7. Autentifikacija

Sistemski administrator može privremeno da onemogući kontrolu pristupa ponovnim pokretanjem *mongod* procesa bez parametra *--auth* ili konfiguracije *security.authorization*. Ovo može biti korisno za održavanje i resetovanje lozinke ako se izgubi administratorska lozinka.

Povezivanje sa lozinkama: Kada se povezujete na MongoDB iz drajvera ili alata za istraživanje podataka kao što je MongoDB Compass ili Mongo shell, potrebno je da unesete i korisničko ime i lozinku ili preko posebnog polja ili u URI-ju veze u obliku:

```
mongodb://[username:password@]host1[:port1][,...hostN[:portN]][/[database]
[?options]]
```

Kod 8. Povezivanje sa lozinkama

Jedan od problema korišćenja lozinki u primeni je taj što će skripte morati da čuvaju lozinke za povezivanje sa MongoDB. Ovo predstavlja bezbednosni rizik pošto lozinke postoje nešifrovane unutar datoteka skripta (i kontrole izvora) i mogu se pojaviti na listi procesa (npr. *ps* na Linux-u), a mogu biti i nesigurno uskladištene na ličnim računarima korisnika.

Povezivanje sa ključ-datotekama: Ključ-datoteka je u suštini veoma duga lozinka uskladištena u bezbednoj datoteci i na koju upućuje svaki mongod čvor u primeni. Deluje kao „deljena tajna“ između komponenti i omogućava im da se autentifikuju prilikom uspostavljanja skupa replika ili kada dodaju potpuno novi čvor postojećem skupu. Bez ove zajedničke tajne, hakeri bi mogli da dodaju novi prazan čvor pod svojom kontrolom u postojeći skup replika i pokrenu početnu sinhronizaciju. Zatim bi se uključili u arhitekturu replikacije MongoDB da bi napravili kopiju baze podataka.

Sadržaj ovih datoteka treba tretirati kao lozinku i pristup za čitanje ograničen na nivou sistema datoteka. Svako ko ima sadržaj mogao bi da lažira pristup skupu replika, tako da je ovo minimalni bezbednosni pristup za vašu primenu. Možete uputiti MongoDB da zahteva ključnu datoteku za internu autentifikaciju tako što ćete navesti putanju na sledeći način i ponovo pokrenuti server.

```
security:
   keyFile: /path/secure.key
```

Kod 9. Povezivanje sa ključ-datotekama

3.7.2 x.509 sertifikati

Vrhunski, ali složeniji metod od korišćenja ključ-datoteke je kreiranje i primena *x.509 sertifikata* za komponente vašeg servera i klijenta. U vreme pisanja, postoji pogodna besplatna lokacija pod nazivom Let's Encript koja olakšava generisanje *x.509 sertifikata* koji se mogu koristiti sa MongoDB, ali zahteva javno adresirana DNS imena hostova. Ovi sertifikati traju samo 90 dana, tako da je neophodno sktiptovati njihovo generisanje i obnavljanje. Dok se *x.509 sertifikati* mogu koristiti za obezbeđivanje interne komunikacije između MongoDB komponenti klastera, malo drugačiji format sertifikata se može koristiti za klijente aplikacije da se autentifikuju sa klasterima. Ako odaberete da koristite ovaj metod, onda svaki drugi korisnički nalog koji se povezuje mora da ima jedinstveni sertifikat kreiran za njega sa korisničkim imenom ugrađenim u metapodatke.

3.7.3 Autentifikacija klijenta

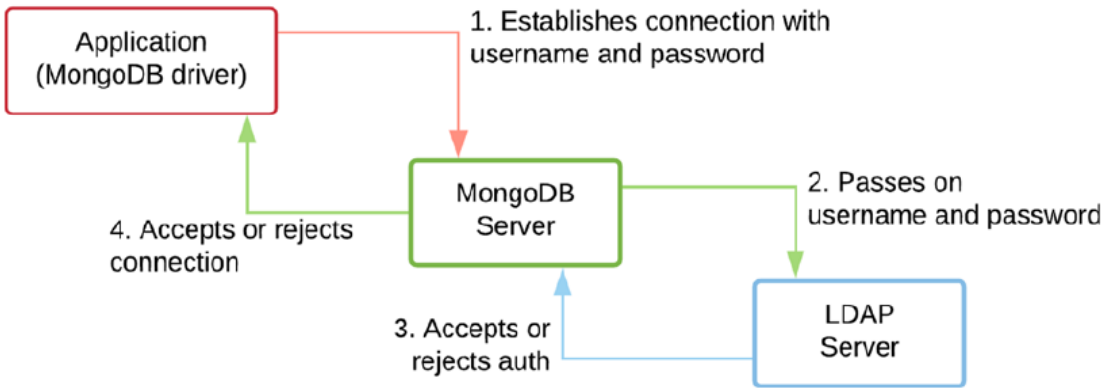
MongoDB ima odvojene podsisteme za autentifikaciju i autorizaciju. Autentifikacija se može izvršiti korišćenjem ugrađenog metoda autentifikacije koji se zove *Salted Challenge Response Authentication Mechanism* (SCRAM) ili preko spoljnog sistema za autentifikaciju koji podržava *Lightweight Directory Access Protocol* (LDAP) ili Kerberos. Sa SCRAM-om, korisnici i njihovi akreditivi se čuvaju šifrovani u admin bazi podataka (i automatski se repliciraju na sve članove u primeni).

Autorizacija je sledeći korak nakon što je autentifikacija uspešna, u kojem će MongoDB proveriti uloge i privilegije autentifikovanog korisnika pre nego što dozvoli pokretanje komande. Ovlašćenje se može dodeliti na nivou baze podataka ili prikupljanja po potrebi. Dozvole za više unapred definisanih uloga mogu se dodati zajedno da bi se postigla prava kombinacija.

3.7.4 LDAP autentifikacija

MongoDB ima ugrađene korisničke uloge i podrazumevano ih isključuje. Međutim, nedostaju mu stavke kao što su složenost lozinke, rotacija zasnovana na uzrastu, centralizacija i identifikacija korisničkih uloga u odnosu na uslužne funkcije. Ovo je neophodno za donošenje propisa kao što je usklađenost sa PCI DSS. Na primer, PCI DSS zabranjuje upotrebu starih lozinki i lozinki koje je lako razbiti i zahteva promene korisničkog pristupa kad god se status promeni (na primer, kada korisnik napusti odeljenje ili kompaniju). Srećom, LDAP se može koristiti za popunjavanje mnogih od ovih praznina. Mnogi konektori omogućavaju korišćenje sistema Windows Active Directory (AD) za

razgovor sa LDAP-om. LDAP podrška je dostupna samo u MongoDB Enterprise. Dostupan je i u drugim verzijama MongoDB otvorenog koda kao što je Percona Server za MongoDB. Nije u verziji zajednice.



Slika 2. LDAP

Koristeći LDAP i AD, možete povezati korisnike sa vašim korporativnim direktorijumom. Kada promene uloge ili napuste kompaniju, ljudski resursi mogu da ih uklone iz vaše grupe baze podataka. Dakle, imate automatizovan sistem koji obezbeđuje da samo oni kojima želite da pristupe podacima ručno to mogu da urade, a da slučajno nešto ne propuste. LDAP u MongoDB-u je zapravo lak. MongoDB ima posebnu komandu koja vrši spoljnu proveru LDAP baze podataka: *$external*.

Mane korišćenja LDAP:

- Kreirajte korisnika sa .*createUser* kao i obično, ali obavezno koristite oznake resursa *db/collection*.
- Pored toga, LDAP autentifikacija zahteva još dva polja:
  - ➔ mechanism: “PLAIN”
  - ➔ digestPassword: false

3.7.5 Kerberos autentifikacija

Ovo je protokol za autentifikaciju tajnog ključa za interakcije između servera i klijenta. Koristeći Kerberos, korisnici se mogu prijaviti samo jednom koristeći pristupnu kartu.

3.8 Šifrovane konekcije

Čak i ako svi MongoDB čvorovi i aplikacije postoje unutar iste bezbedne mreže, i dalje je dobra ideja da razmislite o šifrovanju svih komunikacija. Ovo može sprečiti prisluškivanje mrežnog saobraćaja ka/sa vaše MongoDB

instance i olakšati skaliranje ako želite da migrirate kasnije. Na primer, možda ćete želeti da postepeno pređete sa potpuno lokalnog podešavanja na hibridno rešenje sa nekim čvorovima hostovanim na provajderu u oblaku.

Za bilo koju proizvodnu upotrebu, već bi trebalo da razmišljate o konfiguraciji više centara podataka za primenu MongoDB-a sa fizički odvojenim centrima podataka. Čak i sa VPN-ovima/VPC-ovima, skoro uvek ćete se oslanjati na deljene internet okosnice između centara podataka za sinhronizaciju ovih komponenti, pa su opet šifrovane veze kritične da bi vaši podaci bili bezbedni.

MongoDB Community izdanje uključuje potpunu enkripciju za podatke u tranzitu koristeći TLS/SSL (šifrovanje podataka). Pošto su šifrovane veze kritične za bezbedan sistem, nedavne verzije MongoDB-a su podrazumevano uklonile podršku za starije, manje bezbedne TLS 1.0 sisteme. Možete da koristite bilo koji važeći sertifikat od zvaničnog autoriteta za izdavanje sertifikata ili možete da napravite sopstvene samo-potpisane sertifikate od nule koristeći *openssl* alatku komandne linije. Sa bilo kojim pristupom, tok podataka će biti šifrovan, ali bez potvrde identiteta da su vaši sertifikati potpisani od strane nezavisnog autoriteta za sertifikaciju, vaša komunikacija može biti ranjiva na takozvani napad čovek u sredini.

Čak i kada ste konfigurisali TLS za svoju internu MongoDB komunikaciju, morate eksplicitno da omogućite šifrovanje iz vaše aplikacije u MongoDB dodavanjem *?ssl=true* u string veze. Ako želite da zahtevate da svi klijenti koriste TLS da bi mogli da se povežu sa vašim MongoDB serverom, možete ovo dodati u svoju MongoDB konfiguraciju:

```
net:
   tls:
      mode:requireTLS
```

Kod 10. TLS

U proizvodnim okruženjima, trebalo bi da koristite važeće sertifikate za sve komponente u primeni MongoDB koje je potpisao isti autoritet za sertifikate (bilo interni za celu organizaciju ili TLS dobavljač treće strane od poverenja). Neke organizacije imaju stroge zahteve da se TLS 1.2 koristi za šifrovanje svih veza. Ako je to slučaj, možete eksplicitno da onemogućite TLS 1.0 i 1.1 u MongoDB, dodavanjem parametra *disabledProtocols* na bilo koji MongoDB server sa već omogućenim *SSL/TLS*:

```
net:
   tls:
      disabledProtocols:TLS1_0, TLS1_1
```

Kod 11. Disable TLS 1.0 i 1.1

U mnogim drajverima takođe možete forsirati TLS 1.2 za odlazne veze sa aplikacijama. Na primer, sa Java drajverom možete da primenite TLS 1.2 tako što ćete izgraditi vezu sa sledećim kodom:

```
SSLContext sslContext = SSLContext.getInstance("TLSv1.2");
MongoClientOptions options = MongoClientOptions.builder()
          .sslEnabled(true)
          .sslInvalidHostNameAllowed(true)
          .socketFactory(sslContext.getSocketFactory())
          .build();
```

Kod 12. Forsiranje TLS 1.2

3.9 Šifrovanje u mirovanju

Podrazumevano, MongoDB čuva svoje podatke na disku u obliku Wired-Tiger binarnih datoteka kompresovanih snappy kodekom. Ako bi neovlašćena strana dobila pristup ovim datotekama (ili njihovoj rezervnoj kopiji) bez enkripcije, mogla bi lako pokrenuti sopstveni lokalni MongoDB proces koji ukazuje na te datoteke i pokrenuti bazu podataka, sa onemogućenom autentifikacijom. Tada bi se svaki MongoDB klijent mogao povezati da pregleda ili upita te podatke. Da bi se zaštitila od ove vrste napada, svaka MongoDB baza podataka koja sadrži stvarne korisničke podatke treba da šifruje podatke u mirovanju.

Šifrovanje podataka u mirovanju može se rešiti bilo kojim od sledećih pristupa:

1. Šifrovanje diska
2. Šifrovanje iz aplikacije

Prva stavka se može rešiti šifrovanjem diska na sistemu datoteka. Ako sami hostujete na nekom cloud rešenju, trebalo bi da imate pristup enkripciji na nivou sistema datoteka koja koristi ugrađene usluge upravljanja glavnim ključem.

Alternativno, sve najnovije verzije Linux-a za preduzeća uključuju rešenja za šifrovanje celog diska kao što je *Linux Unified Key Setup-on-disk-format* (ili LUKS). Ovo takođe koristi glavni ključ za šifrovanje čitavih volumena skladišta ili podskupa. Trebalo bi da čuvate svoje MongoDB datoteke sa podacima na posebnom volumenu na root particiji (da biste omogućili ciljano podešavanje performansi), a ovo takođe čini šifrovanje na nivou sistema datoteka praktičnom opcijom.

Međutim, ako ne želite da šifrujete na nivou diska, onda je šifrovanje unutar samog servera baze podataka druga opcija. Šifrovanje u mirovanju je opcioni modul za Enterprise verziju MongoDB-a. Ovaj proces šifrovanja se dešava potpuno automatski. Na disk se uvek zapisuju samo šifrovani podaci, dok su nešifrovani podaci dostupni samo u memoriji host-a.

3.10 Korišćenje SSL-a

SSL pomaže da se osigura bezbednost vaših podataka preko nebezbednih mreža. Ako koristite bazu podataka koja je u interakciji sa internetom, trebalo bi da koristite SSL. Postoje dva veoma dobra razloga za korišćenje SSL-a za obezbeđenje MongoDB-a: privatnost i autentifikacija. Bez SSL-a, vašim podacima se može pristupiti, kopirati ih i koristiti u nezakonite ili štetne svrhe. Sa autentifikacijom, imate sekundarni nivo bezbednosti. Infrastruktura privatnog ključa SSL-a (PKI) garantuje da samo korisnici sa ispravnim CA sertifikatom mogu pristupiti MongoDB-u. MongoDB već dugo ima podršku za SSL, ali je dramatično poboljšao SSL podršku u poslednjih nekoliko verzija. Ranije, ako ste želeli da koristite SSL, morali ste da ga preuzmete i sastavite ručno sa verzijom MongoDB zajednice.

Od MongoDB 3.0 verzije, SSL se podrazumevano kompajlira sa softverom. Zastarelim verzijama MongoDB-a takođe je nedostajala validna provera host-a; validacija host-a je bila samo oznaka, koju ste mogli da proverite u konfiguracionoj datoteci koja je zadovoljila SSL zahtev iz veze.

Najnovije verzije SSL-a u MongoDB-u uključuju sledeće ključne karakteristike:

- Proverava važeće hostove (opciono)
- Sposobnost da se pokaže na određenu datoteku .key za podešavanja koju treba koristiti
- Prilagođeni autoritet sertifikata (CA) za samopotpisane sertifikate ili alternativne potpisnike
- allowSSL, preferSSL, requireSSL modove, koji omogućavaju izbor granularnosti za upotrebu SSL-a (od manje bezbednog do bezbednijeg)

3.11 Revizija, maskiranje log-ova

Od MongoDB 2.6, Enterprise verzija MongoDB-a uključuje funkciju revizije. Kada je revizija omogućena na instancama *mongod* ili *mongos*, ovo omogućava administratorima baze podataka da prate sve ključne promene složenih implementacija. Ovi zapisnici revizije mogu biti kritični prilikom obavljanja obdukcije nakon bilo kakvog bezbednosnog događaja ili kao deo redovnih provera usklađenosti. U ovim situacijama, evidencija revizije može da pokaže sve promene u ulogama korisnika, kreirane nove korisnike ili neočekivani pristup proizvodnim bazama podataka.

Ne postoji ekvivalentna verzija ove funkcije revizije u zajednici, iako su treće strane kreirale rešenja koja repliciraju neke osnovne karakteristike, uključujući filtriranje izlaza određenim korisnicima, bazu podataka, kolekciju ili lokaciju izvora.

Funkcionalnost evidencije revizije treba eksplicitno da bude omogućena u konfiguracionoj datoteci MongoDB i može da se prenese u datoteku, konzolu ili na kompatibilno odredište sistemskog dnevnika za eventualno skladištenje na centralizovanom serveru evidencije radi dodatne bezbednosti. Jednom kada se aktivira, podrazumevano će snimati zapisnik za svaku radnju promene bilo kog korisnika u primeni. Filteri se takođe mogu konfigurisati da pojednostave evidenciju revizije.

```
auditLog:
   destination: file
   format: BSON
   path: /var/lib/mongodb/auditLog.bson
```

Kod 13. Log

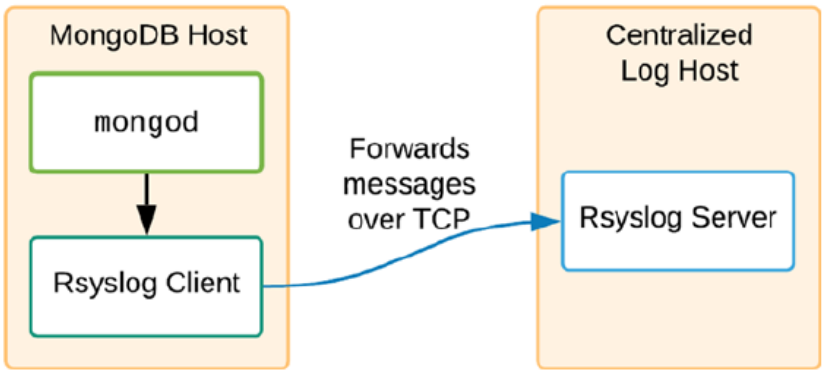
Čuvanje evidencije revizije na istom sistemu datoteka, na kome je i čvor sa sopstevnim podacima, ima značajne nedostatke. Na primer, datoteka evidencije mora da ima dozvole za čitanje/pisanje od strane *mongod* procesa. Ako server bude kompromitovan od strane eksternog hakera ili čak člana osoblja sa akreditivima, oni bi mogli da prikriju svoje tragove uklanjanjem dokaza iz *auditLog.bson* fajla ili potpuno brisanjem fajla čime bi kasnija obdukcija bila nemoguća.

Drugi pristup bi bio da se proverava evidencija kroz *syslog* i instalira sistem za obradu dnevnika kao što je *rsyslog*, koji može da obrađuje različite izvore za generisanje dnevnika i da ih šalje u različite izlaze. Ovo bi uključivalo postavljanje *rsyslog* klijenta na MongoDB host mašini i centralizovani *rsyslog* server koji prihvata poruke dnevnika sa svakog host-a u implementaciji preko otvorenog TCP porta.

Sa strane MongoDB-a, ovo zahteva promenu postavki konfiguracije:

```
auditLog:
   destination: syslog
```

Kod 14. Syslog



Slika 3. Auditing log messages

Podrazumevano, datoteke evidencije nisu ni šifrovane ni maskirane, ali postoji još jedna Enterprise funkcija za uređenje mogućih korisničkih podataka koji ulaze u datoteke evidencije. Omogućava se na sledeći način:

```
security:
   redactClientLogData: true
```

Kod 15. Omogućenje maskiranja

u konfiguracionoj datoteci ili u toku rada na sledeći način:

```
db.adminCommand(
{ setParameter: 1, redactClientLogData : true | false }
)
```

Kod 16. Omogućenje maskiranja u toku rada

3.12 Prilagođene uloge

Kontrola pristupa zasnovana na ulogama (RBAC) je jezgro MongoDB-a. Ugrađene uloge su dostupne u MongoDB od verzije 2.6. Možete čak i da napravite svoje, sve do konkretnih radnji koje bi određenom korisniku moglo da bude dozvoljeno. Ovo vam omogućava da precizno definišete šta određeni korisnik može da uradi ili vidi. Ovo je osnovna funkcija MongoDB-a koja je dostupna u skoro svim verzijama softvera otvorenog koda.

Pet glavnih ugrađenih MongoDB uloga kojih treba biti svestan:

- ***read***:

Read-only pristup, obično se daje većini korisnika

- ***readWrite***:
  readWrite pristup, omogućava uređivanje podataka
- ***dbOwner***:
  Uključuje *readWrite*, *dbAdmin*, *userAdmin*. *userAdmin* znači dodavanje ili uklanjanje korisnika, dodeljivanje privilegija korisnicima i kreiranje uloga. Ove privilegije se dodeljuju samo određenom serveru baze podataka.
- ***dbAdminAnyDatabase***:
  Kreira *dbAdmin* u svim bazama, ali ne dozvoljava administraciju korisnika (na primer, kreiranje ili uklanjanje korisnika). Možete kreirati indekse, zbijanje poziva i još mnogo toga. Ovo ne pruža pristup za deljenje.
- ***root***:
  Ovo je super-korisnik, ali sa ograničenjima. Može da uradi većinu stvari, ali ne sve: Ne može da promeni sistemsku kolekciju. Neke komande još uvek nisu dostupne ovoj ulozi, u zavisnosti od verzije. Na primer, MongoDB 3.2 *root* uloga ne dozvoljava da promenite veličinu *oplog*-a ili *profiler*-a, a MongoDB 3.4 *root* uloga vam ne dozvoljava da čitate trenutne prikaze.

Moć MongoDB uloga dolazi od kreiranja prilagođenih uloga. U prilagođenoj ulozi možete odrediti da se bilo koja radnja na bilo kom resursu može navesti za određenog korisnika. Sa ovim nivoom granularnosti, možete duboko kontrolisati ko može šta da radi u vašem MongoDB okruženju.

Kada je u pitanju određivanje prilagođene uloge, postoje četiri različita tipa resursa:

- **db.** Određuje bazu podataka. Možete da koristite string za ime ili „“ za „bilo koji“ (bez džokera).
- **collection.** Određuje kolekciju dokumenata. Možete koristiti string za ime ili „“ za „bilo koji“ (bez džokera).
- **cluster.** Određuje podeljeni klaster ili druge resurse metapodataka. To je logička vrednost tačno/netačno.
- **anyResource.** Određuje pristup bilo čemu, bilo gde. To je logička vrednost tačno/netačno.

Svaka uloga može naslediti svojstva druge uloge. Postoji niz koji se zove „uloge“ i možete ubaciti novu ulogu u taj niz. On će naslediti svojstva navedene uloge. *createRole* se koristi za dodavanje uloge u niz. Takođe je moguće dodati nove ili postojeće baze podataka korisniku ili ulozi.

Na primer, možete dodati pristup za čitanje i pisanje bazi podataka dodavanjem baze podataka ulozi. Koristite komandu *grantPrivilegesToRole* da dodate nove resurse postojećoj ulozi. Ispod je primer kreiranja nove uloge super-korisnika. Svrha ove uloge je, opet, da ima jednog korisnika koji ni na koji način nije ograničen u MongoDB okruženju (za hitne situacije).

```
db = db.baza("admin");
db.createRole({
     role: "superRoot",
     privileges:[{
          resource: {anyResource:true},
          actions: ['anyAction']
     }]
     roles:[]
});
db.createUser({
     user: "user1",
     pwd: "EWqeeFpUt9*8zq",
     roles: ["superRoot"]
})
```

Kod 17. Super-korisnik

Ove komande kreiraju novu ulogu u bazi podataka pod nazivom *superRoot* i dodeljuju toj ulozi bilo koji resurs i bilo koju radnju. Zatim kreiramo novog korisnika u istoj bazi podataka pod nazivom *user1* (sa lozinkom) i dodeljujemo mu novu *superRoot* ulogu.

3.13 Šifrovanje diska

Podaci su ili „u tranzitu“ ili „u mirovanju“. Mogu se šifrovati pojedinačno ili zajedno u MongoDB instanci. Podaci u stanju mirovanja su podaci koji se čuvaju na disku. Šifrovanje podataka u mirovanju se obično odnosi na podatke sačuvane na šifrovanoj lokaciji za skladištenje. Ovo je da bi se sprečila krađa fizičkim sredstvima i stvorile rezervne kopije koje se čuvaju na adekvatan način, kako treća strana ne može lako da ih pročita. Za ovo postoje praktična ograničenja. Najveće je poverenje sistemskim administratorima - i pod pretpostavkom da haker nije dobio administrativni pristup sistemu. Ovo nije problem jedinstven za MongoDB. Preventivne mere koje se koriste u drugim sistemima deluju i ovde. Oni mogu uključivati alate za šifrovanje kao što su LUKS i *cryptfs* ili još sigurnije metode kao što je potpisivanje ključeva za šifrovanje pomoću LDAP-a, pametnih kartica i tokena RSA tipa.

Kada izvodite ovaj nivo šifrovanja, morate uzeti u obzir faktore kao što su automatsko montiranje i dešifrovanje disk jedinica. Administratori sistema mogu da upravljaju ovim zahtevom na isti način na koji njime upravljaju u drugim delovima mreže. Dodatna prednost je jedna procedura za šifrovanje skladišta, a ne jedna po bilo kojoj tehnologiji koju određena funkcija koristi.

4 MongoDB proaktivna bezbednost

Za svaku bezbednu mrežu, trebalo bi imati redovne preglede fizičke bezbednosti, skeniranje ranjivosti, testiranje penetracije i automatsko upozorenje o nenormalnim obrascima saobraćaja na mreži virtuelnih mašina. Ovo bi trebalo da omogući brzu intervenciju u slučaju bilo kakvog kršenja bezbednosti. MongoDB podržava većinu verzija oko 2 godine nakon objavljivanja, sa redovnim bezbednosnim zakrpama. Trebalo bi brzo da izvršite nadogradnju na bilo koja manja izdanja jer ona često uključuju bezbednosna poboljšanja, ali nikada ne uvode promene u kompatibilnosti. Takođe biste mogli da razmislite o prelasku na najnoviju glavnu nadogradnju verzije u roku od 6 meseci od objavljivanja da biste imali koristi od drugih značajnih poboljšanja bezbednosti i stabilnosti. Takođe možete pratiti Alerts stranicu, gde će biti postavljena kritična upozorenja i saveti.

4.1 Validacija unosa i napadi injekcijom

Pošto je MongoDB zasnovan na JSON dokumentima, ne postoji rizik od tradicionalnih napada SQL injekcije. Međutim, postoji mogućnost napada JSON injekcijom, posebno kada se pokreće JavaScript/NodeJS web aplikacija (kao što je Mongoose ORM) ili API-ji. Zamislite API koji prihvata JSON dokument kao deo upita. Ako bi ovaj upit bio prosleđen direktno upitu za pronalaženje MongoDB-a, zlonamerni korisnik bi mogao da koristi trikove MongoDB jezika upita za upite podataka koji pripadaju drugim korisnicima, poznatim kao „ubrizgavanje objekta“. Poznati napadi koriste kombinaciju *$not* i *$ne* da zaobiđu normalne provere.

Zamislite da ste imali ekran za prijavu koji je prihvatio unos web pretraživača u formu i to je prosleđeno direktno u upit kao npr. :

```
{user: 'nic', password; '123'}
db.users.findOne(query)
```

Kod 18. Validacija unosa

Ako *findOne* vrati rezultat sa odobrenjem, autentifikovali smo korisnika i učitali njegove podatke u jednom koraku. Ako bi haker umesto toga poslao:

```
{user: 'nic', password: {$ne: ''}}

     ili

{user: 'nic', password: {$gt: ''}}
```

Kod 19. Injection

ovo bi takođe vratilo dokument u *findOne* upit, a da haker ne mora ni da pokuša da pogodi lozinku. Naravno, ova vrsta napada se ne odnosi samo na čitanje neovlašćenih podataka. Moglo bi biti moguće čak i umetanje i ažuriranje u bazu podataka iz aplikacija koje ne potvrđuju JSON podatke. Stoga, za dizajnere aplikacija koji prihvataju JSON dokumente u API upitima, od ključne je važnosti da validiraju takve dokumente i izdvoje vrednosti u njihovom očekivanom tipu podataka.

Za NodeJS, postoje neke biblioteke kao što je Joi koje mogu pomoći. Druge opcije uključuju korišćenje *JSON.parse*, korišćenje *npm* paketa kao što je *mongo-sanitize* ili ručno uklanjanje znakova dolara ili vitičastih zagrada u API unosu pre upotrebe podataka.

4.2 $where operator

$where operator je stara funkcija upita za pronalaženje koja omogućava da prosledite ili string koji sadrži Java-Script izraz ili punu Java-Script funkciju u podsistem upita. Očigledno je dozvoliti da se Java-Script komande prosleđuju unutrašnjem Java-Script mehanizmu uključenom u MongoDB, opasno, posebno ako postoji rizik da bi to moglo doći od nepotvrđenog korisničkog unosa.

Dodata su različita bezbednosna poboljšanja i ograničenja kako bi se ograničio njegov uticaj, ali suština je da je ovo veoma opasna karakteristika i da se skoro svaki originalni slučaj upotrebe može rešiti drugim novijim, efikasnijim, bezbednim funkcijama MongoDB-a. U najnovijim verzijama MongoDB-a, ova opcija je podrazumevano omogućena, ali preporuka jeste da je treba onemogućiti u svim primenama dodavanjem:

```
security:
   javascriptEnabled:false
```

Kod 20. $where operator

u konfiguracionu datoteku ili pomoću opcije komandne linije *--noscripting*. Ovo će takođe onemogućiti stare funkcije *mapReduce* i *db.collection.group* – koje se mogu zameniti u aplikaciji korišćenjem cevovoda za agregaciju.

4.3 SELinux

Security-Enhanced Linux (poznat kao SELinux) je Linux modul koji obezbeđuje mehanizam za podršku bezbednosnih politika kontrole pristupa, kao što su obavezne kontrole pristupa. Jednom omogućen, SELinux može sprečiti bilo koji pojedinačni program/uslugu na Linux hostu da kompromituje ceo sistem ograničavanjem pristupa određenim unapred odobrenim direktorijumima, datotekama i mrežnim portovima.

Kada podešavate MongoDB, možete pokrenuti SELinux u dva različita režima. Dozvoljeni režim je dobar za otklanjanje grešaka u vašoj konfiguraciji jer će omogućiti MongoDB-u da pokuša sve pristupe, ali će evidentirati svaki pokušaj. Kada prebacite SELinux u režim sprovođenja, on će blokirati svaki neovlašćeni pristup i možda sprečiti da se vaša MongoDB instanca ponaša kako se očekuje.

Možete da proverite trenutni režim pokretanjem *getenforce* ili da vidite pun status sa *sestatus*. Da biste primenili podrazumevani režim pri pokretanju sistema, obično ćete morati da uredite datoteku */etc/selinuk/config*. Kada se MongoDB instalira preko unapred upakovanih datoteka, postoje određeni standardni direktorijumi koji se koriste za skladištenje podataka i datoteka evidencije:

```
/var/lib/mongo (data directory)
/var/log/mongodb (log directory)
```

Kod 21. Data i log fajlovi

Podrazumevana SELinux konfiguracija za MongoDB koja je uključena u *Red Hat Enterprise Linux* pretpostavlja ove iste direktorijume, tako da ako izaberete da konfigurišete svoj MongoDB za skladištenje podataka ili evidencije na drugoj lokaciji, moraćete ručno da ažurirate SELinux dozvole sa nizom komandi.

Na primer, ako želite da uskladištite svoje podatke u novoj particiji */data/mongodb*, moraćete da pokrenete sledeće komande da biste omogućili MongoDB da se pokrene i pravilno funkcioniše:

```
semanage fcontext -a -t mongod_var_lib_t '/data/mongodb.*'
semanage port -a -t mongod_port_t -p tcp <portnumber>
chcon -Rv -u system_u -t mongod_var_lib_t '/data/mongodb'
restorecon -R -v '/data/mongodb'
```

Kod 22. SELinux

Ako SELinux sprečava da se MongoDB instanca pravilno pokrene, sve detalje će evidentirati u prilagođenom formatu u lokalnu */var/log/audit/audit.log* datoteku. Ako naiđete na bilo kakve probleme prilikom konfigurisanja vašeg MongoDB-a sa primenom SELinux-a, postoji nekoliko alatki i komandi koje vam mogu pomoći da objasnite problem na osnovu ovog *audit.log*. *audit2allov* može pomoći u generisanju novog skupa SELinux smernica koje možete da uvedete sa višim prioritetom od podrazumevanih vrednosti da biste zamenili ove uključene smernice.

Alternativno, pokretanje:

```
sudo sealert -a /var/log/audit/audit.log
```

Kod 23. SELinux promena konfiguracije

će aktivirati klijentski alat, za rešavanje problema i obično dati neke korisne uvide i preporuke za promene same konfiguracije SELinux-a.

Zaista vredi uložiti napore da omogućite SELinux i pokrenete svoju MongoDB proizvodnu instancu u režimu primene. Ako je moguće, držite se podrazumevanih portova i putanja direktorijuma jer će zadržavanje ovih standarda takođe pomoći novim članovima tima da se uključe i pojednostave svako kasnije rešavanje problema.

4.4 Binarni monitoring

Postoji nekoliko rešenja dizajniranih da dodaju dodatni sloj zaštite aplikacijama, koje se koriste u kritičnim infrastrukturama kao što su banke ili bolnice. Ovi alati manipulišu binarnim kodom i zamenjuju određene operacije (kao što je dodela memorije) sopstvenim funkcijama. Ideja je da se prate i spreče određeni vektori napada. Razne kompanije prodaju ovakve alate, za koje se tvrdi da su kompatibilni sa MongoDB-om, ali moje ograničeno iskustvo sa ovim alatima je da mogu da dodaju nestabilnost, a obećanu dodatnu zaštitu je teško kvantifikovati.

4.5 Sertifikati

Porodica standarda ISO/IEC 27000 pomaže organizacijama da zaštite informaciona sredstva. ISO/IEC 27001 posebno specificira skup najboljih praksi i detaljno opisuje listu bezbednosnih kontrola u vezi sa upravljanjem informacionim rizicima.

Iako Standard 27001 ne nalaže posebne kontrole bezbednosti informacija, okvir i kontrolna lista kontrola koje on postavlja omogućavaju kompanijama da obezbede sveobuhvatan i kontinuirano unapređujući model za upravljanje bezbednošću..

Još jedan popularan okvir je standard bezbednosti podataka industrije platnih kartica (PCI DSS). Čak i ako ne čuvate podatke o kreditnoj kartici, osnovni zahtevi su dobra lista bezbednosnih aktivnosti. Ako su vaši podaci posebno osetljivi, vaša organizacija bi trebalo da razmisli o sertifikovanju vaših bezbednosnih praksi. Sertifikati MongoDB Inc. i whitepapers za Atlas su korisna referenca na stranici Trust.

5 Zaključak

Ukratko, evo kontrolne liste svih koraka koje treba preduzeti da bi se obezbedila bezbedna implementacija MongoDB baze podataka:

1. Omogućite autentifikaciju (i kreirajte MongoDB korisnike sa najmanje privilegija neophodnih da dovrše svoje slučajeve korišćenja), i ako je dostupno, koristite Kerberos za spoljnu autentifikaciju.
2. Osigurajte mrežnu infrastrukturu blokiranjem dolaznih veza sa svih osim pouzdanih IP-adresa/mreža i blokiranjem svih nekritičnih portova.
3. Uverite se da većina korisnika na hostu može samo da čita, ali ne i da menja MongoDB podatke ili datoteke evidencije.
4. Konfigurišite šifrovanje između svih komponenti (interna autentifikacija, replikacija) i od klijenata sa dobro definisanim *x.509 sertifikatima*.
5. Šifrujte podatke u mirovanju šifrovanjem sistema datoteka ili šifrovanjem na nivou aplikacije pomoću redovno rotiranog glavnog ključa (Enterprise).
6. Omogućite evidenciju revizije koja se šalje na centralizovani server evidencije (Enterprise).
7. Organizovati datoteke evidencije da biste izbegli curenje osetljivih korisničkih podataka u log fajlovima (Enterprise).
8. Isključiti server-side JavaScript.
9. Forsirati SELinux.
10. Uklonite korisničke naloge kada više nisu potrebni.
11. Nadograditi na bilo koje manje izdanje MongoDB što je pre moguće da bi se iskoristile prednosti bezbednosnih zakrpa.
12. Nadgledati bezbednosna ažuriranja Alerts stranice.

Reference

1. https://www.mongodb.com/docs/manual/security/
2. https://www.mongodb.com/docs/atlas/setup-cluster-security/
3. MongoDB: The Definitive Guide: Powerful and Scalable Data Storage - Shannon Bradshaw, Kristina Chodorow
4. Node.js, MongoDB, React, React Native Full-Stack Fundamentals and Beyond - Eric Bush
5. MongoDB Fundamentals: A hands-on guide to using MongoDB and Atlas in the real world - Amit Phaltankar, Juned Ahsan
6. MongoDB Complete Guide: Develop Strong Understanding of Administering MongoDB - Manu Sharma
7. SEC-NoSQL: Towards Implementing High Performance Security-as-a-Service for NoSQL Databases - G. Dumindu Samaraweera, J. Morris Chang
8. https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/05.6-Testing_for_NoSQL_Injection.html
9. A Deep Dive into NoSQL Databases: The Use Cases and Applications - Pethuru Raj, Ganesh Chandra Deka